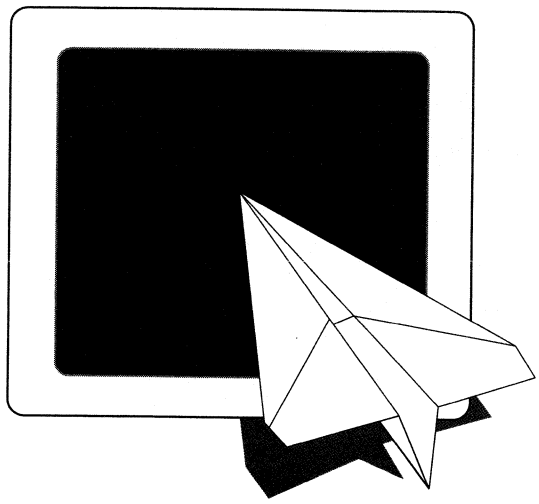


Xilinx Unified Libraries

Selection Guide

Design Elements

Attributes, Constraints, and Carry Logic

Index

XACT Libraries Guide

XACT Libraries Guide — April, 1994 (0401098 01)

Printed in U.S.A.

Preface

About This Manual

This manual describes Xilinx's Unified Libraries.

Before using this manual, you should be familiar with the operations that are common to all Xilinx's software tools: how to bring up the system, select a tool for use, specify operations, and manage design data. These topics are covered in the *XACT Reference Guide*.

Manual Contents

This libraries guide covers the following topics.

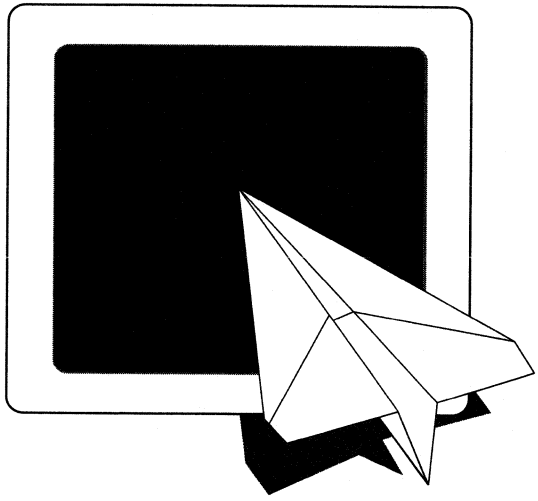
- Chapter 1, XILINX UNIFIED LIBRARIES, discusses the contents of the other chapters, general naming conventions, and performance issues.
- Chapter 2, SELECTION GUIDE, describes then lists design elements by function that are described in detail in the "Design Elements" chapter. Also included are tables that list Unified Libraries' replacements for existing and obsolete elements for each family.
- Chapter 3, DESIGN ELEMENTS, provides a graphic symbol, functional description, primitive versus macro table, truth table (when applicable), topology (when applicable), and schematics for macros of the design elements.
- Chapter 4, ATTRIBUTES, CONSTRAINTS, AND CARRY LOGIC, provides information on all attributes, Partition, Place, and Route (PPR) constraints, in particular, the relative location (RLOC) constraint, as well as Relationally Placed Macros (RPMs), and carry logic.

Conventions

The following conventions are used in this manual's syntactical statements:

`Courier font regular`	System messages or program files appear in regular Courier font.
`Courier font bold`	Literal commands that you must enter in syntax statements are in bold Courier font.
italic font	Variables that you replace in syntax statements are in italic font.
`[ ]`	Square brackets denote optional items or parameters. However, in bus specifications, such as bus [7:0], they are required.
`{ }`	Braces enclose a list of items from which you must choose one or more.
. . .	A vertical ellipsis indicates material that has been omitted.
. . .	A horizontal ellipsis indicates that the preceding can be repeated one or more times.
|	A vertical bar separates items in a list of choices.
↵	This symbol denotes a carriage return.

Contents



Xilinx Unified Libraries

XACT Libraries Guide

XACT Libraries Guide — April, 1994 (0401098 01)

Printed in U.S.A.

Chapter 1

Xilinx Unified Libraries

Xilinx maintains software libraries with thousands of functional design elements (primitives and macros) for different device architectures. New functional elements are assembled with each release of development system software. The latest catalog of design elements are known as "Unified Libraries." Elements in these libraries are common to all Xilinx device architectures. This "unified" approach means that you can use your circuit design created with "unified" library elements across all current Xilinx device architectures that recognize the element you are using.

Elements that exist in multiple architectures look and function the same, but their implementations might differ to make them more efficient for a particular architecture. A separate library still exists for each architecture and common symbols are duplicated in each one, which is necessary for simulation (especially board level) where timing depends on a particular architecture.

Note: OrCAD symbols differ in appearance. They do not support busing; each input and output pin appears on the symbol. Inputs and outputs only appear on the left and right sides of symbols, respectively (none appear on the top or bottom).

If you have active designs that were created with former Xilinx library primitives or macros, you may need to change references to the design elements that you were using to reflect the new Unified Libraries' elements.

Overview

The XACT Libraries Guide describes the primitive and macro logic elements available in the new Unified Libraries for XC2000, XC3000, XC4000, and XC7000 architectures. Common logic functions can be

implemented with these elements and more complex functions can be built by combining macros and primitives. Several hundred design elements (primitives and macros) are available across multiple device architectures, providing a common base for programmable logic designs.

This libraries guide provides a functional selection guide, describes the design elements, and addresses attributes, constraints, and carry logic.

This book is organized into four parts.

- Xilinx Unified Libraries
- Selection guide
- Design elements
- Constraints, attributes, and carry logic

Xilinx Unified Libraries

This chapter describes the Unified Libraries, briefly discusses the contents of the other chapters, the general naming conventions, and performance issues.

Selection Guide

The "Selection Guide" briefly describes, then tabularly lists the macro logic elements that are described in detail in the "Design Elements" chapter. The tables included in this section are organized into functional categories specifying all the available elements from each of the XC2000, XC3000, XC4000, and XC7000 families. Also included are tables that list Unified Libraries' replacements for existing and obsolete elements for each family.

Design Elements

Design elements are organized in alphanumeric order, with all numeric suffixes in ascending order. For example, ADD4 precedes ADD8 and FDR precedes FDRS.

The following information is provided for each library element.

- Graphic symbol
- Functional description

- Primitive versus macro table
- Truth table (when applicable)
- Topology (when applicable)
- Schematic for macros

Note: Schematics are included for each architecture if the implementation differs. Also, design elements with bused or multiple I/O pins typically include just one schematic — generally the 8-bit version. (In cases where no 8-bit version exists, an appropriate smaller or larger element serves as the schematic example.)

Attributes, Constraints, and Carry Logic

The "Attributes, Constraints, and Carry Logic" chapter provides information on all attributes and constraints. Attributes are instructions placed on symbols or nets in a schematic to indicate their placement, implementation, naming, directionality, and so forth. Constraints are a type of attribute used only to indicate where an element should be placed. The chapter describes Partition, Place, and Route (PPR) constraints, in particular, the relative location (RLOC) constraint, as well as Relationally Placed Macros (RPMs), and carry logic.

Naming Conventions

Examples of the general naming conventions for the Unified Libraries are shown in the following figures.

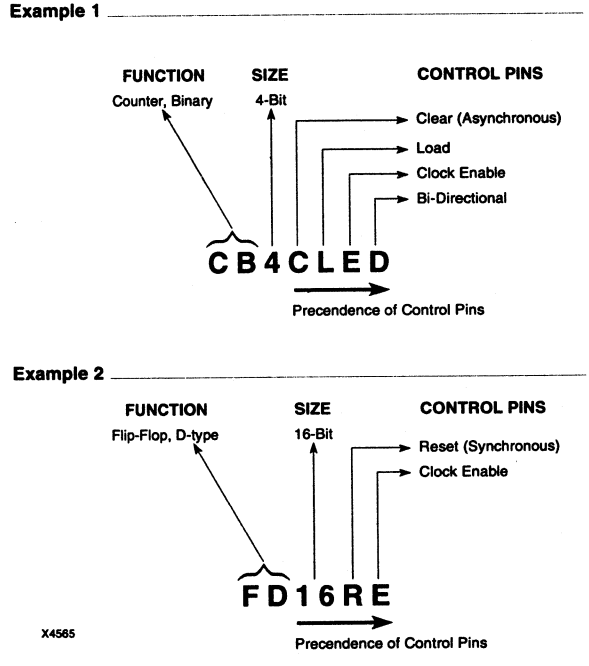


Figure 1-1 Naming Conventions

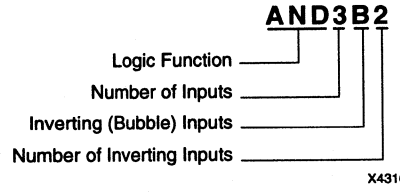


Figure 1-2 Combinatorial Naming Conventions

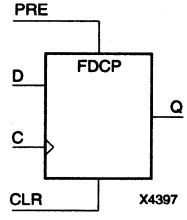
Refer to the Selection Guide for examples of functional component naming conventions.

Flip-Flop, Counter, and Register Performance

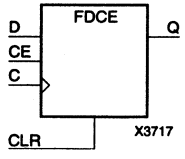
All counter, register, and storage functions are derived from the flip-flops (and latches in XC2000) available in the Configurable Logic Blocks (CLBs).

The D flip-flop is the basic building block for all four architectures. Differences occur from the availability of asynchronous Clear (CLR) and Preset (PRE) inputs, and the source of the synchronous control signals, such as, Clock Enable (CE), Clock (C), Load enable (L), synchronous Reset (R), and synchronous Set (S). The basic flip-flop configuration for each architecture follows.

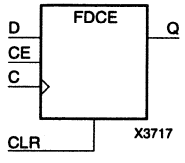
The basic XC2000 and XC7000 flip-flops have both Clear and Preset inputs.

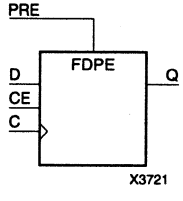


The XC3000 has a direct-connect Clock Enable input and a Clear input.



The XC4000 has a direct-connect Clock Enable input and a choice of either the Clear or the Preset inputs, but not both.





The asynchronous and synchronous control functions, when used, have a priority that is consistent across all devices and architectures. These inputs can be either active-High or active-Low as defined by the macro. The priority, from highest to lowest is as follows.

- Asynchronous Clear (C)
- Asynchronous Preset (PRE)
- Synchronous Set (S)
- Synchronous Reset (R)
- Load Enable (L)
- Shift Left/Right (LEFT)
- Clock Enable (CE)

Note: The asynchronous C and PRE inputs, by definition, have priority over all the synchronous control and clock inputs.

The Clock Enable (CE) function is implemented using two different methods in the Xilinx Unified Libraries; both are shown in the following figure. In method 1, CE is implemented by connecting the CE pin of the macro directly to the dedicated Enable Clock (EC) pin of the internal Configurable Logic Block (CLB) flip-flop. In method 2, CE is implemented using function generator logic. CE takes precedence over the L, S, and R inputs in method 1. CE has the same priority as the L, S, and R inputs in method 2. The method used in a particular macro is indicated in the macro's description.

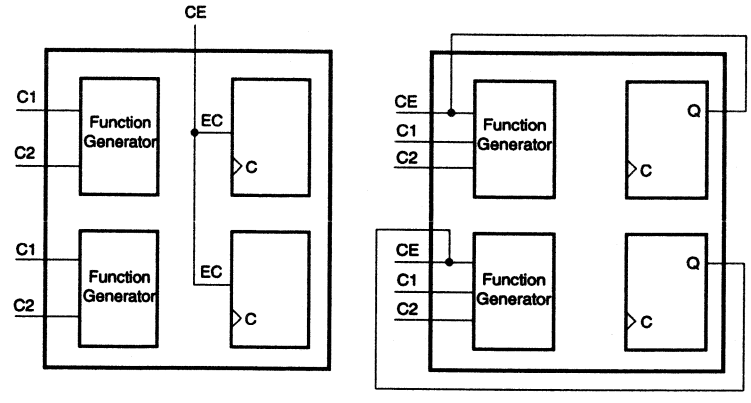
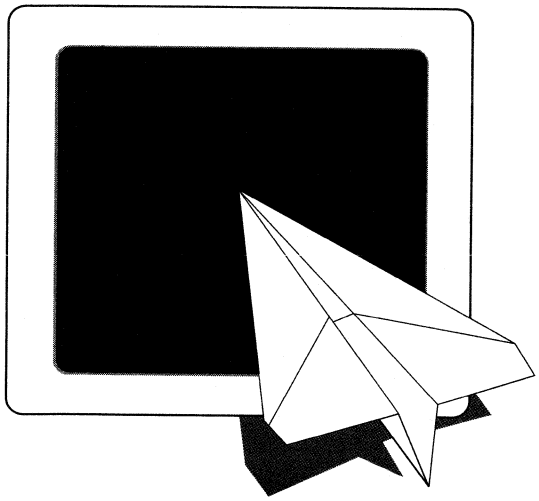


Figure 1-3 Clock Enable Implementation Methods



Selection Guide

XACT Libraries Guide

XACT Libraries Guide — April, 1994 (0401098 01)

Printed in U.S.A.

Chapter 2

Selection Guide

The Selection Guide briefly describes, then tabularly lists the macro logic elements that are described in detail in the "Design Elements" chapter. The tables included in this section are organized into functional categories specifying all the available macros from each of the XC2000, XC3000, XC4000, and XC7000 families. The tables categorize the elements into sub-categories based on similar functions. The sequence of each sub-category is based on an ascending order of complexity. The categories are as follows.

- Arithmetic functions
- Buffers
- Comparators
- Counters
- Data registers
- Decoders
- Edge decoders
- Encoders
- Flip-Flops
- General
- Input/output flip-flops
- Input/output functions
- Input latches
- Latches
- Logic primitives
- Map elements
- Memory elements

- Multiplexers
- PLD elements
- Shift registers
- Shifters

The elements from each architecture that provide the same function are listed adjacent to each other in the table, even though they might not have the same name. For particular elements, use the name specified for the architecture of interest.

Note: When converting your design between FPGA families, use macros that have equivalent functions in each of the families to minimize re-designing.

There are a number of standard TTL 7400-type functions in the XC2000, XC3000, XC4000, and XC7000 architectures. All 7400-type functions are in alphanumeric order starting with "X," and the numeric sequence uses ascending numbers following the "74" prefix. For example, X74_42 precedes X74_138.

Functional Categories

The following sections briefly describe, then tabularly list the Unified Libraries design element functions by category. Elements are listed in alphanumeric order according to architecture in each applicable architecture column. N/A means the element does not exist in that particular architecture.

Following these functional listings, replacement and obsolete elements are discussed.

Arithmetic Functions

There are three types of arithmetic functions: accumulators (ACC), adders (ADD), and adder/subtracters (ADSU). With an ADSU, either unsigned binary or twos-complement operations cause an overflow. If the result crosses the overflow boundary, an overflow is generated. Similarly, when the result crosses the carry-out boundary, a carry-out is generated. The following figure shows the ADSU carry-out and overflow boundaries.

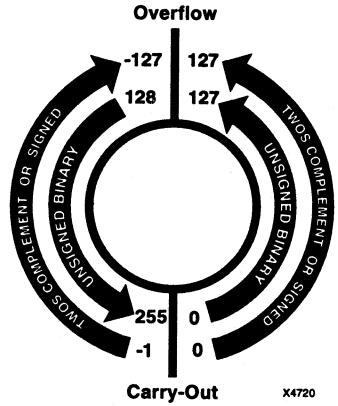


Figure 2-1 ADSU Carry-Out and Overflow Boundaries

XC2000	XC3000	XC4000	XC7000	Description
ACC1	N/A	N/A	ACC1	1-Bit Accumulator with Carry-In, Carry-Out, and Synchronous Reset
N/A	N/A	N/A	ACC1X1	1-Bit Accumulator with Carry-Out for EPLD
N/A	N/A	N/A	ACC1X2	1-Bit Accumulator with Carry-In and Carry-Out for EPLD
N/A	ACC4	ACC4	ACC4	4-Bit Accumulator with Carry-In, Carry-Out, and Synchronous Reset
N/A	N/A	N/A	ACC4X1	4-Bit Accumulator with Carry-Out for EPLD
N/A	N/A	N/A	ACC4X2	4-Bit Accumulator with Carry-In and Carry-Out for EPLD
N/A	ACC8	ACC8	ACC8	8-Bit Accumulator with Carry-In, Carry-Out, and Synchronous Reset

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	N/A	ACC8X1	8-Bit Accumulator with Carry-Out for EPLD
N/A	N/A	N/A	ACC8X2	8-Bit Accumulator with Carry-In and Carry-Out for EPLD
N/A	ACC16	ACC16	ACC16	16-Bit Accumulator with Carry-In, Carry-Out, and Synchronous Reset
N/A	N/A	N/A	ACC16X1	16-Bit Accumulator with Carry-Out for EPLD
N/A	N/A	N/A	ACC16X2	16-Bit Accumulator with Carry-In and Carry-Out for EPLD
ADD1	N/A	N/A	ADD1	1-Bit Full Adder with Carry-In and Carry-Out
N/A	N/A	N/A	ADD1X1	1-Bit Adder with Carry-Out for EPLD
N/A	N/A	N/A	ADD1X2	1-Bit Adder with Carry-In and Carry-Out for EPLD
N/A	ADD4	ADD4	ADD4	4-Bit Cascadable Full Adder with Carry-In and Carry-Out
N/A	N/A	N/A	ADD4X1	4-Bit Adder with Carry-Out for EPLD
N/A	N/A	N/A	ADD4X2	4-Bit Adder with Carry-In and Carry-Out for EPLD
N/A	ADD8	ADD8	ADD8	8-Bit Cascadable Full Adder with Carry-In and Carry-Out
N/A	N/A	N/A	ADD8X1	8-Bit Adder with Carry-Out for EPLD
N/A	N/A	N/A	ADD8X2	8-Bit Adder with Carry-In and Carry-Out for EPLD
N/A	ADD16	ADD16	ADD16	16-Bit Cascadable Full Adder with Carry-In and Carry-Out
N/A	N/A	N/A	ADD16X1	16-Bit Adder with Carry-Out for EPLD
N/A	N/A	N/A	ADD16X2	16-Bit Adder with Carry-In and Carry-Out for EPLD
ADSU1	N/A	N/A	ADSU1	1-Bit Adder/Substracter with Carry-In and Carry-Out
N/A	N/A	N/A	ADSU1X1	1-Bit Adder/Subtracter with Carry-Out for EPLD
N/A	N/A	N/A	ADSU1X2	1-Bit Adder/Subtracter with Carry-In and Carry-Out for EPLD
N/A	ADSU4	ADSU4	ADSU4	4-Bit Cascadable Adder/Subtracter with Carry-In and Carry-Out

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	N/A	ADSU4X1	4-Bit Adder/Subtracter with Carry-Out for EPLD
N/A	N/A	N/A	ADSU4X2	4-Bit Adder/Subtracter with Carry-In and Carry-Out for EPLD
N/A	ADSU8	ADSU8	ADSU8	8-Bit Adder/Subtracter with Carry-In, Carry-Out, and Overflow
N/A	N/A	N/A	ADSU8X1	8-Bit Adder/Subtracter with Carry-Out for EPLD
N/A	N/A	N/A	ADSU8X2	8-Bit Adder/Subtracter with Carry-In and Carry-Out for EPLD
N/A	ADSU16	ADSU16	ADSU16	16-Bit Adder/Subtracter with Overflow
N/A	N/A	N/A	ADSU16X1	16-Bit Adder/Subtracter with Carry-Out for EPLD
N/A	N/A	N/A	ADSU16X2	16-Bit Adder/Subtracter with Carry-In and Carry-Out for EPLD
X74_280	X74_280	X74_280	X74_280	9-Bit Odd/Even Parity Generator/ Checker
X74_283	X74_283	X74_283	X74_283	4-Bit Full Adder with Carry-In and Carry-Out

Buffers

The buffers in this section route high fan-out signals, 3-state signals, and clocks inside a PLD device. The "Input/Output Functions" section later in this chapter covers off-chip interface buffers.

XC2000	XC3000	XC4000	XC7000	Description
ACLK	ACLK	N/A	N/A	Alternate Clock Buffer
BUF	BUF	BUF	BUF	General Purpose Buffers
N/A	N/A	N/A	BUF4, BUF8, BUF16	
N/A	N/A	N/A	BUFCE	Global Clock-Enable Input Buffer for EPLD

XC2000	XC3000	XC4000	XC7000	Description
N/A	BUFE, BUFE4, BUFE8, BUFE16	BUFE, BUFE4, BUFE8, BUFE16	BUFE, BUFE4, BUFE8, BUFE16	Internal 3-State Buffers with Active-High Enable
N/A	N/A	N/A	BUFFOE	Global Fast-Output-Enable (FOE) Input Buffer for EPLD
BUFG	BUFG	BUFG	BUFG	Global Clock Buffer
N/A	N/A	BUFGP	BUFGP	Primary Global Buffer for Driving Clocks or Longlines (4 per device)
N/A	N/A	BUFGS	BUFGS	Secondary Global Buffer for Driving Clocks or Longlines
N/A	N/A	BUFOD	N/A	Open-Drain Buffer
N/A	BUFT, BUFT4, BUFT8, BUFT16	BUFT, BUFT4, BUFT8, BUFT16	BUFT, BUFT4, BUFT8, BUFT16	Internal 3-State Buffers with Active-Low Enable
GCLK	GCLK	N/A	N/A	Global Clock Buffer

Comparators

There are two types of comparators, identity (COMP) and magnitude (COMPM).

XC2000	XC3000	XC4000	XC7000	Description
COMP2	COMP2	COMP2	COMP2	2-Bit Identity Comparator
COMP4	COMP4	COMP4	COMP4	4-Bit Identity Comparator
COMP8	COMP8	COMP8	COMP8	8-Bit Identity Comparator
COMP16	COMP16	COMP16	COMP16	16-Bit Identity Comparator
COMPM2	COMPM2	COMPM2	COMPM2	2-Bit Magnitude Comparator
COMPM4	COMPM4	COMPM4	COMPM4	4-Bit Magnitude Comparator
COMPM8	COMPM8	COMPM8	COMPM8	8-Bit Magnitude Comparator
COMPM16	COMPM16	COMPM16	N/A	16-Bit Magnitude Comparator
N/A	N/A	COMPMC8	N/A	8-Bit Magnitude Comparator
N/A	N/A	COMPMC16	N/A	16-Bit Magnitude Comparator
X74_L85	X74_L85	X74_L85	X74_L85	4-Bit Expandable Magnitude Comparator

XC2000	XC3000	XC4000	XC7000	Description
X74_518	X74_518	X74_518	X74_518	8-Bit Identity Comparator with Active-Low Enable
X74_521	X74_521	X74_521	X74_521	8-Bit Identity Comparator with Active-Low Enable and Output

Counters

There are six types of counters with various synchronous and asynchronous inputs. The name of the counter defines the modulo or bit size, the counter type, and which control functions are included. The counter naming convention is shown in the following figure.

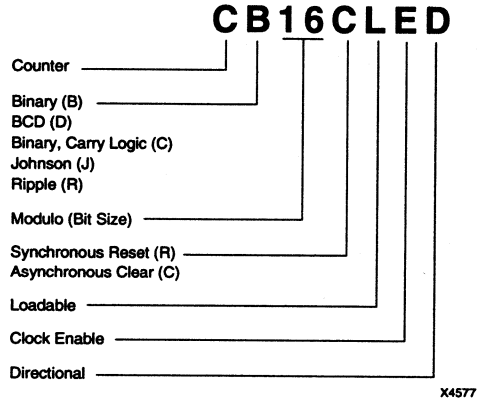


Figure 2-2 Counter Naming Convention

A carry-lookahead design accommodates large counters without extra gating. On TTL 7400-type counters with trickle clock enable (ENT), parallel clock enable (ENP), and ripple carry-out (RCO), both the ENT and ENP inputs must be High to count. ENT is propagated forward to enable RCO, which produces a High output with the approximate duration of the QA output. The following figure illustrates a carry-lookahead design.

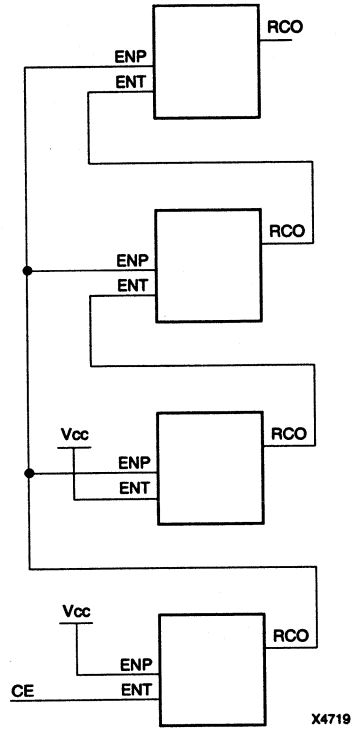


Figure 2-3 Carry-Lookahead Design

The RCO output of the first stage of the ripple carry is connected to the ENP input of the second stage and all subsequent stages. The RCO output of second stage and all subsequent stages is connected to the ENT input of the next stage. The ENT of the second stage is always enabled/tied to VCC. CE is always connected to the ENT input of the first stage. This cascading method allows the first stage of the ripple carry to be built as a prescaler. In other words, the first stage is built to count very fast.

XC2000	XC3000	XC4000	XC7000	Description
CB2CE	CB2CE	CB2CE	CB2CE	2-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear
CB2CLE	CB2CLE	CB2CLE	CB2CLE	2-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear
CB2CLED	CB2CLED	CB2CLED	CB2CLED	2-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear
CB2RE	CB2RE	CB2RE	CB2RE	2-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	N/A	CB2RLE	2-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	N/A	CB2X1	2-Bit Loadable Cascadable Bidirectional Binary Counter with Asynchronous Clear for EPLD
N/A	N/A	N/A	CB2X2	2-Bit Loadable Cascadable Bidirectional Binary Counter with Synchronous Reset for EPLD
CB4CE	CB4CE	CB4CE	CB4CE	4-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear
CB4CLE	CB4CLE	CB4CLE	CB4CLE	4-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear
CB4CLED	CB4CLED	CB4CLED	CB4CLED	4-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear
CB4RE	CB4RE	CB4RE	CB4RE	4-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	N/A	CB4RLE	4-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	N/A	CB4X1	4-Bit Loadable Cascadable Bidirectional Binary Counter with Asynchronous Clear for EPLD
N/A	N/A	N/A	CB4X2	4-Bit Loadable Cascadable Bidirectional Binary Counter with Synchronous Reset for EPLD
CB8CE	CB8CE	CB8CE	CB8CE	8-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear
CB8CLE	CB8CLE	CB8CLE	CB8CLE	8-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear
CB8CLED	CB8CLED	CB8CLED	CB8CLED	8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear
CB8RE	CB8RE	CB8RE	CB8RE	8-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	N/A	CB8RLE	8-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	N/A	CB8X1	8-Bit Loadable Cascadable Bidirectional Binary Counter with Asynchronous Clear for EPLD
N/A	N/A	N/A	CB8X2	8-Bit Loadable Cascadable Bidirectional Binary Counter with Synchronous Reset for EPLD
CB16CE	CB16CE	CB16CE	CB16CE	16-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear
CB16CLE	CB16CLE	CB16CLE	CB16CLE	16-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear

XC2000	XC3000	XC4000	XC7000	Description
CB16CLED	CB16CLED	CB16CLED	CB16CLED	16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear
CB16RE	CB16RE	CB16RE	CB16RE	16-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	N/A	CB16RLE	16-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	N/A	CB16X1	16-Bit Loadable Cascadable Bidirectional Binary Counter with Asynchronous Clear for EPLD
N/A	N/A	N/A	CB16X2	16-Bit Loadable Cascadable Bidirectional Binary Counter with Synchronous Reset for EPLD
N/A	N/A	CC8CE	N/A	8-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear
N/A	N/A	CC8CLE	N/A	8-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear
N/A	N/A	CC8CLED	N/A	8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear
N/A	N/A	CC8RE	N/A	8-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset
N/A	N/A	CC16CE	N/A	16-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear
N/A	N/A	CC16CLE	N/A	16-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	CC16CLED	N/A	16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear
N/A	N/A	CC16RE	N/A	16-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset
CD4CE	CD4CE	CD4CE	CD4CE	4-Bit Cascadable BCD Counter with Clock Enable and Asynchronous Clear
CD4CLE	CD4CLE	CD4CLE	CD4CLE	4-Bit Loadable Cascadable BCD Counter with Clock Enable and Asynchronous Clear
CD4RE	CD4RE	CD4RE	CD4RE	4-Bit Cascadable BCD Counter with Clock Enable and Synchronous Reset
CD4RLE	CD4RLE	CD4RLE	CD4RLE	4-Bit Loadable Cascadable BCD Counter with Clock Enable and Synchronous Reset
CJ4CE	CJ4CE	CJ4CE	CJ4CE	4-Bit Johnson Counter with Clock Enable and Asynchronous Clear
CJ4RE	CJ4RE	CJ4RE	CJ4RE	4-Bit Johnson Counter with Clock Enable and Synchronous Reset
CJ5CE	CJ5CE	CJ5CE	CJ5CE	5-Bit Johnson Counter with Clock Enable and Asynchronous Clear
CJ5RE	CJ5RE	CJ5RE	CJ5RE	5-Bit Johnson Counter with Clock Enable and Synchronous Reset
CJ8CE	CJ8CE	CJ8CE	CJ8CE	8-Bit Johnson Counter with Clock Enable and Asynchronous Clear
CJ8RE	CJ8RE	CJ8RE	CJ8RE	8-Bit Johnson Counter with Clock Enable and Synchronous Reset
CR8CE	CR8CE	CR8CE	CR8CE	8-Bit Negative-Edge Binary Ripple Counter with Clock Enable and Asynchronous Clear
CR16CE	CR16CE	CR16CE	CR16CE	16-Bit Negative-Edge Binary Ripple Counter with Clock Enable and Asynchronous Clear

XC2000	XC3000	XC4000	XC7000	Description
X74_160	X74_160	X74_160	X74_160	4-Bit Loadable Cascadable BCD Counter with Parallel and Trickle Enables and Asynchronous Clear
X74_161	X74_161	X74_161	X74_161	4-Bit Loadable Cascadable Binary Counter with Parallel and Trickle Enables and Asynchronous Clear
X74_162	X74_162	X74_162	X74_162	4-Bit Loadable Cascadable BCD Counter with Parallel and Trickle Enables and Synchronous Reset
X74_163	X74_163	X74_163	X74_163	4-Bit Loadable Cascadable Binary Counter with Parallel and Trickle Enables and Synchronous Reset
X74_168	X74_168	X74_168	X74_168	4-Bit Loadable Cascadable Bidirectional BCD Counter with Parallel and Trickle Enables
X74_390	X74_390	X74_390	X74_390	4-Bit BCD/Bi-Quinary Ripple Counter with Negative-Edge Clocks and Asynchronous Clear

Data Registers

There are three TTL 7400-type data registers designed to function exactly as the TTL elements for which they are named.

XC2000	XC3000	XC4000	XC7000	Description
X74_174	X74_174	X74_174	X74_174	6-Bit Data Register with Active-Low Asynchronous Clear
X74_273	X74_273	X74_273	X74_273	8-Bit Data Register with Active-Low Asynchronous Clear
X74_377	X74_377	X74_377	X74_377	8-Bit Data Register with Active-Low Clock Enable

Decoders

Decoder names, shown in the following figure, indicate the number of inputs and outputs and if an enable is available. Decoders with an enable can be used as multiplexers. This group includes some standard TTL 7400-type decoders whose names have an "X74" prefix.

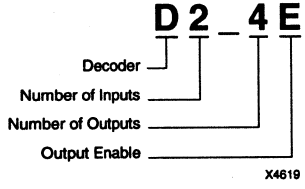


Figure 2-4 Decoder Naming Convention

XC2000	XC3000	XC4000	XC7000	Description
D2_4E	D2_4E	D2_4E	D2_4E	2- to 4-Line Decoder/Demultiplexer with Enable
D3_8E	D3_8E	D3_8E	D3_8E	3- to 8-Line Decoder/Demultiplexer with Enable
D4_16E	D4_16E	D4_16E	D4_16E	4- to 16-Line Decoder/Demultiplexer with Enable
X74_42	X74_42	X74_42	X74_42	4- to 10-Line BCD-to-Decimal Decoder with Active-Low Outputs

XC2000	XC3000	XC4000	XC7000	Description
X74_138	X74_138	X74_138	X74_138	3- to 8-Line Decoder/Demultiplexer with Active-Low Outputs and Three Enables
X74_139	X74_139	X74_139	X74_139	2- to 4-Line Decoder/Demultiplexer with Active-Low Outputs and Active-Low Enable
X74_154	X74_154	X74_154	X74_154	4- to 16-Line Decoder/Demultiplexer with Two Enables and Active-Low Outputs

Edge Decoders

Edge decoders are open-drain wired-AND gates that are available in different bit sizes.

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	DECODE4	N/A	4-Bit Active-Low Decoder
N/A	N/A	DECODE8	N/A	8-Bit Active-Low Decoder
N/A	N/A	DECODE16	N/A	16-Bit Active-Low Decoder

Encoders

There are two priority encoders (ENCPR) that function like the TTL 7400-type elements they are named after. There is a 10- to 4-line BCD encoder and an 8- to 3-line binary encoder.

XC2000	XC3000	XC4000	XC7000	Description
X74_147	X74_147	X74_147	X74_147	10- to 4-Line Priority Encoder with Active-Low Inputs and Outputs
X74_148	X74_148	X74_148	X74_148	8- to 3-Line Cascadable Priority Encoder with Active-Low Inputs and Outputs

Flip-Flops

There are three types of flip-flops (D, J-K, toggle) with various synchronous and asynchronous inputs. Some are available with inverted clock inputs and/or the ability to set in response to global set/reset rather than reset. The naming convention shown in the following figure provides a description for each flip-flop. D-type flip-flops are available in multiples of up to 16 in one macro.

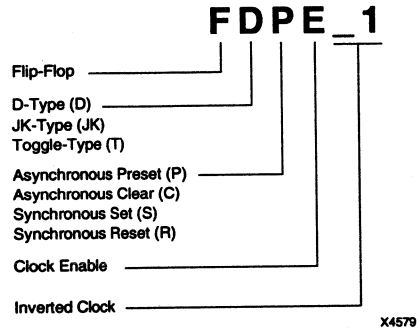


Figure 2-5 Flip-Flop Naming Convention

XC2000	XC3000	XC4000	XC7000	Description
FD	FD	FD	FD	Single and Multiple D Flip-Flops
N/A	N/A	N/A	FD4, FD8, FD16	
FD4CE	FD4CE	FD4CE	FD4CE	4-Bit Data Register with Clock Enable and Asynchronous Clear
FD4RE	FD4RE	FD4RE	FD4RE	4-Bit Data Register with Clock Enable and Synchronous Reset
FD8CE	FD8CE	FD8CE	FD8CE	8-Bit Data Register with Clock Enable and Asynchronous Clear
FD8RE	FD8RE	FD8RE	FD8RE	8-Bit Data Register with Clock Enable and Synchronous Reset
FD16CE	FD16CE	FD16CE	FD16CE	16-Bit Data Register with Clock Enable and Asynchronous Clear
FD16RE	FD16RE	FD16RE	FD16RE	16-Bit Data Register with Clock Enable and Synchronous Reset
FD_1	FD_1	FD_1	N/A	D Flip-Flop with Negative-Edge Clock

XC2000	XC3000	XC4000	XC7000	Description
FDC	FDC	FDC	FDC	D Flip-Flop with Asynchronous Clear
FDC_1	FDC_1	FDC_1	N/A	D Flip-Flop with Negative-Edge Clock and Asynchronous Clear
FDCE	FDCE	FDCE	FDCE	D Flip-Flop with Clock Enable and Asynchronous Clear
FDCE_1	FDCE_1	FDCE_1	N/A	D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Clear
FDCP	N/A	N/A	FDCP	D Flip-Flop with Asynchronous Preset and Clear
FDCPE	N/A	N/A	FDCPE	D Flip-Flop with Clock Enable and Asynchronous Preset and Clear
N/A	N/A	FDP	FDP	D Flip-Flop with Asynchronous Preset
N/A	N/A	FDP_1	N/A	D Flip-Flop with Negative-Edge Clock and Asynchronous Preset
N/A	N/A	FDPE	FDPE	D Flip-Flop with Clock Enable and Asynchronous Preset
N/A	N/A	FDPE_1	N/A	D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Preset
FDR	FDR	FDR	FDR	D Flip-Flop with Synchronous Reset
FDRE	FDRE	FDRE	FDRE	D Flip-Flop with Clock Enable and Synchronous Reset
FDRS	FDRS	FDRS	FDRS	D Flip-Flop with Synchronous Reset and Synchronous Set
FDRSE	FDRSE	FDRSE	FDRSE	D Flip-Flop with Synchronous Reset and Set and Clock Enable
FDS	FDS	FDS	FDS	D Flip-Flop with Synchronous Set
FDSE	FDSE	FDSE	FDSE	D Flip-Flop with Clock Enable and Synchronous Set
FDSR	FDSR	FDSR	FDSR	D Flip-Flop with Synchronous Set and Reset
FDSRE	FDSRE	FDSRE	FDSRE	D Flip-Flop with Synchronous Set and Reset and Clock Enable

XC2000	XC3000	XC4000	XC7000	Description
FJKC	FJKC	FJKC	FJKC	J-K Flip-Flop with Asynchronous Clear
FJKCE	FJKCE	FJKCE	FJKCE	J-K Flip-Flop with Clock Enable and Asynchronous Clear
FJKCP	N/A	N/A	FJKCP	J-K Flip-Flop with Asynchronous Clear and Preset
FJKCPE	N/A	N/A	FJKCPE	J-K Flip-Flop with Asynchronous Clear and Preset and Clock Enable
N/A	N/A	FJKP	FJKP	J-K Flip-Flop with Asynchronous Preset
N/A	N/A	FJKPE	FJKPE	J-K Flip-Flop with Clock Enable and Asynchronous Preset
FJKRSE	FJKRSE	FJKRSE	FJKRSE	J-K Flip-Flop with Clock Enable and Synchronous Reset and Set
FJKSRE	FJKSRE	FJKSRE	FJKSRE	J-K Flip-Flop with Clock Enable and Synchronous Set and Reset
FTC	FTC	FTC	FTC	Toggle Flip-Flop with Toggle Enable and Asynchronous Clear
FTCE	FTCE	FTCE	FTCE	Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear
FTCLE	FTCLE	FTCLE	FTCLE	Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear
FTCP	N/A	N/A	FTCP	Toggle Flip-Flop with Toggle Enable and Asynchronous Clear and Preset
FTCPE	N/A	N/A	FTCPE	Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear and Preset
FTCPLE	N/A	N/A	FTCPLE	Loadable Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear and Preset
N/A	N/A	FTP	FTP	Toggle Flip-Flop with Toggle Enable and Asynchronous Preset
N/A	N/A	FTPE	FTPE	Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Preset

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	FTPLE	FTPLE	Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Preset
FTRSE	FTRSE	FTRSE	FTRSE	Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set
FTRSLE	FTRSLE	FTRSLE	FTRSLE	Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set
FTSRE	FTSRE	FTSRE	FTSRE	Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Set and Reset
FTSRLE	FTSRLE	FTSRLE	FTSRLE	Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Set and Reset

General

General elements include FPGA configuration functions, oscillators, boundary-scan logic, and other functions not classified in other sections.

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	BSCAN	N/A	Boundary Scan Logic Control Circuit
CLB	CLB	N/A	N/A	CLB Configuration Symbol
GND	GND	GND	GND	Ground-Connection Signal Tag
GXTL	GXTL	N/A	N/A	Crystal Oscillator with ACLK Buffer
IOB	IOB	N/A	N/A	IOB Configuration Symbol
N/A	N/A	MD0	N/A	Mode 0/Input Pad Used for Readback Trigger Input
N/A	N/A	MD1	N/A	Mode 1/Output Pad Used for Readback Data Output
N/A	N/A	MD2	N/A	Mode 2/Input Pad
OSC	OSC	N/A	N/A	Crystal Oscillator

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	OSC4	N/A	Internal 5-Frequency Clock-Signal Generator
N/A	N/A	PULLDOWN	N/A	Resistor to GND for Input Pads
N/A	PULLUP	PULLUP	PULLUP	Resistor to VCC for Input PADs, Open-Drain and 3-State Outputs
N/A	N/A	READBACK	N/A	FPGA Bitstream Readback Controller
N/A	N/A	STARTUP	N/A	User Interface to Global Clock, Reset, and 3-State Controls
N/A	N/A	TCK	N/A	Boundary-Scan Test Clock Input Pad
N/A	N/A	TDI	N/A	Boundary-Scan Test Data Input Pad
N/A	N/A	TDO	N/A	Boundary-Scan Data Output Pad
N/A	TIMEGRP	TIMEGRP	N/A	Schematic-Level Table of Basic Timing Specification Groups
N/A	TIMESPEC	TIMESPEC	TIMESPEC	Schematic-Level Timing Requirement Table
N/A	N/A	TMS	N/A	Boundary-Scan Test Mode Select Input Pad

Input/Output Flip-Flops

Input/output flip-flops are configured in IOBs. They include flip-flops whose outputs are enabled by 3-state buffers, flip-flops that can be set upon global set/reset rather than reset, and flip-flops with inverted clock inputs. The naming convention specifies each flip-flop function and is illustrated in the following figure.

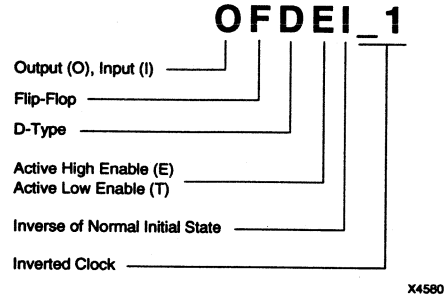


Figure 2-6 Input/Output Flip-Flop Naming Convention

XC2000	XC3000	XC4000	XC7000	Description
IFD, IFD4, IFD8, IFD16	IFD, IFD4, IFD8, IFD16	IFD, IFD4, IFD8, IFD16	IFD, IFD4, IFD8, IFD16	Single- and Multiple-Input D Flip-Flops
IFD_1	IFD_1	IFD_1	N/A	D Flip-Flop with Inverted Clock
N/A	N/A	N/A	IFDX1, IFD4X1, IFD8X1, IFD16X1	Input D Flip-Flops with Clock Enable for EPLD
N/A	N/A	IFDI	N/A	Input D Flip-Flop (Asynchronous Set)
N/A	N/A	IFDI_1	N/A	D Flip-Flop with Inverted Clock (Asynchronous Set)
N/A	OFD, OFD4, OFD8, OFD16	OFD, OFD4, OFD8, OFD16	OFD, OFD4, OFD8, OFD16	Single- and Multiple-Output D Flip-Flops
N/A	OFD_1	OFD_1	N/A	Output D Flip-Flop with Inverted Clock

XC2000	XC3000	XC4000	XC7000	Description
N/A	OFDE, OFDE4, OFDE8, OFDE16	OFDE, OFDE4, OFDE8, OFDE16	OFDE, OFDE4, OFDE8, OFDE16	D Flip-Flops with Active-High 3-State Output Buffers
N/A	OFDE_1	OFDE_1	N/A	D Flip-Flop with Active-High 3-State Output Buffer and Inverted Clock
N/A	N/A	OFDEI	N/A	D Flip-Flop with Active-High 3-State Output Buffer (Asynchronous Set)
N/A	N/A	OFDEI_1	N/A	D Flip-Flop with Active-High 3-State Output Buffer and Inverted Clock (Asynchronous Set)
N/A	N/A	OFDI	N/A	Output D Flip-Flop (Asynchronous Set)
N/A	N/A	OFDI_1	N/A	Output D Flip-Flop with Inverted Clock (Asynchronous Set)
N/A	OFDT, OFDT4, OFDT8, OFDT16	OFDT, OFDT4, OFDT8, OFDT16	OFDT, OFDT4, OFDT8, OFDT16	Single and Multiple D Flip-Flops with Active-Low 3-State Output Buffers
N/A	OFDT_1	OFDT_1	N/A	D Flip-Flop with Active-Low 3-State Output Buffer and Inverted Clock
N/A	N/A	OFDTI	N/A	D Flip-Flop with Active-Low 3-State Output Buffer (Asynchronous Set)
N/A	N/A	OFDTI_1	N/A	D Flip-Flop with Active-Low 3-State Output Buffer and Inverted Clock (Asynchronous Set)

Input/Output Functions

Input/Output Block (IOB) resources are configured into various I/O primitives and macros for convenience, such as, output buffers (OBUFs) and output buffers with an enable (OBUFEs). Pads used to connect the circuit to PLD device pins are also included.

XC2000	XC3000	XC4000	XC7000	Description
IBUF, IBUF4, IBUF8, IBUF16	IBUF, IBUF4, IBUF8, IBUF16	IBUF, IBUF4, IBUF8, IBUF16	IBUF, IBUF4, IBUF8, IBUF16	Single- and Multiple-Input Buffers
IOPAD, IOPAD4, IOPAD8, IOPAD16	IOPAD, IOPAD4, IOPAD8, IOPAD16	IOPAD, IOPAD4, IOPAD8, IOPAD16	IOPAD, IOPAD4, IOPAD8, IOPAD16	Single- and Multiple-Input/ Output Pads
IPAD, IPAD4, IPAD8, IPAD16	IPAD, IPAD4, IPAD8, IPAD16	IPAD, IPAD4, IPAD8, IPAD16	IPAD, IPAD4, IPAD8, IPAD16	Single- and Multiple-Input Pads
OBUF, OBUF4, OBUF8, OBUF16	OBUF, OBUF4, OBUF8, OBUF16	OBUF, OBUF4, OBUF8, OBUF16	OBUF, OBUF4, OBUF8, OBUF16	Single- and Multiple-Output Buffers
OBUFE, OBUFE4, OBUFE8, OBUFE16	OBUFE, OBUFE4, OBUFE8, OBUFE16	OBUFE, OBUFE4, OBUFE8, OBUFE16	OBUFE, OBUFE4, OBUFE8, OBUFE16	3-State Output Buffers with Active-High Fast Output Enable
N/A	N/A	N/A	OBUFEX1, OBUFE4X1, OBUFE8X1, OBUFEX2	3-State Output Buffers with Active-High Fast Output Enable for EPLD
OBUFT, OBUFT4, OBUFT8, OBUFT16	OBUFT, OBUFT4, OBUFT8, OBUFT16	OBUFT, OBUFT4, OBUFT8, OBUFT16	OBUFT, OBUFT4, OBUFT8, OBUFT16	Single and Multiple 3-State Output Buffers with Active-Low Enable
OPAD, OPAD4, OPAD8, OPAD16	OPAD, OPAD4, OPAD8, OPAD16	OPAD, OPAD4, OPAD8, OPAD16	OPAD, OPAD4, OPAD8, OPAD16	Single- and Multiple-Output Pads

XC2000	XC3000	XC4000	XC7000	Description
UPAD	UPAD	UPAD	UPAD	Connects the I/O Node of an IOB to the Internal PLD Circuit
VCC	VCC	VCC	VCC	VCC Connection Signal Tag

Input Latches

Single and multiple input latches can hold transient data entering a chip. Input latches use the same naming convention as I/O flip-flops.

XC2000	XC3000	XC4000	XC7000	Description
N/A	ILD, ILD4, ILD8, ILD16	ILD, ILD4, ILD8, ILD16	ILD, ILD4, ILD8, ILD16	Input Transparent Data Latches
N/A	ILD_1	ILD_1	N/A	Transparent Input Data Latch with Inverted Gate
N/A	N/A	ILDI	N/A	Input Transparent Data Latch (Asynchronous Set)
N/A	N/A	ILDI_1	N/A	Transparent Input Data Latch with Inverted Gate (Synchronous Set)

Latches

Latches (LD) are only available in the XC2000 and XC7000 architectures. XC3000 and XC4000 latches that existed in previous macro libraries are not recommended for new designs.

XC2000	XC3000	XC4000	XC7000	Description
LD	N/A	N/A	LD	Single and Multiple Transparent Data Latches
N/A	N/A	N/A	LD4, LD8, LD16	
LD_1	N/A	N/A	N/A	Transparent Data Latch with Inverted Gate
LDC	N/A	N/A	N/A	Transparent Data Latch with Asynchronous Clear
LD4CE, LD8CE, LD16CE	N/A	N/A	N/A	Transparent Data Latches with Asynchronous Clear and Clock Enable

XC2000	XC3000	XC4000	XC7000	Description
LDCP	N/A	N/A	N/A	Transparent Data Latch with Asynchronous Clear and Preset
LDCPE	N/A	N/A	N/A	Transparent Data Latch with Asynchronous Clear and Preset and Clock Enable
LDC_1	N/A	N/A	N/A	Transparent Data Latch with Asynchronous Clear and Inverted Gate Input

Logic Primitives

Combinatorial logic gates that implement the basic Boolean functions are available in XC2000, XC3000, XC4000, and XC7000 architectures with up to five inputs in all combinations of inverted and non-inverted inputs, and with six to nine inputs non-inverted.

XC2000	XC3000	XC4000	XC7000	Description
AND2, AND2B1, AND2B2, AND3, AND3B1, AND3B2, AND3B3, AND4, AND4B1, AND4B2, AND4B3, AND4B4, AND5, AND5B1, AND5B2, AND5B3, AND5B4, AND5B5, AND6, AND7, AND8, AND9	AND2, AND2B1, AND2B2, AND3, AND3B1, AND3B2, AND3B3, AND4, AND4B1, AND4B2, AND4B3, AND4B4, AND5, AND5B1, AND5B2, AND5B3, AND5B4, AND5B5, AND6, AND7, AND8, AND9	AND2, AND2B1, AND2B2, AND3, AND3B1, AND3B2, AND3B3, AND4, AND4B1, AND4B2, AND4B3, AND4B4, AND5, AND5B1, AND5B2, AND5B3, AND5B4, AND5B5, AND6, AND7, AND8, AND9	AND2, AND2B1, AND2B2, AND3, AND3B1, AND3B2, AND3B3, AND4, AND4B1, AND4B2, AND4B3, AND4B4, AND5, AND5B1, AND5B2, AND5B3, AND5B4, AND5B5, AND6, AND7, AND8, AND9	2- to 9-Input AND Gates with Inverted and Non-Inverted Inputs

XC2000	XC3000	XC4000	XC7000	Description
INV, INV4, INV8, INV16	INV, INV4, INV8, INV16	INV, INV4, INV8, INV16	INV, INV4, INV8, INV16	Single and Multiple Inverters
NAND2, NAND2B1, NAND2B2, NAND3, NAND3B1, NAND3B2, NAND3B3, NAND4, NAND4B1, NAND4B2, NAND4B3, NAND4B4, NAND5, NAND5B1, NAND5B2, NAND5B3, NAND5B4, NAND5B5, NAND6, NAND7, NAND8, NAND9	NAND2, NAND2B1, NAND2B2, NAND3, NAND3B1, NAND3B2, NAND3B3, NAND4, NAND4B1, NAND4B2, NAND4B3, NAND4B4, NAND5, NAND5B1, NAND5B2, NAND5B3, NAND5B4, NAND5B5, NAND6, NAND7, NAND8, NAND9	NAND2, NAND2B1, NAND2B2, NAND3, NAND3B1, NAND3B2, NAND3B3, NAND4, NAND4B1, NAND4B2, NAND4B3, NAND4B4, NAND5, NAND5B1, NAND5B2, NAND5B3, NAND5B4, NAND5B5, NAND6, NAND7, NAND8, NAND9	NAND2, NAND2B1, NAND2B2, NAND3, NAND3B1, NAND3B2, NAND3B3, NAND4, NAND4B1, NAND4B2, NAND4B3, NAND4B4, NAND5, NAND5B1, NAND5B2, NAND5B3, NAND5B4, NAND5B5, NAND6, NAND7, NAND8, NAND9	2- to 9-Input NAND Gates with Inverted and Non-Inverted Inputs

XC2000	XC3000	XC4000	XC7000	Description
NOR2, NOR2B1, NOR2B2, NOR3, NOR3B1, NOR3B2, NOR3B3, NOR4, NOR4B1, NOR4B2, NOR4B3, NOR4B4, NOR5, NOR5B1, NOR5B2, NOR5B3, NOR5B4, NOR5B5, NOR6, NOR7, NOR8, NOR9	NOR2, NOR2B1, NOR2B2, NOR3, NOR3B1, NOR3B2, NOR3B3, NOR4, NOR4B1, NOR4B2, NOR4B3, NOR4B4, NOR5, NOR5B1, NOR5B2, NOR5B3, NOR5B4, NOR5B5, NOR6, NOR7, NOR8, NOR9	NOR2, NOR2B1, NOR2B2, NOR3, NOR3B1, NOR3B2, NOR3B3, NOR4, NOR4B1, NOR4B2, NOR4B3, NOR4B4, NOR5, NOR5B1, NOR5B2, NOR5B3, NOR5B4, NOR5B5, NOR6, NOR7, NOR8, NOR9	NOR2, NOR2B1, NOR2B2, NOR3, NOR3B1, NOR3B2, NOR3B3, NOR4, NOR4B1, NOR4B2, NOR4B3, NOR4B4, NOR5, NOR5B1, NOR5B2, NOR5B3, NOR5B4, NOR5B5, NOR6, NOR7, NOR8, NOR9	2- to 9-Input NOR Gates with Inverted and Non-Inverted Inputs

XC2000	XC3000	XC4000	XC7000	Description
OR2, OR2B1, OR2B2, OR3, OR3B1, OR3B2, OR3B3, OR4, OR4B1, OR4B2, OR4B3, OR4B4, OR5, OR5B1, OR5B2, OR5B3, OR5B4, OR5B5, OR6, OR7, OR8, OR9	OR2, OR2B1, OR2B2, OR3, OR3B1, OR3B2, OR3B3, OR4, OR4B1, OR4B2, OR4B3, OR4B4, OR5, OR5B1, OR5B2, OR5B3, OR5B4, OR5B5, OR6, OR7, OR8, OR9	OR2, OR2B1, OR2B2, OR3, OR3B1, OR3B2, OR3B3, OR4, OR4B1, OR4B2, OR4B3, OR4B4, OR5, OR5B1, OR5B2, OR5B3, OR5B4, OR5B5, OR6, OR7, OR8, OR9	OR2, OR2B1, OR2B2, OR3, OR3B1, OR3B2, OR3B3, OR4, OR4B1, OR4B2, OR4B3, OR4B4, OR5, OR5B1, OR5B2, OR5B3, OR5B4, OR5B5, OR6, OR7, OR8, OR9	2- to 9-Input OR Gates with Inverted and Non-Inverted Inputs
SOP3, SOP3B1A, SOP3B1B, SOP3B2A, SOP3B2B, SOP3B3, SOP4, SOP4B1, SOP4B2A, SOP4B2B, SOP4B3, SOP4B4	SOP3, SOP3B1A, SOP3B1B, SOP3B2A, SOP3B2B, SOP3B3, SOP4, SOP4B1, SOP4B2A, SOP4B2B, SOP4B3, SOP4B4	SOP3, SOP3B1A, SOP3B1B, SOP3B2A, SOP3B2B, SOP3B3, SOP4, SOP4B1, SOP4B2A, SOP4B2B, SOP4B3, SOP4B4	SOP3, SOP3B1A, SOP3B1B, SOP3B2A, SOP3B2B, SOP3B3, SOP4, SOP4B1, SOP4B2A, SOP4B2B, SOP4B3, SOP4B4	Sum of Products
N/A	N/A	WAND1, WAND4, WAND8, WAND16	N/A	Open-Drain Buffers

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	WOR2AND	N/A	2-Input OR Gate with Wired-AND Open-Drain Buffer Output
XNOR2, XNOR3, XNOR4, XNOR5, XNOR6, XNOR7, XNOR8, XNOR9	XNOR2, XNOR3, XNOR4, XNOR5, XNOR6, XNOR7, XNOR8, XNOR9	XNOR2, XNOR3, XNOR4, XNOR5, XNOR6, XNOR7, XNOR8, XNOR9	XNOR2, XNOR3, XNOR4, XNOR5, XNOR6, XNOR7, XNOR8, XNOR9	2- to 9-Input XNOR Gates with Non-Inverted Inputs
XOR2, XOR3, XOR4, XOR5, XOR6, XOR7, XOR8, XOR9	XOR2, XOR3, XOR4, XOR5, XOR6, XOR7, XOR8, XOR9	XOR2, XOR3, XOR4, XOR5, XOR6, XOR7, XOR8, XOR9	XOR2, XOR3, XOR4, XOR5, XOR6, XOR7, XOR8, XOR9	2- to 9-Input XOR Gates with Non-Inverted Inputs

Map Elements

Map elements are used in conjunction with logic symbols to constrain the logic to particular CLBs or particular F or H function generators.

XC2000	XC3000	XC4000	XC7000	Description
CLBMAP	CLBMAP	N/A	N/A	Logic Partitioning Control Symbol
N/A	N/A	FMAP	N/A	F Function Generator Partitioning Control Symbol
N/A	N/A	HMAP	N/A	Random-Logic Design Constraint Symbol

Memory Elements

The XC4000 architecture has a number of static RAM configurations defined as macros. These 16- or 32-word RAMs are 1, 2, 4, and 8 bits wide. There are also two ROMs in the XC4000 architecture, 16X1 and 32X1. ROMs only exist in XC4000.

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	RAM16X1	N/A	16-Deep by 1-Wide Static RAM
N/A	N/A	RAM16X2	N/A	16-Deep by 2-Wide Static RAM
N/A	N/A	RAM16X4	N/A	16-Deep by 4-Wide Static RAM
N/A	N/A	RAM16X8	N/A	16-Deep by 8-Wide Static RAM
N/A	N/A	RAM32X1	N/A	32-Deep by 1-Wide Static RAM
N/A	N/A	RAM32X2	N/A	32-Deep by 2-Wide Static RAM
N/A	N/A	RAM32X4	N/A	32-Deep by 4-Wide Static RAM
N/A	N/A	RAM32X8	N/A	32-Deep by 8-Wide Static RAM
N/A	N/A	ROM16X1	N/A	16-Deep by 1-Wide ROM
N/A	N/A	ROM32X1	N/A	32-Deep by 1-Wide ROM

Multiplexers

The multiplexer naming convention shown in the following figure, indicates the number of inputs and outputs and if an enable is available. There are a number of TTL 7400-type multiplexers that have active-Low or inverted outputs.

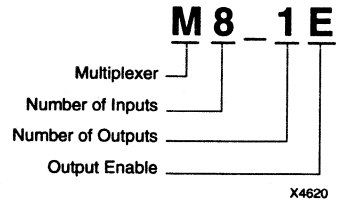


Figure 2-7 Multiplexer Naming Convention

XC2000	XC3000	XC4000	XC7000	Description
M2_1	M2_1	M2_1	M2_1	2-to-1 Multiplexer
M2_1B1	M2_1B1	M2_1B1	M2_1B1	2-to-1 Multiplexer with D0 Inverted
M2_1B2	M2_1B2	M2_1B2	M2_1B2	2-to-1 Multiplexer with D0 and D1 Inverted
M2_1E	M2_1E	M2_1E	M2_1E	2-to-1 Multiplexer with Enable
M4_1E	M4_1E	M4_1E	M4_1E	4-to-1 Multiplexer with Enable
M8_1E	M8_1E	M8_1E	M8_1E	8-to-1 Multiplexer with Enable
M16_1E	M16_1E	M16_1E	M16_1E	16-to-1 Multiplexer with Enable
X74_150	X74_150	X74_150	X74_150	16-to-1 Multiplexer with Active-Low Enable and Output
X74_151	X74_151	X74_151	X74_151	8-to-1 Multiplexer with Active-Low Enable and Complementary Outputs
X74_152	X74_152	X74_152	X74_152	8-to-1 Multiplexer with Active-Low Output
X74_153	X74_153	X74_153	X74_153	Dual 4-to-1 Multiplexer with Active-Low Enables and Common Select Input
X74_157	X74_157	X74_157	X74_157	Quadruple 2-to-1 Multiplexer with Common Select and Active-Low Enable

XC2000	XC3000	XC4000	XC7000	Description
X74_158	X74_158	X74_158	X74_158	Quadruple 2-to-1 Multiplexer with Common Select, Active-Low Enable, and Active-Low Outputs
X74_298	X74_298	X74_298	X74_298	Quadruple 2-Input Multiplexer with Storage and Negative-Edge Clock
X74_352	X74_352	X74_352	X74_352	Dual 4-to-1 Multiplexer with Active-Low Enables and Outputs

PLD Elements

PLD elements represent custom logic functions that are defined by an equation file in EPLD designs.

XC2000	XC3000	XC4000	XC7000	Description
N/A	N/A	N/A	PL20PIN	Generic PLD Symbols for EPLD
N/A	N/A	N/A	PL24PIN	
N/A	N/A	N/A	PL48PIN	
N/A	N/A	N/A	PL20V8	20V8-Compatible PLD Symbol for EPLD
N/A	N/A	N/A	PL22V10	22V10-Compatible PLD Symbol for EPLD
N/A	N/A	N/A	PLFB9	EPLD High-Density Function Block PLD Symbol
N/A	N/A	N/A	PLFFB9	EPLD Fast Function Block PLD Symbol

Shift Registers

Shift registers are available in a variety of sizes and capabilities. The naming convention shown in the following figure illustrates available features.

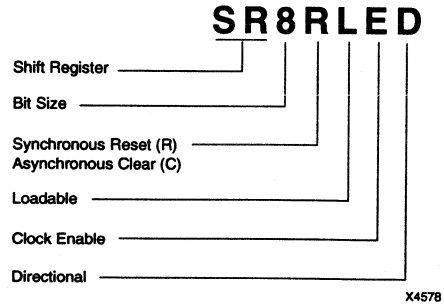


Figure 2-8 Shift Register Naming Convention

XC2000	XC3000	XC4000	XC7000	Description
SR4CE	SR4CE	SR4CE	SR4CE	4-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear
SR4CLE	SR4CLE	SR4CLE	SR4CLE	4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear
SR4CLED	SR4CLED	SR4CLED	SR4CLED	4-Bit Shift Register with Clock Enable and Asynchronous Clear
SR4RE	SR4RE	SR4RE	SR4RE	4-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
SR4RLE	SR4RLE	SR4RLE	SR4RLE	4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
SR4RLED	SR4RLED	SR4RLED	SR4RLED	4-Bit Shift Register with Clock Enable and Synchronous Reset
SR8CE	SR8CE	SR8CE	SR8CE	8-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear

XC2000	XC3000	XC4000	XC7000	Description
SR8CLE	SR8CLE	SR8CLE	SR8CLE	8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear
SR8CLED	SR8CLED	SR8CLED	SR8CLED	8-Bit Shift Register with Clock Enable and Asynchronous Clear
SR8RE	SR8RE	SR8RE	SR8RE	8-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
SR8RLE	SR8RLE	SR8RLE	SR8RLE	8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
SR8RLED	SR8RLED	SR8RLED	SR8RLED	8-Bit Shift Register with Clock Enable and Synchronous Reset
SR16CE	SR16CE	SR16CE	SR16CE	16-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear
SR16CLE	SR16CLE	SR16CLE	SR16CLE	16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear
SR16CLED	SR16CLED	SR16CLED	SR16CLED	16-Bit Shift Register with Clock Enable and Asynchronous Clear
SR16RE	SR16RE	SR16RE	SR16RE	16-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
SR16RLE	SR16RLE	SR16RLE	SR16RLE	16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
SR16RLED	SR16RLED	SR16RLED	SR16RLED	16-Bit Shift Register with Clock Enable and Synchronous Reset
X74_164	X74_164	X74_164	X74_164	8-Bit Serial-In Parallel-Out Shift Register with Active-Low Asynchronous Clear

XC2000	XC3000	XC4000	XC7000	Description
X74_165S	X74_165S	X74_165S	X74_165S	8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable
X74_194	X74_194	X74_194	X74_194	4-Bit Loadable Directional Serial/ Parallel-In Parallel-Out Shift Register
X74_195	X74_195	X74_195	X74_195	4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register

Shifters

Shifters are barrel shifters (BRLSHFT) of four and eight bits.

XC2000	XC3000	XC4000	XC7000	Description
N/A	BRLSHFT4	BRLSHFT4	BRLSHFT4	4-Bit Barrel Shifter
N/A	BRLSHFT8	BRLSHFT8	BRLSHFT8	8-Bit Barrel Shifter

Obsolete Macros

Xilinx maintains software libraries with thousands of functional design elements (primitives and macros) for different device architectures. When new elements are introduced that can provide additional functions, greater flexibility, increased speed, or enhanced system performance, it is necessary to remove or replace existing elements.

In some cases, design elements in the following tables have been obsoleted because their names changed to conform with the Unified Libraries' naming conventions. In other cases, duplicate functions have been eliminated. If you want a function that appears in the following tables, and an exact or functionally similar replacement does not exist, check the appropriate functional table listed in the Selection Guide, earlier in this chapter to determine the appropriate current macro.

The Unified Libraries make certain types of elements obsolete for the following reasons.

- Simple gates with names ending in B.

  (Bs indicate inversion of all inputs) The gates are still available as options for the generic macro (for example, for AND3B, refer to AND).
- Some redundancies have been eliminated.
- Some macros have been eliminated because they were meaningless inside an FPGA (for example, X74-240).
- Some macros have been eliminated because they were inefficient or had sub–optimal implementation.

If you have active designs that were created with former Xilinx libraries' primitives or macros, you may need to change references to the design elements that you were using to reflect the new Unified Libraries elements.

The following tables list Unified Libraries exact replacements and substitutions for existing elements that you can use to update your designs. Exact replacements are just that; you can use them for exactly the same function(s) as before. Substitutions provide at least the same functionality, but may afford additional advantages. Elements listed as obsolete are not recommended for new designs. They can still be found in some macro libraries, but support for them is being discontinued.

The elements are listed in alphanumeric order by architecture. Previous library element names appear in the left-most column followed by their exact Unified Libraries' replacement, if available. If an exact replacement does not exist, the closest substitution/an element with similar functions is provided in the third column. If you are not sure of the function provided by an exact replacement or substitution, refer to the Selection Guide, earlier in this chapter. Macro functions that are no longer supported are indicated in the Obsolete column.

XC2000 Replacement and Obsolete Macro Functions

Existing XC2000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
ASHEET	ASHEETP		
BPAD	IOPAD		
BSHEET	BSHEETL		
C2BCR		CB2RE	
C2BCRD		CB2CE	
C2BP		CB2CLE	
C2BR		CB2RE	
C2BRD		CB2CLE	
C4BCP		CB4CLE	
C4BCR	CB2RE		
C4BCRD	CB2CE		
C4JCR		CJ4RE	
C6JCR		CJ4RE	
C8BCP		CB4CLE	
C8BCR		CB4RE	
C8BCRD		CB4CLE	
C8JCR	CJ4RE		
C10BCPRD	CD4CLE		
C10BCRD	CD4CE		
C10BPRD		CD4CLE	
C10JCR	CJ5RE		
C12JCR		CJ8RE	
C16BARD		CB4CE	
C16BCPR		CB4CLE	
C16BCPRD	CB4CLE		
C16BCRD	CB4CE		
C16BPRD		CB4CLE	
C16BUDRD	CB4CLED		

Existing XC2000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
C16JCR	CJ8RE		
C256FCRD		CB8CLED	
CSHEET	CSHEETL		
D2-4		D2_4E	
D2-4E	D2_4E		
D3-8		D3_8E	
D3-8E	D3_8E		
DFF	FD		
DLAT	LDCP		
DSHEET	DSHEETL		
ESHEET	ESHEETL		
FDC		FDCE	
FDCR	FDRE		
FDCS	FDSE		
FDM			Obsolete
FDMR			Obsolete
FDMRD			Obsolete
FDMS			Obsolete
FDMSD			Obsolete
FDRD	FDC		
FDSD		FDCP	
FDSRD	FDCP		
FJK		FJKC	
FJKRD	FJKC		
FJKS		FJKSRE	
FJKSD		FJKCP	
FJKSRD	FJKCP		
FRS			Obsolete
FSR			Obsolete
FT		FTC	

Existing XC2000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
FT0		FTC	
FT0R		FTRSE	
FT2		FTCE	
FT2R		FTRSE	
FTP		FTCP	
FTPRD		FTRSLE	
FTR		FTSRE	
FTRD	FTC		
FTS		FTSRE	
GADD	ADD1		
GCOMP	COMP2		
GEQGT			Obsolete
GMAJ			Obsolete
GMUX	M2_1		
GOSC			Obsolete
GPAR			Obsolete
GXOR	XOR2		
GXOR2			Obsolete
INFF	IFD		
LDM			Obsolete
LDMRD			Obsolete
LDMSD			Obsolete
LDRD	LDC		
LDSD	LDP		
LDSRD	LDCP		
M3-1		M4_1	
M3-1E		M4_1E	
M4-1	M4_1		
M4-1E	M4_1E		
M8-1	M8_1		

Existing XC2000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
M8-1E	M8_1E		
NDFF			Obsolete
OBUFZ	OBUFT		
OUTFF	OFD		
PAD		IOPAD	
PAL2RA10			Obsolete
PAL6L16A			Obsolete
PAL8L14A			Obsolete
PAL10H8			Obsolete
PAL10H20			Obsolete
PAL10L8			Obsolete
PAL12H6			Obsolete
PAL12L6			Obsolete
PAL12L10			Obsolete
PAL14H4			Obsolete
PAL14L4			Obsolete
PAL14L8			Obsolete
PAL16A4			Obsolete
PAL16C1			Obsolete
PAL16H2			Obsolete
PAL16L2			Obsolete
PAL16L6			Obsolete
PAL16L8			Obsolete
PAL16P8			Obsolete
PAL16P8A			Obsolete
PAL16R4			Obsolete
PAL16R4A			Obsolete
PAL16R6			Obsolete
PAL16R6A			Obsolete
PAL16R8			Obsolete

Existing XC2000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PAL16R8A			Obsolete
PAL16X4			Obsolete
PAL16RA8			Obsolete
PAL16RP4			Obsolete
PAL16RP6			Obsolete
PAL16RP8			Obsolete
PAL18L4			Obsolete
PAL20C1			Obsolete
PAL20L2			Obsolete
PAL20L8			Obsolete
PAL20L10			Obsolete
PAL20R4			Obsolete
PAL20R6			Obsolete
PAL20R8			Obsolete
PAL20R10			Obsolete
PAL20RS4			Obsolete
PAL20RS8			Obsolete
PAL20S10			Obsolete
PAL20X4			Obsolete
PAL20X8			Obsolete
PAL20X10			Obsolete
PAL22RX8			Obsolete
PAL22V10			Obsolete
PAL32R16			Obsolete
PAL32V10			Obsolete
PAL64R32			Obsolete
RD4	FD4RE		
RD8	FD8CE		
RD8CR	FD8RE		
RS4		SR4CE	

Existing XC2000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
RS8		SR8CE	
RS8CR	SR8RE		
RS8PR		SR8RLE	
RS8R		SR8RE	
ZMX2000			Obsolete
ZXPAL			Obsolete
ZX2000			Obsolete
74-42	X74_42		
74-138	X74_138		
74-139	X74_139		
74-151	X74_151		
74-152	X74_152		
74-160	X74_160		
74-161	X74_161		
74-164	X74_164		
74-194	X74_194		
74-195	X74_195		
74-352	X74_352		

XC3000 Replacement and Obsolete Macro Functions

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
ASHEET	ASHEETP		
BPAD	IOPAD		
BRM			Obsolete
BRM2			Obsolete
BSHEET	BSHEETL		
CBINRIP			Obsolete
CDECRIP		CD4CE	
CSHEET	CSHEETL		
C2BCP			Obsolete
C2BCPRD		CB2CLE	
C2BCR		CB2RE	
C2BCRD		CB2CE	
C2BP		CB2CLE	
C2BR		CB2RE	
C2BRD		CB2CLE	
C3BIT8			Obsolete
C3BIT8O7			Obsolete
C3SQUARE			Obsolete
C4BCP		CB4CLE	
C4BCPRD	CB2CLE		
C4BCR	CB2RE		
C4BCRD	CB2CE		
C4JX		CJ4CE	
C4JXC		CJ4CE	
C4JXCR		CJ4RE	
C4JXCRD		CJ4CE	
C4JXRD		CJ4CE	
C5BIT32			Obsolete

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
C5SQUARE			Obsolete
C6JCR		CJ4RE	
C8BCP		CB4CLE	
C8BCPRD		CB4CLE	
C8BCR		CB4RE	
C8BCRD		CB4CLE	
C8JCR	CJ4RE		
C8UDLD			Obsolete
C10BCPRD	CD4CLE		
C10BCRD	CD4CE		
C10BPRD		CD4CLE	
C10JCR	CJ5RE		
C12JCR		CJ8RE	
C16BARD		CB4CE	
C16BCP		CB4CLE	
C16BCPR		CB4CLE	
C16BCPRD	CB4CLE		
C16BCRD	CB4CE		
C16BPRD		CB4CLE	
C16BUDRD	CB4CLED		
C16DNLD		CB4CLED	
C16JCR	CJ8RE		
C16UDLD		CB4CLED	
C16UPLD		CB4CLED	
C256BCP		CB8CLE	
C256BCPR	CB8CLE		
C256BCR	CB8RE		
C256BCRD	CB8CE		
C256FCRD		CB8CLED	
DFF	FD		

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
DSHEET	DSHEETL		
D2-4		D2_4E	
D2-4E	D2_4E		
D3-8		D3_8E	
D3-8E	D3_8E		
ESHEET	ESHEETL		
FDC		FDCE	
FDCR	FDRE		
FDCRD	FDCE		
FDCS	FDSE		
FDM			Obsolete
FDMR			Obsolete
FDMRD			Obsolete
FDMS			Obsolete
FDRD	FDC		
FJK		FJKC	
FJKRD	FJKC		
FJKS		FJKSRE	
FRS			Obsolete
FSR			Obsolete
FT		FTC	
FT0		FTC	
FT0R		FTRSE	
FTP		FTCP	
FTPRD		FTRSLE	
FTRD	FTC		
FTS		FTSRE	
GADD	ADD1		
GCOMP	COMP2		
GLTGT	COMPM2		

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
GMUX	M2_1		
GOSC			Obsolete
HX42	X74_42		
HX48			Obsolete
HX77			Obsolete
HX125			Obsolete
HX138	X74_138		
HX139	X74_139		
HX147	X74_147		
HX148	X74_148		
HX151	X74_151		
HX152	X74_152		
HX153	X74_153		
HX154	X74_154		
HX157	X74_157		
HX158	X74_158		
HX160	X74_160		
HX161	X74_161		
HX162	X74_162		
HX163	X74_163		
HX164	X74_164		
HX166			Obsolete
HX168	X74_168		
HX169			Obsolete
HX174	X74_174		
HX179			Obsolete
HX194	X74_194		
HX195	X74_195		
HX198			Obsolete
HX199			Obsolete

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
HX240			Obsolete
HX241			Obsolete
HX244			Obsolete
HX257			Obsolete
HX258			Obsolete
HX259			Obsolete
HX273	X74_273		
HX278			Obsolete
HX280	X74_280		
HX283	X74_283		
HX298	X74_298		
HX352	X74_352		
HX373			Obsolete
HX374			Obsolete
HX377	X74_377		
HX390	X74_390		
HX393			Obsolete
HX518	X74_518		
HX521	X74_521		
HX541			Obsolete
HX577			Obsolete
HX590			Obsolete
HX595			Obsolete
INFF	IFD		
INLAT	ILD		
M3-1		M4_1	
M3-1E		M4_1E	
M4-1	M4_1		
M4-1C			Obsolete
M4-1E	M4_1E		

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
M4-2		M2_1	
M8-1	M8_1		
M8-1E	M8_1E		
OBUFZ	OBUFT		
OUTFF	OFD		
OUTFFT	OFDT		
OUTFFZ	OFDT		
PAD		IOPAD	
PAL2RA10			Obsolete
PAL6L16A			Obsolete
PAL8L14A			Obsolete
PAL10H8			Obsolete
PAL10H20			Obsolete
PAL10L8			Obsolete
PAL12H6			Obsolete
PAL12L6			Obsolete
PAL12L10			Obsolete
PAL14H4			Obsolete
PAL14L4			Obsolete
PAL14L8			Obsolete
PAL16A4			Obsolete
PAL16C1			Obsolete
PAL16H2			Obsolete
PAL16L2			Obsolete
PAL16L6			Obsolete
PAL16L8			Obsolete
PAL16P8			Obsolete
PAL16P8A			Obsolete
PAL16RA8			Obsolete
PAL16RP4			Obsolete

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PAL16RP6			Obsolete
PAL16RP8			Obsolete
PAL16R4			Obsolete
PAL16R4A			Obsolete
PAL16R6			Obsolete
PAL16R6A			Obsolete
PAL16R8			Obsolete
PAL16R8A			Obsolete
PAL16X4			Obsolete
PAL18L4			Obsolete
PAL20C1			Obsolete
PAL20L2			Obsolete
PAL20L8			Obsolete
PAL20L10			Obsolete
PAL20RS4			Obsolete
PAL20RS8			Obsolete
PAL20R4			Obsolete
PAL20R6			Obsolete
PAL20R8			Obsolete
PAL20R10			Obsolete
PAL20S10			Obsolete
PAL20X4			Obsolete
PAL20X8			Obsolete
PAL20X10			Obsolete
PAL22RX8			Obsolete
PAL22V10			Obsolete
PAL32R16			Obsolete
PAL32V10			Obsolete
PAL64R32			Obsolete
PHFRCOMP			Obsolete

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
RD4	FD4RE		
RD4RD		FD4RE	
RD8	FD8CE		
RD8CR	FD8RE		
RD8RD		FD8RE	
RS4		SR4CE	
RS4C		SR4CE	
RS4CR	SR4RE		
RS4CRD	SR4CE		
RS4RD		SR4RE	
RS8		SR8CE	
RS8C		SR8CE	
RS8CR	SR8RE		
RS8CRD	SR8CE		
RS8PR		SR8RLE	
RS8R		SR8RE	
RS8RD		SR8RE	
SAR			Obsolete
TBUF	BUFT		
WM8-1			Obsolete
WM16-1			Obsolete
X74160D			Obsolete
X74160U		X74_160	
X74161D			Obsolete
X74161U		X74_161	
X74165A		X74_165S	
X74165S	X74_165S		
X7474		FDCP	
ZMX3000			Obsolete
ZX3000			Obsolete

Existing XC3000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
ZXPAL			Obsolete
ZXTTL			Obsolete
74-42	X74_42		
74-138	X74_138		
74-139	X74_139		
74-151	X74_151		
74-152	X74_152		
74-160	X74_160		
74-161	X74_161		
74-162	X74_162		
74-163	X74_163		
74-164	X74_164		
74-194	X74_194		
74-195	X74_195		
74-352	X74_352		

XC4000 Replacement and Obsolete Macro Functions

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
ACC8H		ACC8	
ACC16H		ACC16	
ADD1		ADD4	
ADD2		ADD4	
ADD12		ADD16	
ADD24			Obsolete
ADD32			Obsolete
ADDSUB1		ADSU4	
ADSU8H		ADSU8	
ADSU16H		ADSU16	
ASHEET	ASHEETP		
BIDI4			Obsolete
BIDI8			Obsolete
BIDI16			Obsolete
BPAD	IOPAD		
BSHEET	BSHEETL		
CDECRIP		CD4CE	
COMP8H	COMP8		
COMP16H	COMP16		
COMP32			Obsolete
COMPM8H		COMPMC8	
COMPM16H		COMPMC16	
COMPM32			Obsolete
CSHEET	CSHEETL		
CUP8H	CB8CLE		
CUP16H	CB16CLE		
C2BCPRD		CB2CLE	
C2BCR		CB2RE	

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
C2BCRD		CB2CE	
C2BINRIP		CB2CE	
C4BCPRD	CB2CLE		
C4BCR	CB2RE		
C4BCRD	CB2CE		
C4BINRIP		CB4CE	
C4JXCR		CJ4RE	
C4JXCRD		CJ4CE	
C8BCPRD		CB4CLE	
C8BCR		CB4RE	
C8BCRD		CB4CLE	
C8JCR	CJ4RE		
C8JCRD	CJ4CE		
C10BCPRD	CD4CLE		
C10BCRD	CD4CE		
C10JCR	CJ5RE		
C10JCRD	CJ5CE		
C16BCPRD	CB4CLE		
C16BCR	CB4RE		
C16BCRD	CB4CE		
C16BUDRD	CB4CLED		
C16JCR	CJ8RE		
C16JCRD	CJ8CE		
C32BUDRD		CB8CLED	
C64BUDRD		CB8CLED	
C256BCPR	CB8CLE		
C256BCR	CB8RE		
C256BCRD	CB8CE		
DEC2-4EH		X74_139	
DEC3-8EH		X74_138	

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
DECODE24			Obsolete
DSHEET	DSHEETL		
D2-4		D2_4E	
D2-4E	D2_4E		
D3-8		D3_8E	
D3-8E	D3_8E		
D4-16			Obsolete
D4-16E	D4_16E		
D7SEGH			Obsolete
D7SEGMH			Obsolete
ENCPR8H		X74_148	
ESHEET	ESHEETL		
FDCR	FDRE		
FDCS	FDSE		
FDMRD			Obsolete
FDMSD			Obsolete
FDRD	FDCE		
FDRDKN	FDCE_1		
FDSD	FDPE		
FDSDKN	FDPE_1		
FJKRD	FJKCE		
FJKSD	FJKPE		
FRD		FDRSE	
FSD		FDRSE	
FTPRD		FTRSLE	
IN4	IBUF4		
IN8	IBUF8		
IN16	IBUF16		
INFF	IFD		
INFF4	IFD4		

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
INFF8	IFD8		
INFF16	IFD16		
INFFS	IFDI		
INLAT	ILD		
INLAT4	ILD4		
INLAT8	ILD8		
INLAT16	ILD16		
INLATS	ILDI		
INREG			Obsolete
INREGS			Obsolete
LD			Obsolete
LDE			Obsolete
LDM			Obsolete
LDRD			Obsolete
LDSD			Obsolete
LRS			Obsolete
LSR			Obsolete
MAJ4			Obsolete
MUX4-1H	M4_1		
MUX8-1H	M8_1		
MUX16-1H	M16_1		
M2-1	M2_1		
M2-1E	M2_1E		
M4-1	M4_1		
M4-1E	M4_1E		
M8-1	M8_1		
M8-1E	M8_1E		
M16-1	M16_1		
M16-1E	M16_1E		
OUT4	OBUF4		

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
OUT8	OBUF8		
OUT16	OBUF16		
OUTFF	OFD		
OUTFF4	OFD4		
OUTFF8	OFD8		
OUTFF16	OFD16		
OUTFFS		OFDT	
OUTFFT	OFDT		
OUTFFTS		OFDT	
PAD		IOPAD	
PADU	UPAD		
PAL2RA10			Obsolete
PAL6L16A			Obsolete
PAL8L14A			Obsolete
PAL10H8			Obsolete
PAL10H20			Obsolete
PAL10L8			Obsolete
PAL12H6			Obsolete
PAL12L6			Obsolete
PAL12L10			Obsolete
PAL14H4			Obsolete
PAL14L4			Obsolete
PAL14L8			Obsolete
PAL16A4			Obsolete
PAL16C1			Obsolete
PAL16H2			Obsolete
PAL16L2			Obsolete
PAL16L6			Obsolete
PAL16L8			Obsolete
PAL16P8			Obsolete

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PAL16P8A			Obsolete
PAL16R4			Obsolete
PAL16R4A			Obsolete
PAL16R6			Obsolete
PAL16R6A			Obsolete
PAL16R8			Obsolete
PAL16R8A			Obsolete
PAL16RA8			Obsolete
PAL16RP4			Obsolete
PAL16RP6			Obsolete
PAL16RP8			Obsolete
PAL16X4			Obsolete
PAL18L4			Obsolete
PAL20C1			Obsolete
PAL20L2			Obsolete
PAL20L8			Obsolete
PAL20L10			Obsolete
PAL20R4			Obsolete
PAL20R6			Obsolete
PAL20R8			Obsolete
PAL20R10			Obsolete
PAL20RS4			Obsolete
PAL20RS8			Obsolete
PAL20S10			Obsolete
PAL20X4			Obsolete
PAL20X8			Obsolete
PAL20X10			Obsolete
PAL22RX8			Obsolete
PAL22V10			Obsolete
PAL32R16			Obsolete

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PAL32V10			Obsolete
PAL64R32			Obsolete
PARE9H		X74_280	
PARO9H		X74_280	
PHFRCOMP			Obsolete
PRSC8-9			Obsolete
RAM64X4		RAM32X4	
RAM64X8		RAM32X8	
RAM128X4		RAM32X4	
RAM128X8		RAM32X8	
RD4	FD4CE		
RD4R	FD4RE		
RD8	FD8CE		
RD8H	FD8RE		
RD8R	FD8RE		
RD16	FD16CE		
RD16H	RAM16X1		
RD16R	FD16RE		
RF16X4			Obsolete
RF16X8			Obsolete
RF16X16			Obsolete
RF32X4			Obsolete
RF32X8			Obsolete
RF32X16			Obsolete
RM16X2H	RAM16X2		
RM16X4H	RAM16X4		
RM16X8H	RAM16X8		
RM32X4H	RAM32X4		
RM32X8H	RAM32X8		
RM64X4H		RAM32X4	

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
RM64X8H		RAM32X8	
RM128X4H		RAM16X4	
RM128X8H		RAM16X8	
RS4	SR4CE		
RS4P	SR4CLE		
RS4R	SR4RE		
RS8	SR8CE		
RS8P	SR8CLE		
RS8PH		SR8RLE	
RS8R	SR8RE		
RS16	SR16CE		
RS16P	SR16CLE		
RS16PH		SR16RLE	
RS16R	SR16RE		
TBUF	BUFT		
WM8-1			Obsolete
WM16-1			Obsolete
X74-42	X74_42		
X74-48			Obsolete
X74-83		X74_283	
X74-85		X74_L85	
X74-138	X74_138		
X74-139	X74_139		
X74-147	X74_147		
X74-148	X74_148		
X74-150	X74_150		
X74-151	X74_151		
X74-152	X74_152		
X74-153	X74_153		
X74-154	X74_154		

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
X74-157	X74_157		
X74-158	X74_158		
X74-160	X74_160		
X74-161	X74_161		
X74-162	X74_162		
X74-163	X74_163		
X74-164	X74_164		
X74-165S	X74_165S		
X74-166		SR8CLE	
X74-168	X74_168		
X74-174	X74_174		
X74-194	X74_194		
X74-195	X74_195		
X74-198		X74_195	
X74-199		X74_195	
X74-240		BUFT8	
X74-241		BUFT8	
X74-244			Obsolete
X74-245			Obsolete
X74-257		M2_1	
X74-258		M2_1	
X74-259			Obsolete
X74-273	X74_273	FD8CE	
X74-278			Obsolete
X74-280	X74_280		
X74-283	X74_283		
X74-298	X74_298		
X74-352	X74_352		
X74-373			Obsolete
X74-374			Obsolete

Existing XC4000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
X74-377	X74_377		
X74-390	X74_390		
X74-518	X74_518		
X74-521	X74_521		
X74-540			Obsolete
X74-541			Obsolete
X74-577			Obsolete
X74-595		SR8CE	
X74160D	X74_160		
X74160U	X74_160		
X74161D	X74_161		
X74161U	X74_161		
X74_162	X74_163		
ZHM4000			Obsolete
ZMX4000			Obsolete
ZX4000			Obsolete
ZXPAL			Obsolete

XC7000 Replacement and Obsolete Macro Functions

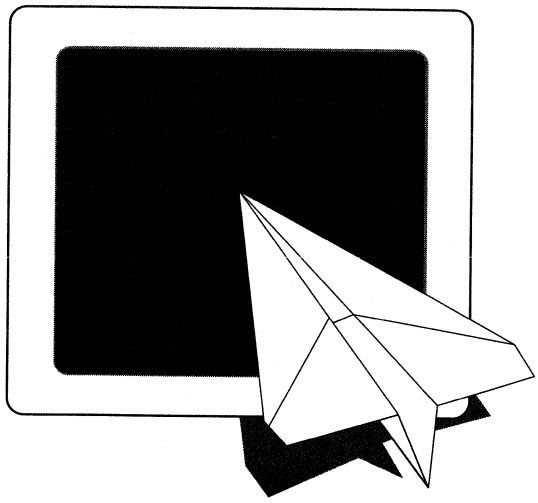
Existing XC7000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PA7236A			Obsolete
PA7272A			Obsolete
PA7272B			Obsolete
PA7272C			Obsolete
PA73108A			Obsolete
PA73108B			Obsolete
PL00	NAND2		
PL02	NOR2		
PL04	INV		
PL08	AND2		
PL10	NAND3		
PL11	AND3		
PL20	NAND4		
PL21	AND4		
PL27	NOR3		
PL30	NAND8		
PL32	OR2		
PL74		FDCP	
PL74PZ	FD		
PL76P		FJKCP	
PL83		ADD4	
PL85	X74_L85		
PL86	XOR2		
PL126	BUFE		
PL138	X74_138		
PL139	X74_139		
PL148P		X74_148	
PL150	X74_150		
PL151	X74_151		

Existing XC7000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PL153	X74_153		
PL157	X74_157		
PL161	X74_161		
PL163	X74_163		
PL164	X74_164		
PL166			Obsolete
PL191P		CB4X2	
PL194		SR4CLED	
PL198P		SR8RLED	
PL240			Obsolete
PL244	BUFT4		
PL266	XNOR2		
PL298P		X74_298	
PL373P	LD8		
PL374		FD8	
PL374PZ	FD8		
PL377	X74_377		
PL518P		X74_518	
PL869P		CB8X2	
PLADD4		ADD4	
PLADD8		ADD8	
PLALU8		ACC8X1	
PLALU8H		ACC8X2	
PLAND2	AND2		
PLAND3	AND3		
PLAND4	AND4		
PLAND8	AND8		
PLBI			Obsolete
PLBI8			Obsolete
PLBUF	BUF		

Existing XC7000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PLBUFT	BUFE		
PLBUFT4	BUFT4		
PLCE	BUFCE		
PLCEIO			Obsolete
PLCOMP8		COMP8	
PLCOMP8R			Obsolete
PLCTR4A	X74_161		
PLCTR4S	X74_163		
PLCTR8		CB8RLE	
PLCTR8T			Obsolete
PLDECOD2	X74_139		
PLDECOD3	X74_138		
PLDFF	FD		
PLDFF8	FD8		
PLDFFE8	X74_377		
PLDFFEI	IFDX1		
PLDFFEI8	IFD8X1		
PLDFFEIO			Obsolete
PLDFFI	IFD		
PLDFFI8	IFD8		
PLDFFIO			Obsolete
PLDFFRSC		FDCP	
PLDFFT8		FD8	
PLDLAT	LD		
PLDLAT8	LD8		
PLDLATI	ILD		
PLDLATI8	ILD8		
PLDLATIO			Obsolete
PLENCOD8		X74_148	
PLFCLKIO			Obsolete

Existing XC7000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PLFCOMP			Obsolete
PLFOE	BUFFOE		
PLFOEIO			Obsolete
PLFPLA48		PL48PIN	
PLFSTCLK	BUFG		
PLIN	IBUF		
PLIN8	IBUF8		
PLIN8A	IBUF8		
PLIO			Obsolete
PLIO8			Obsolete
PLJKFFC		FJKCP	
PLMAG4	X74_L85		
PLMAG8		COMPM8	
PLMAG4R			Obsolete
PLMAG8R			Obsolete
PLMUX2	M2_1		
PLMUX4		M4_1E	
PLMUX8	X74_151		
PLMUX16	X74_150		
PLMUX2R4			X74_298
PLMUX2X4	X74_157		
PLMUX4X2	X74_153		
PLNAND2	NAND2		
PLNAND3	NAND3		
PLNAND4	NAND4		
PLNAND8	NAND8		
PLNOR2	NOR2		
PLNOR3	NOR3		
PLNOR4	NOR4		
PLNOR8	NOR8		

Existing XC7000 Name	Exact Unified Replacement	Closest Unified Replacement	Obsolete
PLNOT	INV		
PLNOTT4			Obsolete
PLOR2	OR2		
PLOR3	OR3		
PLOR4	OR4		
PLOR8	OR8		
PLOUT	OBUF		
PLOUT8	OBUF8		
PLOUT8A	OBUF8		
PLOUTT	OBUFEX1		
PLOUTT8	OBUFE8X1		
PLPLD9	PLFB9		
PLPLD9F	PLFFB9		
PLSHIF4		SR4RLED	
PLSHIF4A		SR4CLED	
PLSHIF8		SR8RLED	
PLSHIF8I	X74_164		
PLSHIF8O			Obsolete
PLUPDN4		CB4X2	
PLUPDN8		CB8X2	
PLUPDN8T			Obsolete
PLXNOR2	XNOR2		
PLXOR2	XOR2		
PLXOR3	XOR3		
PLXOR4	XOR4		
PLXOR5	XOR5		
PLXOR6	XOR6		
PLXOR7	XOR7		
PLXOR8	XOR8		
PLXOR9	XOR9		



Design Elements

XACT Libraries Guide

XACT Libraries Guide — April, 1994 (0401098 01)

Printed in U.S.A.

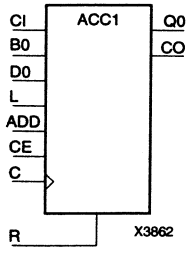
Chapter 3

Design Elements

This chapter contains design elements for the XC2000, XC3000, XC4000, and XC7000 architectures. The elements are organized in alphanumeric order, with all numeric suffixes in ascending order.

ACC1

1-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7272 or XC7336 designs

ACC1 can add or subtract a 1-bit unsigned-binary word to or from the contents of a 1-bit data register and store the results in the register. The register can be loaded with a 1-bit word. The synchronous reset (R) has priority over all other inputs and, when High, causes the output to go to logic level zero. Clock (C) transitions are ignored when clock enable (CE) is Low.

The accumulator is asynchronously reset, output Low, when power is applied or when global reset, GR, is active (Low).

Load

When the load input (L) is High, CE is ignored and the data on the input D0 is loaded into the 1-bit register.

Add

When control inputs ADD and CE are both High, the accumulator adds a 1-bit word (B0) and carry-in (CI) to the contents of the 1-bit register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. The carry-out (CO) is not registered synchronously with the data output. CO always reflects the accumulation of input B0 and the contents of the register, which allows cascading of ACC1s by connecting CO of one stage to CI of the next stage. In add mode, CO acts as a carry-out, and CO and CI are active-High.

Subtract

When ADD is Low and CE is High, the 1-bit word B0 and CI are subtracted from the contents of the register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. The carry-out (CO) is not registered synchronously with the data output. CO always reflects the accumulation of input B0 and the contents of the register, which allows cascading of ACC1s by connecting CO of one stage to CI of the next stage. In subtract mode, CO acts as a borrow, and CO and CI are active-Low.

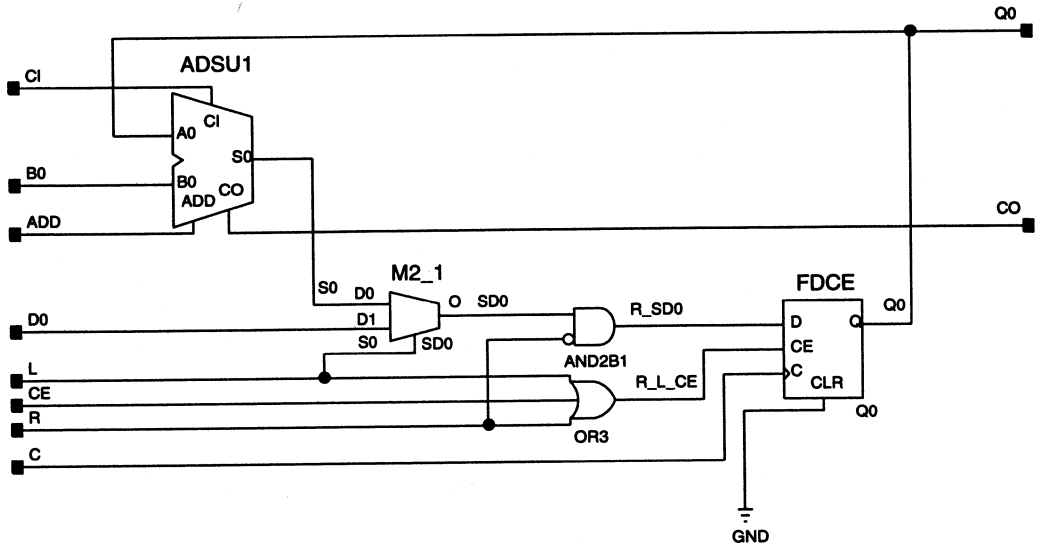
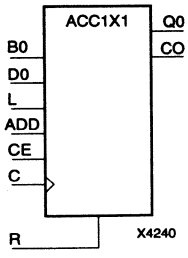


Figure 3-1 ACC1 XC2000 Implementation

For the XC7000 EPLD architecture, the CO output is not valid during load (L=High), during reset (R=High), or while CE is inactive (Low). Also, the CI and CO pins are not implemented using the EPLD arithmetic carry path and should be used to cascade accumulators. Refer to "ACC1X1" and "ACC1X2" for descriptions of cascadable EPLD accumulators.

ACC1X1

1-Bit Loadable Cascadable Accumulator with Carry-Out and Synchronous Reset for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

*not supported for XC7272 or XC7336 designs

ACC1X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC1X1 is a low-order adder component that can be used as a stand-alone or cascaded with high-order accumulators through its CO output. ACC1X1 adds or subtracts a 1-bit binary word (B0) to or from the contents of a 1-bit data register and stores the results in the register. The register can be loaded with a 1-bit word. When the load input (L) is High, CE is ignored and the data on input D0 is loaded into the 1-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds a 1-bit word (B0) to the contents of the 1-bit register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out and is active-High.

Subtract

When ADD is Low and CE is High, the 1-bit word B0 is subtracted from the contents of the register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the accumulator and tie its A

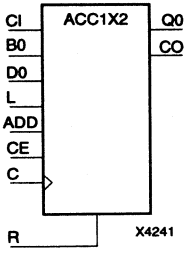
and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ACC1X2 for the least-significant accumulator and connect an ADD1X1 to its CI input. Then connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the accumulator. The accumulator register is initialized to zero when powered is applied or when the device Master Reset input is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs							Outputs	
R	**L**	**CE**	**B0**	**D0**	**C**	**ADD**	**Q0**	**CO**
1	X	X	X	X	↑	X	0	0
0	1	X	X	D0	↑	X	d	0
0	0	0	X	X	X	X	No Chg	0
0	0	1	B0	X	↑	1	q+b	CO
0	0	1	B0	X	↑	0	q-b	CO

d, q, b = state of referenced input one set-up time prior to active clock transition

ACC1X2

1-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7272 or XC7336 designs

ACC1X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC1X2 is a high-order adder component cascaded to lower-order accumulators through its CI input. ACC1X2 adds or subtracts a 1-bit binary word (B0) to or from the contents of a 1-bit data register and stores the results in the register. The register can be loaded with a 1-bit word. When the load input (L) is High, CE is ignored and the data on input D0 is loaded into the 1-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds a 1-bit word (B0) and carry-in (CI) to the contents of the 1-bit register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out, and CO and CI are active-High.

Subtract

When ADD is Low and CE is High, the 1-bit word B0 and CI are subtracted from the contents of the register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore, must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and can only be connected to the CI input of another

EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the accumulator and tie its A and B inputs to GND; the S output becomes the carry-out.

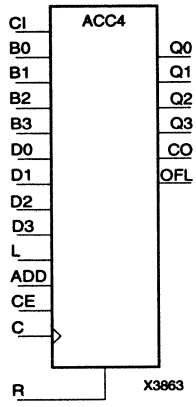
The accumulator register is initialized to zero when power is applied or when the device Master Reset input is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs								Outputs	
R	**L**	**CE**	**B0**	**D0**	**CI**	**C**	**ADD**	**Q0**	**CO**
1	X	X	X	X	X	↑	X	0	0
0	1	X	X	D0	X	↑	X	d	0
0	0	0	X	X	X	X	X	No Chg	0
0	0	1	B0	X	CI	↑	1	q+b+ci	CO
0	0	1	B0	X	CI	↑	0	q-b-ci	CO

d, q, b, ci = state of referenced input one set-up time prior to active clock transition

ACC4

4-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Primitive*

* not supported for XC7272 or XC7336 designs

ACC4 can add or subtract a 4-bit unsigned-binary or twos-complement word to or from the contents of a 4-bit data register and store the results in the register. The register can be loaded with a 4-bit word. In the XC4000 family, the accumulator is implemented using carry logic and relative location constraints, which assure most efficient logic placement. The synchronous reset (R) has priority over all other inputs, and when High, causes all outputs to go to logic level zero. Clock (C) transitions are ignored when clock enable (CE) is Low.

The accumulator is asynchronously reset, output Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active Low; the GSR active level is programmable.

Load

When the load input (L) is High, CE is ignored and the data on inputs D3 – D0 is loaded into the 4-bit register.

Unsigned Binary Versus Twos-Complement

ACC4 can operate on either 4-bit unsigned binary numbers or 4-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

For the XC7000 EPLD architecture, the CO output is not valid during load (L=High), during reset (R=High), or while CE is inactive (Low).

Also, the CI and CO pins are not implemented using the EPLD arithmetic carry path and should be used to cascade accumulators. Refer to "ACC1X1" and "ACC1X2" for descriptions of cascadable EPLD accumulators. The OFL output is not provided on the ACC4 symbol in XC7000.

Unsigned Binary Operation

For unsigned binary operation, the ACC4 can represent numbers between 0 and 15, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtracter. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds. The carry-out (CO) is not registered synchronously with the data outputs. CO always reflects the accumulation of inputs B3 – B0 and the contents of the register, which allows cascading of ACC4s by connecting CO of one stage to CI of the next stage. An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

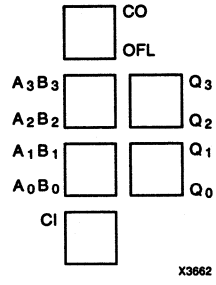
OFL should be ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ACC4 can represent numbers between -8 and +7, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High. The overflow (OFL) is not registered synchronously with the data outputs. OFL always reflects the accumulation of inputs B3 – B0 and the contents of the register, which allows cascading of ACC4s by connecting OFL of one stage to CI of the next stage.

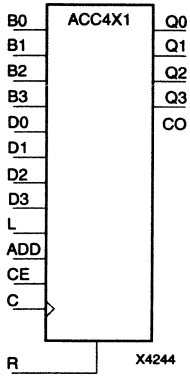
CO should be ignored in twos-complement operation.

XC4000 Topology



ACC4X1

4-Bit Loadable Cascadable Accumulator with Carry-Out and Synchronous Reset for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7272 or XC7336 designs

ACC4X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC4X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order accumulators through its CO output. ACC4X1 adds or subtracts a 4-bit binary word (B3 – B0) to or from the contents of a 4-bit data register and stores the results in the register. The register can be loaded with a 4-bit word. When the load input (L) is High, CE is ignored and the data on inputs D3 – D0 is loaded into the 4-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds a 4-bit word (B3 – B0) to the contents of the 4-bit register. The result is stored in the register and appears on outputs Q3 – Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out and is active-High.

Subtract

When ADD is Low and CE is High, the 4-bit word B3 – B0 is subtracted from the contents of the register. The result is stored in the register and appears on outputs Q3 – Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic,

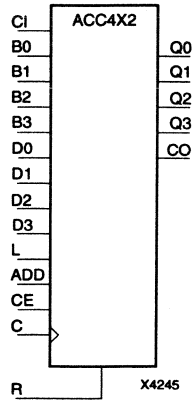
connect an ADD1X2 to the CO output of the accumulator and tie its A and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ACC4X2 for the least-significant accumulator and connect an ADD1X1 to its CI input. Then connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the accumulator.

The accumulator register is initialized to zero when power is applied or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Refer to "ACC1X1" for truth table derivation.

ACC4X2

4-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7272 or XC7336 designs

ACC4X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC4X2 is a high-order adder component cascaded to lower-order accumulators through its CI input. ACC4X2 adds or subtracts a 4-bit binary word (B3 – B0) to or from the contents of a 4-bit data register and stores the results in the register. The register can be loaded with a 4-bit word. When the load input (L) is High, CE is ignored and the data on inputs D3 – D0 is loaded into the 4-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds a 4-bit word (B3 – B0) and carry-in (CI) to the contents of the 4-bit register. The result is stored in the register and appears on outputs Q3 – Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out, and CO and CI are active-High.

Subtract

When ADD is Low and CE is High, the 4-bit word B3 – B0 and CI are subtracted from the contents of the register. The result is stored in the register and appears on outputs Q3 – Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain and can only be connected to the CI input of another EPLD-specific

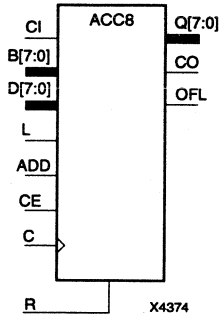
arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the accumulator and tie its A and B inputs to GND; the S output becomes the carry-out.

The accumulator register is initialized to zero when power is applied or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Refer to "ACC1X2" for truth table derivation.

ACC8

8-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7272 or XC7336 designs

ACC8 can add or subtract an 8-bit unsigned-binary or twos- complement word to or from the contents of an 8-bit data register and store the results in the register. The register can be loaded with an 8-bit word. In the XC4000 family, the accumulator is implemented using carry logic and relative location constraints, which assure most efficient logic placement. The synchronous reset (R) has priority over all other inputs, and when High, causes all outputs to go to logic level zero. Clock (C) transitions are ignored when clock enable (CE) is Low.

The accumulator is asynchronously reset, output Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active Low; the GSR active level is programmable.

Load

When the load input (L) is High, CE is ignored and the data on inputs D7 – D0 is loaded into the 8-bit register.

Unsigned Binary Versus Twos-Complement

ACC8 can operate on either 8-bit unsigned binary numbers or 8-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

For the XC7000 EPLD architecture, the CO output is not valid during load (L=High), during reset (R=High), or while CE is inactive (Low).

Also, the CI and CO pins are not implemented using the EPLD arithmetic carry path and should be used to cascade accumulators. Refer to "ACC8X1" and "ACC8X2" for descriptions of cascadable EPLD accumulators. The OFL output is not provided on the ACC8 symbol in XC7000.

Unsigned Binary Operation

For unsigned binary operation, ACC8 can represent numbers between 0 and 255, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtracter. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds. The carry-out (CO) is not registered synchronously with the data outputs. CO always reflects the accumulation of inputs B7 – B0 and the contents of the register, which allows cascading of ACC8s by connecting CO of one stage to CI of the next stage. An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

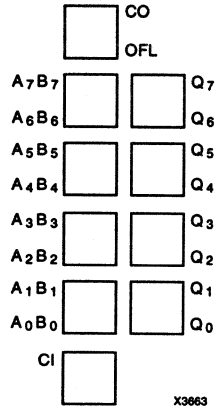
OFL should be ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ACC8 can represent numbers between -128 and +127, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High. The overflow (OFL) is not registered synchronously with the data outputs. OFL always reflects the accumulation of inputs B7 – B0 and the contents of the register, which allows cascading of ACC8s by connecting OFL of one stage to CI of the next stage.

CO should be ignored in twos-complement operation.

XC4000 Topology



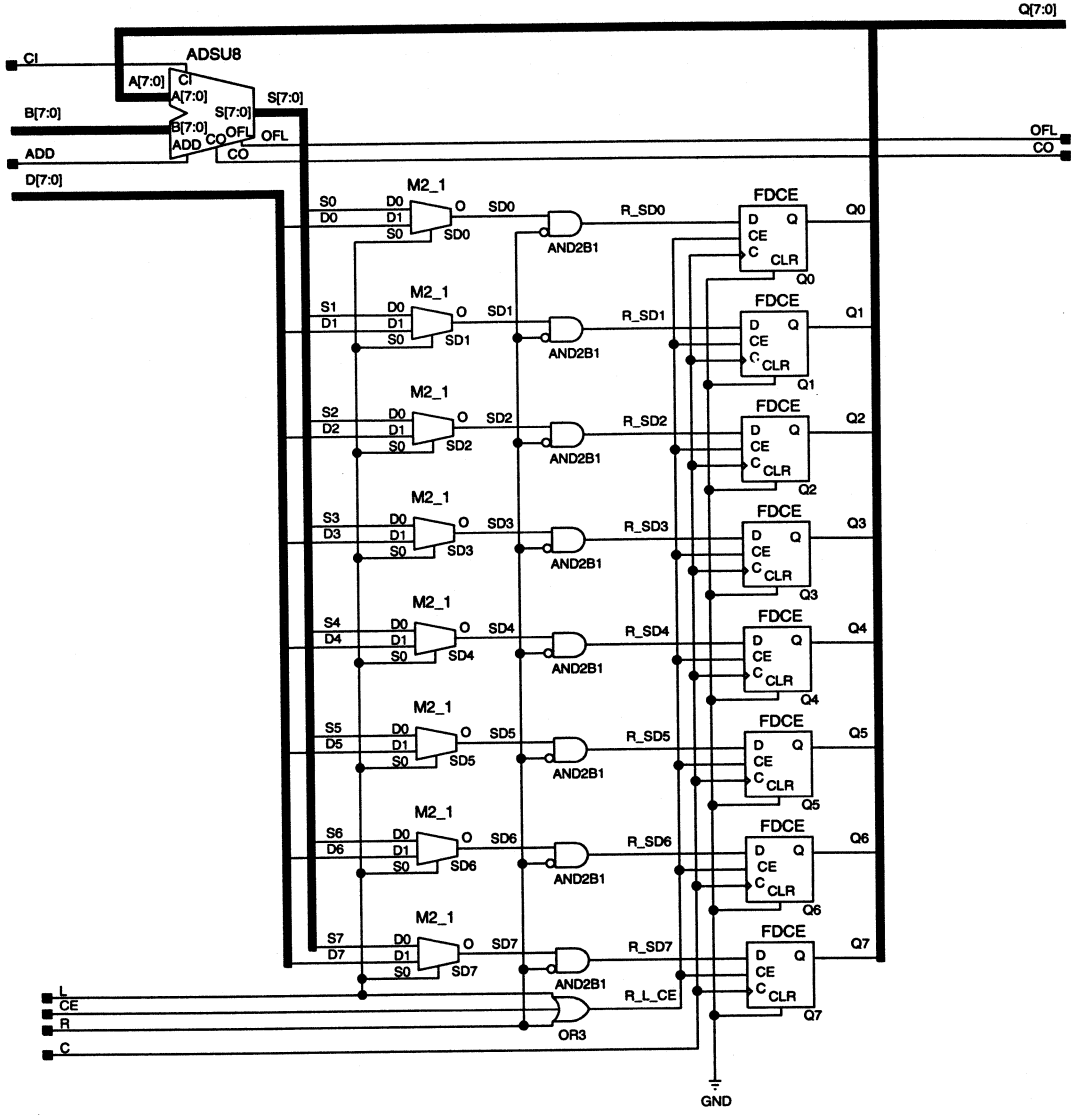


Figure 3-2 ACC8 XC3000 Implementation

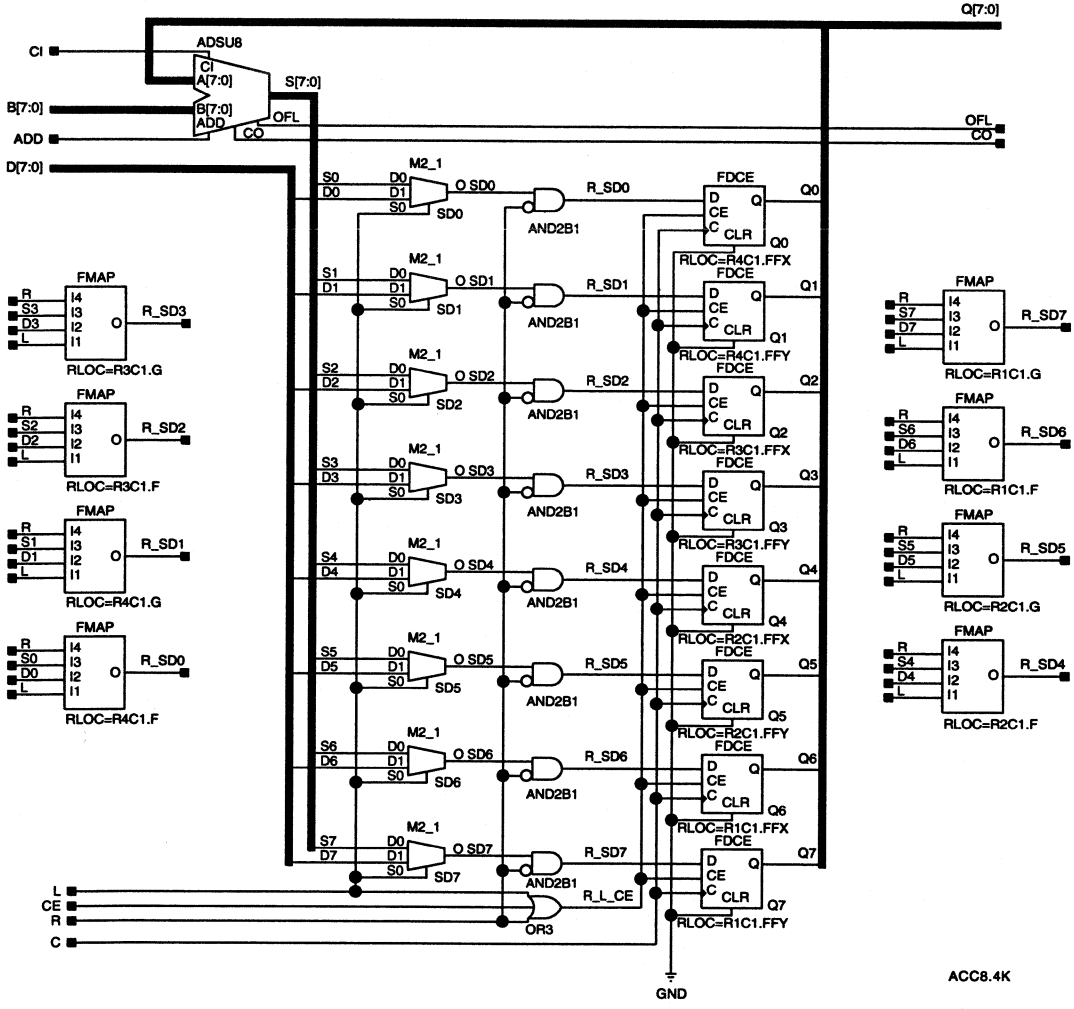


Figure 3-3 ACC8 XC4000 Implementation

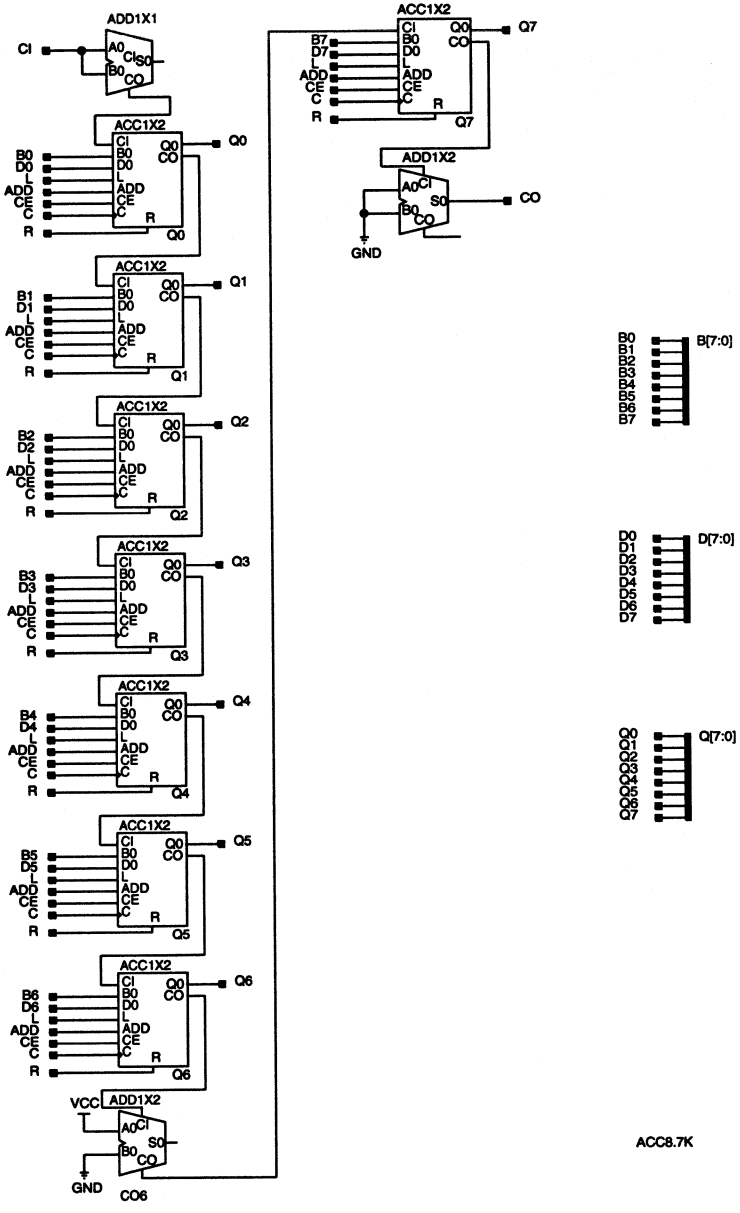
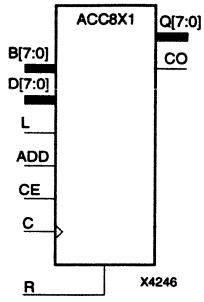


Figure 3-4 ACC8 XC7000 Implementation

ACC8X1

8-Bit Loadable Cascadable Accumulator with Carry-Out and Synchronous Reset for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7272 or XC7336 designs

ACC8X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC8X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order accumulators through its CO output. ACC8X1 adds or subtracts an 8-bit binary word (B7 – B0) to or from the contents of an 8-bit data register and stores the results in the register. The register can be loaded with an 8-bit word. When the load input (L) is High, CE is ignored and the data on inputs D7 – D0 is loaded into the 8-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds an 8-bit word (B7 – B0) to the contents of the 8-bit register. The result is stored in the register and appears on outputs Q7 – Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out and is active-High.

Subtract

When ADD is Low and CE is High, the 8-bit word B7 – B0 is subtracted from the contents of the register. The result is stored in the register and appears on outputs Q7 – Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic,

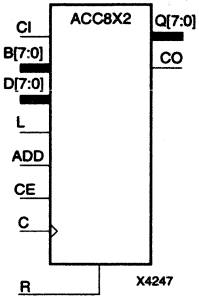
connect an ADD1X2 to the CO output of the accumulator and tie its A and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ACC8X2 for the least-significant accumulator and connect an ADD1X1 to its CI input. Then connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the accumulator.

The accumulator register is initialized to zero when power is applied or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Refer to "ACC1X1" for truth table derivation.

ACC8X2

8-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7272 or XC7336 designs

ACC8X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC8X2 is a high-order adder component cascaded to lower-order accumulators though its CI input. ACC8X2 adds or subtracts an 8-bit binary word (B7 – B0) to or from the contents of an 8-bit data register and stores the results in the register. The register can be loaded with an 8-bit word. When the load input (L) is High, CE is ignored and the data on inputs D7 – D0 is loaded into the 8-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds an 8-bit word (B7 – B0) and carry-in (CI) to the contents of the 8-bit register. The result is stored in the register and appears on outputs Q7 – Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out, and CO and CI are active-High.

Subtract

When ADD is Low and CE is High, the 8-bit word B7 – B0 and CI are subtracted from the contents of the register. The result is stored in the register and appears on outputs Q7 – Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another

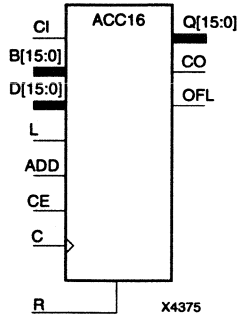
EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the accumulator and tie its A and B inputs to GND; the S output becomes the carry-out.

The accumulator register is initialized to zero when power is applied or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Refer to "ACC1X2" for truth table derivation.

ACC16

16-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7272 or XC7336 designs

ACC16 can add or subtract a 16-bit unsigned-binary or twos-complement word to or from the contents of a 16-bit data register and store the results in the register. The register can be loaded with a 16-bit word. In the XC4000 family, the accumulator is implemented using carry logic and relative location constraints, which assure most efficient logic placement. The synchronous reset (R) has priority over all other inputs, and when High, causes all outputs to go to logic level zero. Clock (C) transitions are ignored when clock enable (CE) is Low.

The accumulator is asynchronously reset, output Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active Low; the GSR active level is programmable.

Load

When the load input (L) is High, CE is ignored and the data on inputs D15 – D0 is loaded into the 16-bit register.

Unsigned Binary Versus Twos-Complement

ACC16 can operate on either 16-bit unsigned binary numbers or 16-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

For the XC7000 EPLD architecture, the CO output is not valid during load (L=High), during reset (R=High), or while CE is inactive (Low).

Also, the CI and CO pins are not implemented using the EPLD arithmetic carry path and should be used to cascade accumulators. Refer to "ACC8X1" and "ACC8X2" for descriptions of cascadable EPLD accumulators. The OFL output is not provided on the ACC16 symbol in XC7000.

Unsigned Binary Operation

For unsigned binary operation, ACC16 can represent numbers between 0 and 65535, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtracter. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds. The carry-out (CO) is not registered synchronously with the data outputs. CO always reflects the accumulation of inputs B15 – B0 and the contents of the register, which allows cascading of ACC16s by connecting CO of one stage to CI of the next stage. An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

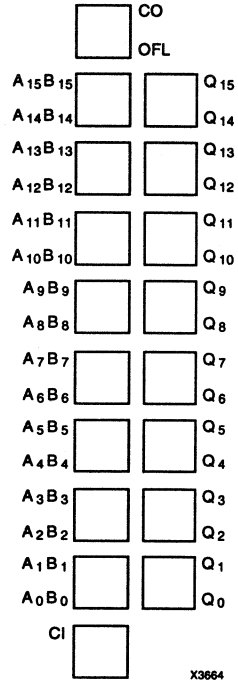
OFL should be ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ACC16 can represent numbers between -32768 and +32767, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High. The overflow (OFL) is not registered synchronously with the data outputs. OFL always reflects the accumulation of inputs B15 – B0 and the contents of the register, which allows cascading of ACC16s by connecting OFL of one stage to CI of the next stage.

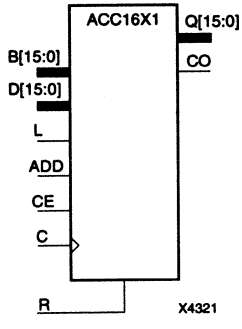
CO should be ignored in twos-complement operation.

XC4000 Topology



ACC16X1

16-Bit Loadable Cascadable Accumulator with Carry-Out and Synchronous Reset for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Macro*

* not supported for XC7272 or XC7336 designs

ACC16X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC16X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order accumulators though its CO output. ACC16X1 adds or subtracts a 16-bit binary word (B15 – B0) to or from the contents of a 16-bit data register and stores the results in the register. The register can be loaded with a 16-bit word. When the load input (L) is High, CE is ignored and the data on inputs D15 – D0 is loaded into the 16-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds a 16-bit word (B15 – B0) to the contents of the 16-bit register. The result is stored in the register and appears on outputs Q15 – Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out and is active-High.

Subtract

When ADD is Low and CE is High, the 16-bit word B15 – B0 is subtracted from the contents of the register. The result is stored in the register and appears on outputs Q15 – Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic,

connect an ADD1X2 to the CO output of the accumulator and tie its A and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ACC16X2 for the least-significant accumulator and connect an ADD1X1 to its CI input. Then connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the accumulator.

The accumulator register is initialized to zero when power is applied or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Refer to "ACC1X1" for truth table derivation.

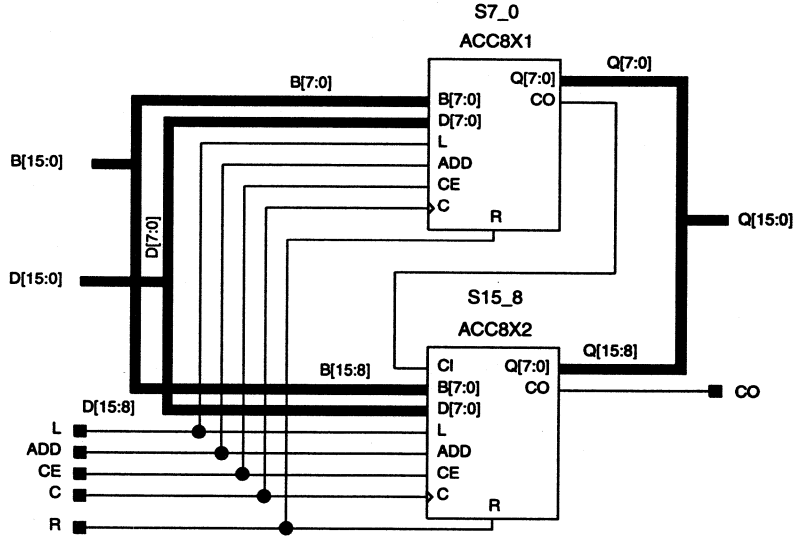
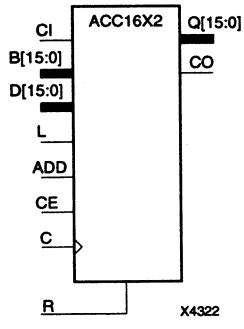


Figure 3-5 ACC16X1 XC7000 Implementation

ACC16X2

16-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Macro*

* not supported for XC7272 or XC7336 designs

ACC16X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ACC16X2 is a high-order adder component cascaded to lower-order accumulators through its CI input. ACC16X2 adds or subtracts a 16-bit binary word (B15 – B0) to or from the contents of a 16-bit data register and stores the results in the register. The register can be loaded with a 16-bit word. When the load input (L) is High, CE is ignored and the data on inputs D15 – D0 is loaded into the 16-bit register. The synchronous reset (R) has priority over all other inputs and, when High, causes all outputs to go to logic level zero. When reset (R) and load (L) are inactive, clock (C) transitions are ignored when clock enable (CE) is Low.

Add

When control inputs ADD and CE are both High, the accumulator adds a 16-bit word (B15 – B0) and carry-in (CI) to the contents of the 16-bit register. The result is stored in the register and appears on outputs Q15 – Q0 during the Low-to-High clock transition. In add mode, CO acts as a carry-out, and CO and CI are active-High.

Subtract

When ADD is Low and CE is High, the 16-bit word B15 – B0 and CI are subtracted from the contents of the register. The result is stored in the register and appears on outputs Q15 – Q0 during the Low-to-High clock transition. In subtract mode, CO acts as a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another

EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the accumulator and tie its A and B inputs to GND; the S output becomes the carry-out.

The accumulator register is initialized to zero when power is applied or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Refer to "ACC1X2" for truth table derivation.

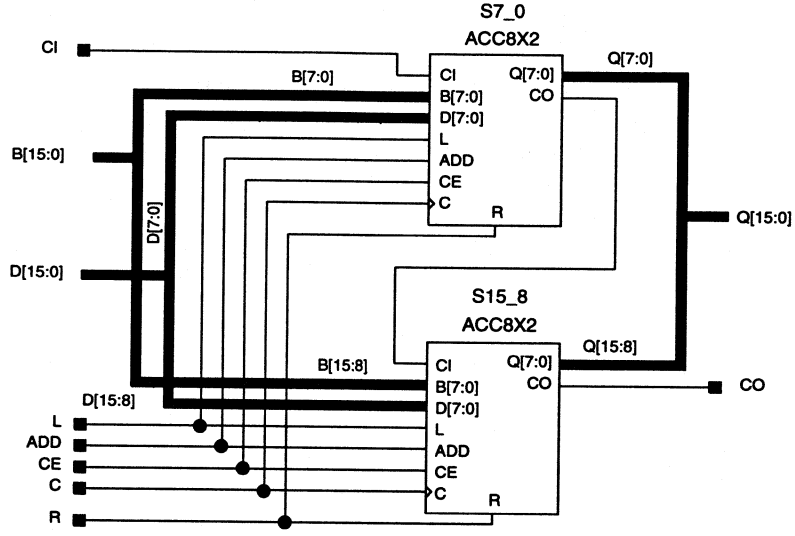
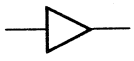


Figure 3-6 ACC16X2 XC7000 Implementation

ACLK

Alternate Clock Buffer

ACLK



X3883

XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	N/A	N/A

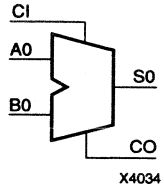
ACLK, the alternate clock buffer, is used to distribute high fan-out clock signals throughout a PLD device. One ACLK buffer on each device provides direct access to every Configurable Logic Block (CLB) and Input Output Block (IOB) clock pin. The ACLK buffer is slightly slower than the global clock buffer (GCLK) but otherwise similar. Unlike GCLK, the routing resources used for the ACLK network can be used to route other signals if it is not used. For this reason, if only one of the GCLK and ACLK buffers is used, GCLK is preferred. The ACLK input (I) can come from one of the following sources.

- A CMOS-level signal on the dedicated BCLKIN pin (XC3000 only). BCLKIN is a direct CMOS-only input to the ACLK buffer. To use the BCLKIN pin, connect the input of the ACLK element directly to the PAD element (without using an IBUF in between).
- A CMOS- or TTL-level external signal. To connect an external input to the ACLK buffer, connect the input of the ACLK element to the output of the IBUF for that signal. Unless the corresponding PAD element is constrained otherwise, APR or PPR typically places that IOB directly adjacent to the ACLK buffer.
- The on-chip crystal oscillator. The output of the XTAL oscillator on XC2000 and XC3000 devices is directly adjacent to the ACLK buffer input. If the GXTL element is used, the output of the XTAL oscillator is automatically connected to the ACLK buffer; do not use the ACLK element for anything else.
- An internal signal. To drive the ACLK buffer with an internal signal, connect that signal directly to the input of the ACLK element.

For a negative-edge clock, insert an INV (inverter) element between the ACLK output and the clock input. Inversion is performed inside the CLB, or in the case of IOB clock pins, on the IOB clock line (that controls the clock sense for the IOBs on an entire edge of the chip).

ADD1

1-Bit Full Adder with Carry-In and Carry-Out



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

ADD1, a cascadable 1-bit full adder with carry-in and carry-out, adds two 1-bit words (A and B) and a carry-in (CI), producing a binary sum (S0) output and a carry-out (CO). For XC7000 cascadable adders, refer to "ADD1X1" and "ADD1X2."

Inputs			Outputs	
A0	B0	CI	S0	CO
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

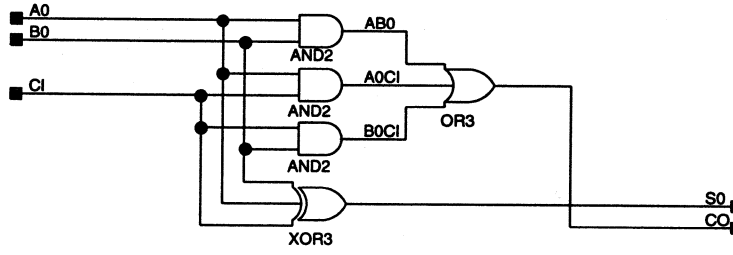
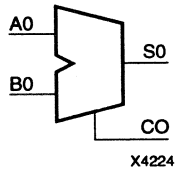


Figure 3-7 ADD1 XC2000 Implementation

ADD1X1

1-Bit Cascadable Full Adder with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

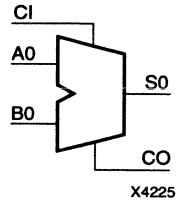
ADD1X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD1X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order adders through its CO output. ADD1X2 adds two words (A0 and B0) and produces a sum output (ADD1X2) and carry-out (CO).

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out. If a carry-in is required from general-purpose logic, extend the length of the adder by one additional bit and connect the carry-in signal to both the least-significant A and B inputs (the least-significant S output is not used) to generate a carry into the carry chain for the second bit of the adder.

Refer to "ADD1" for truth table derivation.

ADD1X2

1-Bit Cascadable Full Adder with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

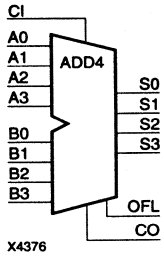
ADD1X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD1X2 is a high-order adder component cascaded to lower-order adders through its CI input. ADD1X2 adds two words (A0 and B0) and a carry-in (CI), producing a sum output (S0) and carry-out (CO).

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out.

Refer to "ADD1" for truth table derivation.

ADD4

4-Bit Cascadable Full Adder with Carry-In, Carry-Out, and Overflow



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7336 designs

ADD4 is implemented in the XC4000 family using carry logic and relative location constraints, which assure most efficient logic placement. ADD4 adds two words (A3 – A0 and B3 – B0) and a carry-in (CI), producing a sum output (S3 – S0) and carry-out (CO) or overflow (OFL). For XC7000 cascadable adders, refer to "ADD4X1" and "ADD4X2." The ADD4 CI and CO pins do not use the EPLD carry chain.

Unsigned Binary Versus Twos Complement

ADD4 can operate on either 4-bit unsigned binary numbers or 4-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

Unsigned Binary Operation

For unsigned binary operation, ADD4 can represent numbers between 0 and 15, inclusive. CO is active (High) when the sum exceeds the bounds of the adder.

An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

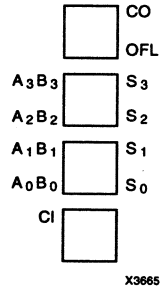
OFL is ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ADD4 can represent numbers between -8 and +7, inclusive. OFL is active (High) when the sum exceeds the bounds of the adder.

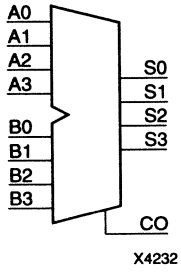
CO is ignored in twos-complement operation.

XC4000 Topology



ADD4X1

4-Bit Cascadable Full Adder with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

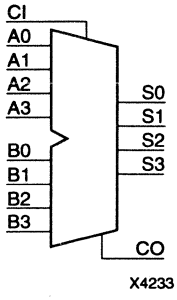
ADD4X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD4X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order adders through its CO output. ADD4X2 adds two words (A3 – A0 and B3 – B0), producing a sum output (S3 – S0) and carry-out (CO).

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out. If a carry-in is required from general-purpose logic, extend the length of the adder by one additional bit and connect the carry-in signal to both the least-significant A and B inputs (the least-significant S output is not used) to generate a carry into the carry chain for the second bit of the adder.

Refer to "ADD1" for truth table derivation.

ADD4X2

4-Bit Cascadable Full Adder with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

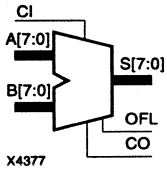
ADD4X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD4X2 is a high-order adder component cascaded to lower-order adders through its CI input. ADD4X2 adds two words (A3 – A0 and B3 – B0) and a carry-in (CI), producing a sum output (S3 – S0) and carry-out (CO).

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out.

Refer to "ADD1" for truth table derivation.

ADD8

8-Bit Cascadable Full Adder with Carry-In, Carry-Out, and Overflow



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7336 designs

ADD8 is implemented in the XC4000 family using carry logic and relative location constraints, which assure most efficient logic placement. ADD8 adds two words (A7 – A0 and B7 – B0) and a carry-in (CI), producing a sum output (S7 – S0) and carry-out (CO) or overflow (OFL). For XC7000 cascadable adders, refer to "ADD8X1" and "ADD8X2." The ADD8 CI and CO pins do not use the EPLD carry chain.

Unsigned Binary Versus Twos-Complement

ADD8 can operate on either 8-bit unsigned binary numbers or 8-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

Unsigned Binary Operation

For unsigned binary operation, ADD8 can represent numbers between 0 and 255, inclusive. CO is active (High) when the sum exceeds the bounds of the adder.

An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

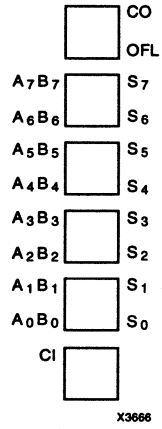
OFL is ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ADD8 can represent numbers between -128 and +127, inclusive. OFL is active (High) when the sum exceeds the bounds of the adder.

CO is ignored in twos-complement operation.

XC4000 Topology



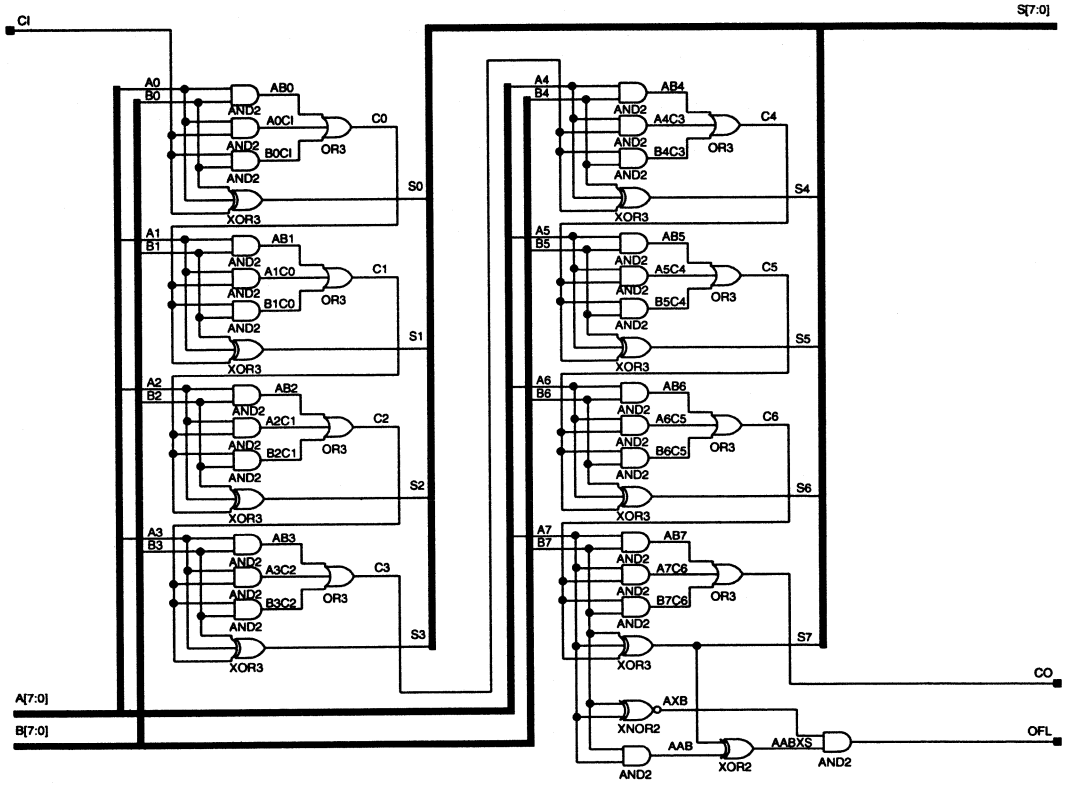


Figure 3-8 ADD8 XC3000 Implementation

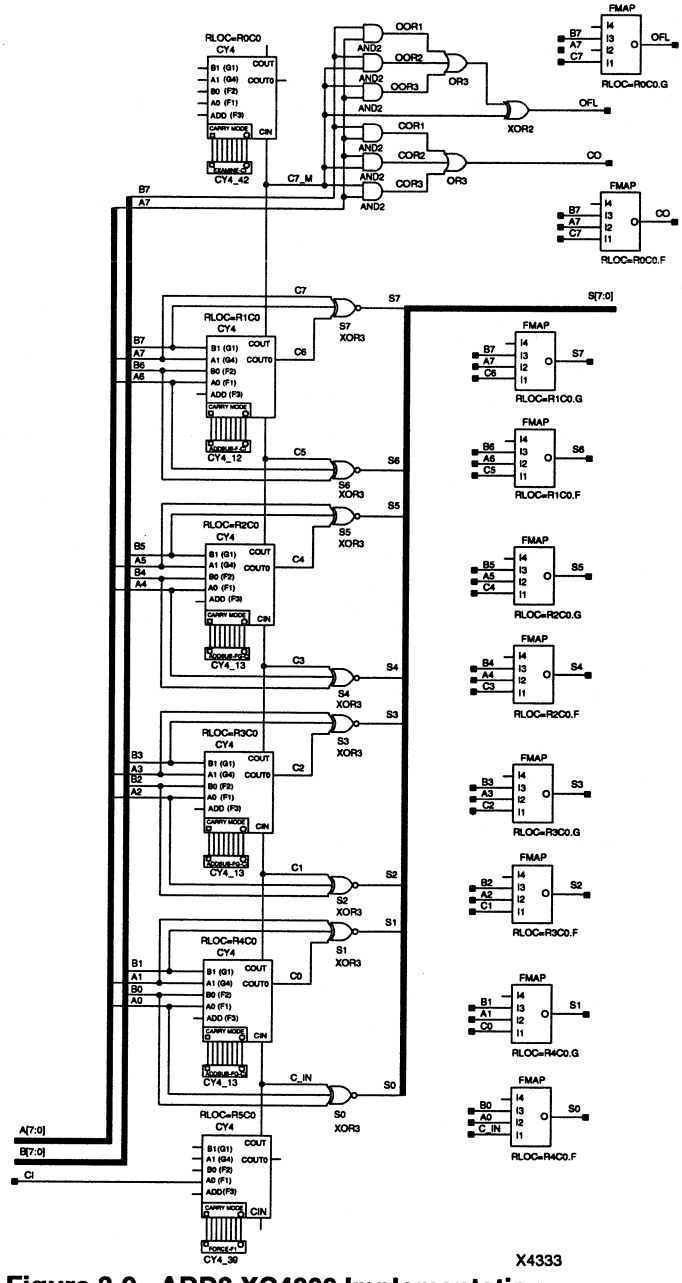
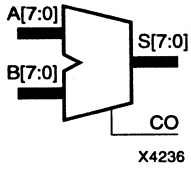


Figure 3-9 ADD8 XC4000 Implementation

ADD8X1

8-Bit Loadable Cascadable Full Adder with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

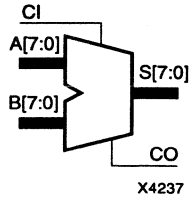
* not supported for XC7336 designs

ADD8X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD8X1 is a low-order adder component that can be used stand-alone or cascaded with high-order adders through its CO output. ADD8X2 adds two words (A7 – A0 and B7 – B0), producing a sum output (S7 – S0) and carry-out (CO). The CO output passes into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out. If a carry-in is required from general-purpose logic, extend the length of the adder by one additional bit and connect the carry-in signal to both the least-significant A and B inputs (the least-significant S output is not used). This procedure generates a carry into the carry chain for the second bit of the adder.

Refer to "ADD1" for truth table derivation.

ADD8X2

8-Bit Cascadable Full Adder with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

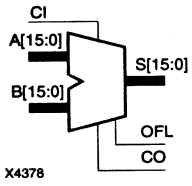
ADD8X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD8X2 is a high-order adder component cascaded to lower-order adders through its CI input. ADD8X2 adds two words (A7 – A0 and B7 – B0) and a carry-in (CI), producing a sum output (S7 – S0) and carry-out (CO).

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out.

Refer to "ADD1" for truth table derivation.

ADD16

16-Bit Cascadable Full Adder with Carry-In, Carry-Out, and Overflow



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7336 designs

ADD16 is implemented in the XC4000 family using carry logic and relative location constraints, which assure most efficient logic placement. ADD16 adds two words (A15 – A0 and B15 – B0) and a carry-in (CI), producing a sum output (S15 – S0) and carry-out (CO) or overflow (OFL). For XC7000 cascadable adders, refer to "ADD16X1" and "ADD16X2." The ADD16 CI and CO pins do not use the EPLD carry chain.

Unsigned Binary Versus Twos-Complement

ADD16 can operate on either 16-bit unsigned binary numbers or 16-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

Unsigned Binary Operation

For unsigned binary operation, ADD16 can represent numbers between 0 and 65535, inclusive. CO is active (High) when the sum exceeds the bounds of the adder.

An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

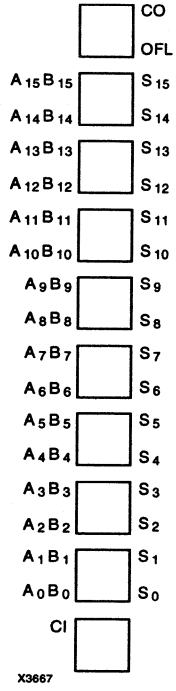
OFL is ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ADD16 can represent numbers between -32768 and +32767, inclusive. OFL is active (High) when the sum exceeds the bounds of the adder.

CO is ignored in twos-complement operation.

XC4000 Topology



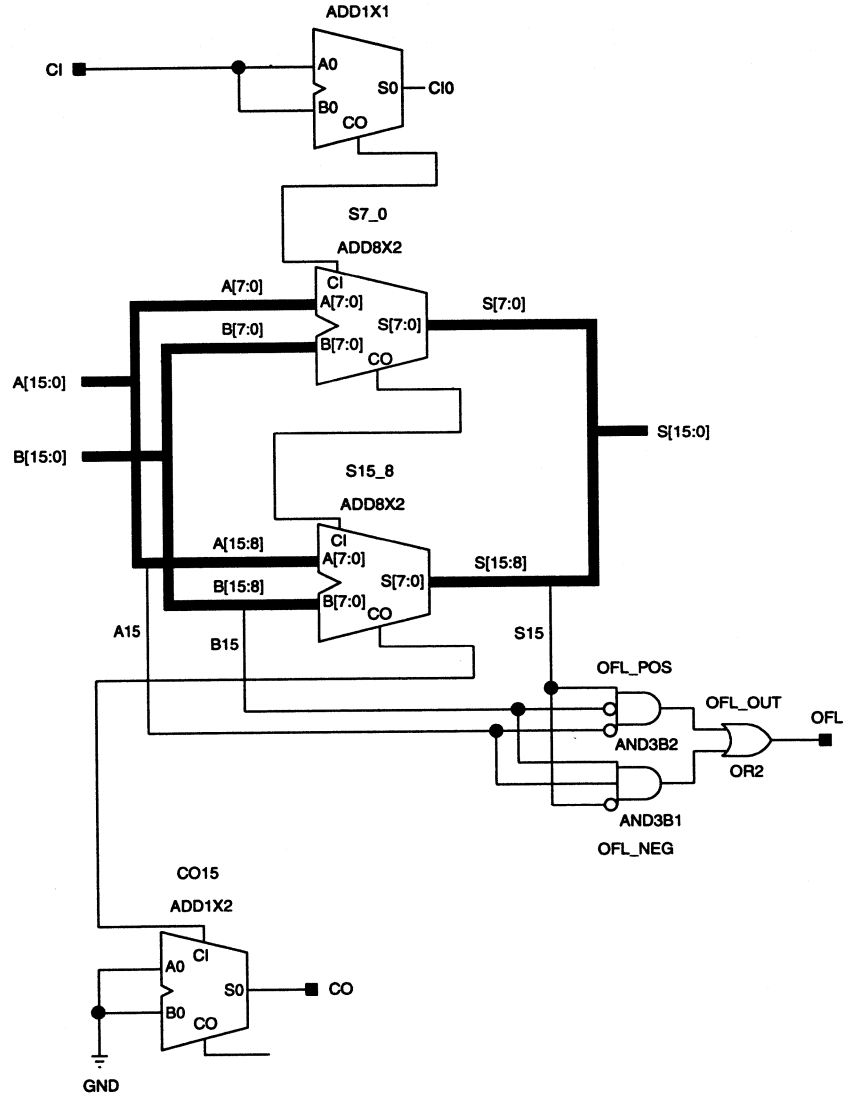
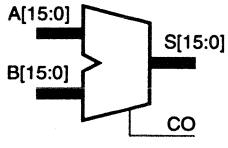


Figure 3-10 ADD16 XC7000 Implementation

ADD16X1

16-Bit Cascadable Full Adder with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Macro*

* not supported for XC7336 designs

ADD16X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD16X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order adders through its CO output. ADD16X2 adds two words (A15 – A0 and B15 – B0), producing a sum output (S15 – S0) and carry-out (CO).

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out. If a carry-in is required from general-purpose logic, extend the length of the adder by one additional bit and connect the carry-in signal to both the least-significant A and B inputs (the least-significant S output is not used) to generate a carry into the carry chain for the second bit of the adder.

Refer to "ADD1" for truth table derivation.

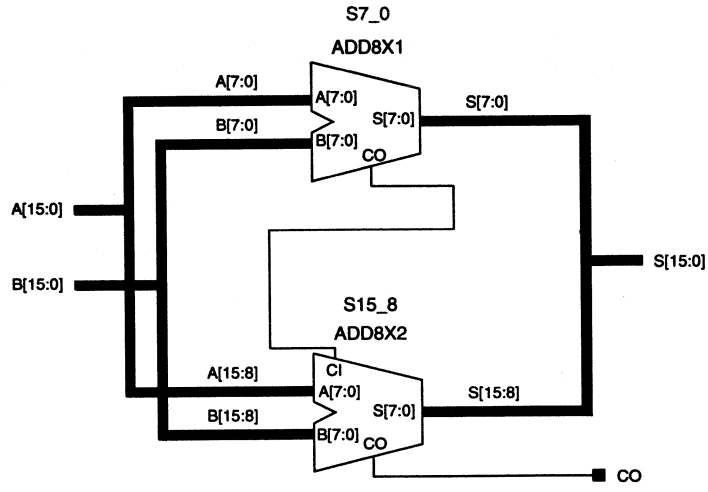
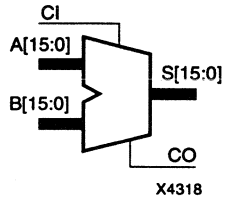


Figure 3-11 ADD16X1 XC7000 Implementation

ADD16X2

16-Bit Cascadable Full Adder with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Macro*

* not supported for XC7336 designs

ADD16X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADD16X2 is a high-order adder component cascaded to lower-order adders through its CI input. ADD16X2 adds two words (A15 – A0 and B15 – B0) and a carry-in (CI), producing a sum output (S15 – S0) and carry-out (CO).

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, use an adder (or cascaded adders) with one extra bit and tie the most-significant A and B inputs to GND; the most-significant S output becomes the carry-out. Refer to "ADD1" for truth table derivation.

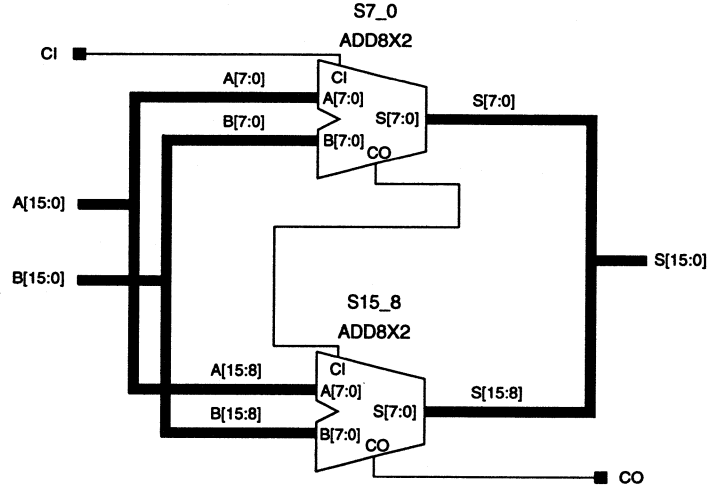
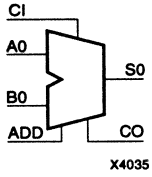


Figure 3-12 ADD16X2 XC7000 Implementation

ADSU1

1-Bit Cascadable Adder/Subtracter with Carry-In and Carry-Out



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

When the ADD input is High, two 1-bit words (A0 and B0) are added with a carry-in (CI), producing a 1-bit output (S0) and a carry-out (CO). When the ADD input is Low, B0 is subtracted from A0, producing a result (S0) and borrow (CO). In add mode, CO represents a carry-out, and CO and CI are active-High. In subtract mode, CO represents a borrow, and CO and CI are active-Low. Refer to "ADSU1X1" and "ADSU1X2" for cascadable EPLD symbols.

Inputs			Outputs	
A0	B0	CI	S0	CO
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Add Function, ADD=1

Inputs			Outputs	
A0	**B0**	**CI**	**S0**	**CO**
0	0	0	1	0
0	1	0	0	0
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	1
1	1	1	0	1

Subtract Function, ADD=0

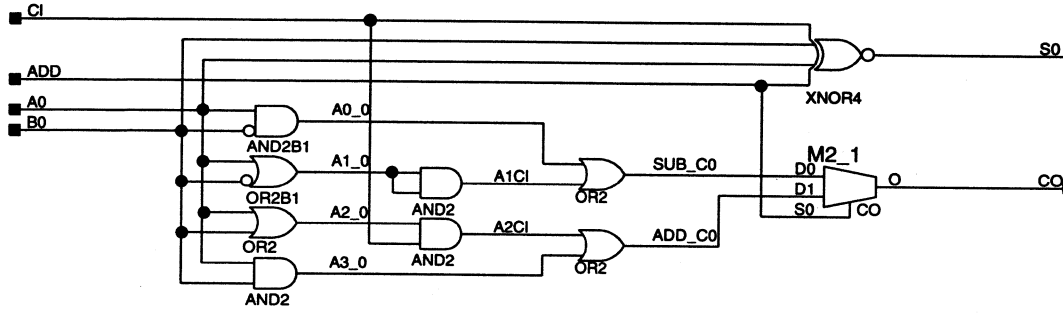
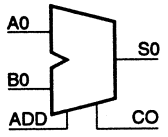


Figure 3-13 ADSU1 XC2000 Implementation

ADSU1X1

1-Bit Cascadable Adder/Subtracter with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

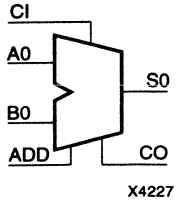
ADSU1X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU1X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order adders through its CO output. When the ADD input is High, two 1-bit words (A0 and B0) are added, producing and a 1-bit output (S0) and carry-out (CO). When the ADD input is Low, B0 is subtracted from A0, producing a result (S0) and borrow (CO). In add mode, CO represents a carry-out and is active-High. In subtract mode, CO represents a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ADSU1X2 for the least-significant adder/subtracter and connect an ADD1X1 to its CI input. Connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the adder/subtracter.

Refer to "ADSU1" for truth table derivation.

ADSU1X2

1-Bit Cascadable Adder/Subtracter with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

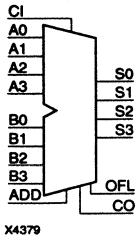
ADSU1X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU1X2 is a high-order adder component cascaded to lower-order adders through its CI input. When the ADD input is High, two 1-bit words (A0 and B0) are added with a carry-in (CI), producing a 1-bit output (S0) and carry-out (CO). When the ADD input is Low, B0 is subtracted from A0, producing a result (S0) and borrow (CO). In add mode, CO represents a carry-out, and CO and CI are active-High. In subtract mode, CO represents a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out.

Refer to "ADSU1" for truth table derivation.

ADSU4

4-Bit Cascadable Adder/Subtracter with Carry-In, Carry-Out, and Overflow



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7336 designs

ADSU4 is implemented in the XC4000 family using carry logic and relative location constraints, which assure most efficient logic placement. When the ADD input is High, two 4-bit words (A3 – A0 and B3 – B0) are added with a carry-in (CI), producing a 4-bit sum (S3 – S0) and carry-out (CO) or overflow (OFL). When the ADD input is Low, B3 – B0 is subtracted from A3 – A0, producing a 4-bit difference (S3 – S0) and CO or OFL. In add mode, CO and CI are active-High. In subtract mode, CO and CI are active-Low. For cascadable EPLD symbols, refer to "ADSU4X1" and "ADSU4X2." ADSU4 CI and CO pins do not use the EPLD carry chain.

Unsigned Binary Versus Twos-Complement

ADSU4 can operate on either 4-bit unsigned binary numbers or 4-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

With adder/subtracters, either unsigned binary or twos-complement operations cause an overflow. If the result crosses the overflow boundary, an overflow is generated. Similarly, when the result crosses the carry-out boundary, a carry-out is generated. The following figure shows the ADSU carry-out and overflow boundaries.

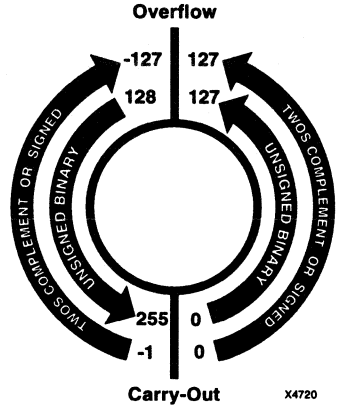


Figure 3-14 ADSU Carry-Out and Overflow Boundaries

Unsigned Binary Operation

For unsigned binary operation, ADSU4 can represent numbers between 0 and 15, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtracter. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds.

An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

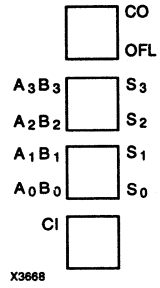
OFL is ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ADSU4 can represent numbers between -8 and +7, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High.

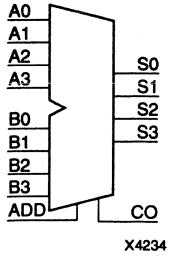
CO is ignored in twos-complement operation.

XC4000 Topology



ADSU4X1

4-Bit Cascadable Adder/Subtracter with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

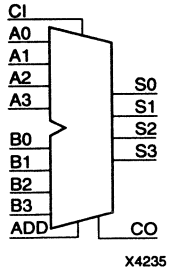
ADSU4X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU4X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order adders through its CO output. When the ADD input is High, two 4-bit words (A3 – A0 and B3 – B0) are added, producing a 4-bit output (S3 – S0) and carry-out (CO). When the ADD input is Low, B3 – B0 is subtracted from A3 – A0, producing a result (S3 – S0) and borrow (CO). In add mode, CO represents a carry-out and is active-High. In subtract mode, CO represents a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ADSU4X2 for the least-significant adder/subtracter and connect an ADD1X1 to its CI input. Connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the adder/subtracter.

Refer to "ADSU1" for truth table derivation.

ADSU4X2

4-Bit Cascadable Adder/Subtracter with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

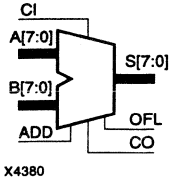
ADSU4X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU4X2 is a high-order adder component cascaded to lower-order adders through its CI input. When the ADD input is High, two 4-bit words (A3 – A0 and B3 – B0) are added with a carry-in (CI), producing a 4-bit output (S3 – S0) and carry-out (CO). When the ADD input is Low, B3 – B0 is subtracted from A3 – A0, producing a result (S3 – S0) and borrow (CO). In add mode, CO represents a carry-out, and CO and CI are active-High. In subtract mode, CO represents a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out.

Refer to "ADSU1" for truth table derivation.

ADSU8

8-Bit Cascadable Adder/Subtracter with Carry-In, Carry-Out, and Overflow



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7336 designs

ADSU8 is implemented in the XC4000 family using carry logic and relative location constraints, which assure most efficient logic placement. When the ADD input is High, two 8-bit words (A7 – A0 and B7 – B0) are added with a carry-in (CI), producing an 8-bit sum (S7 – S0) and carry-out (CO) or overflow (OFL). When the ADD input is Low, B7 – B0 is subtracted from A7 – A0, producing an 8-bit difference (S7 – S0) and CO or OFL. In add mode, CO and CI are active-High. In subtract mode, CO and CI are active-Low. OFL is active-High in add and subtract modes. For cascadable EPLD symbols, refer to "ADSU8X1" and "ADSU8X2." ADSU8 CI and CO pins do not use the EPLD carry chain.

Unsigned Binary Versus Twos-Complement

ADSU8 can operate on either 8-bit unsigned binary numbers or 8-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

With adder/subtracters, either unsigned binary or twos-complement operations cause an overflow. If the result crosses the overflow boundary, an overflow is generated. Similarly, when the result crosses the carry-out boundary, a carry-out is generated. The following figure shows the ADSU carry-out and overflow boundaries.

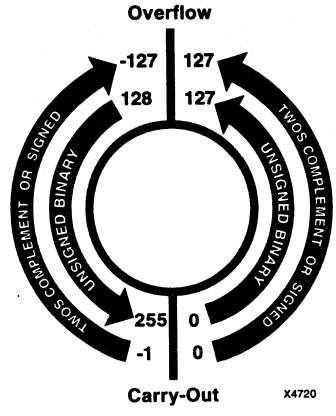


Figure 3-15 ADSU Carry-Out and Overflow Boundaries

Unsigned Binary Operation

For unsigned binary operation, ADSU8 can represent numbers between 0 and 255, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtracter. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds.

An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

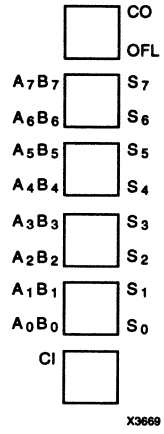
OFL is ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ADSU8 can represent numbers between -128 and +127, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High.

CO is ignored in twos complement operation.

XC4000 Topology



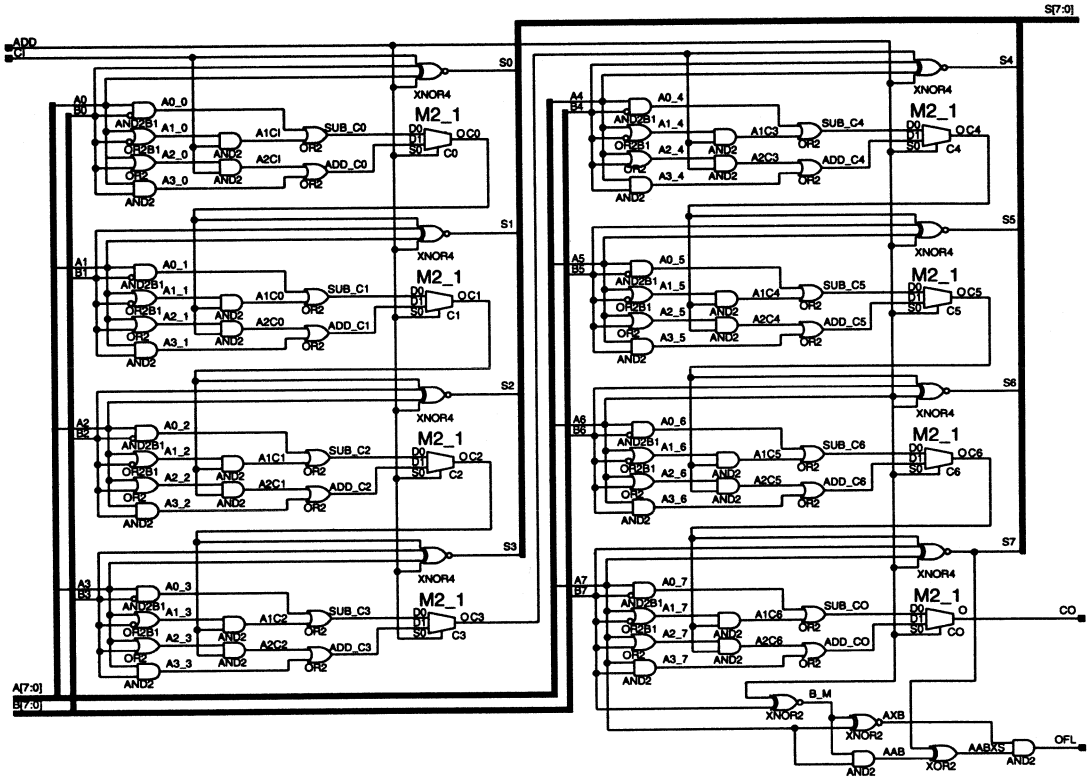


Figure 3-16 ADSU8 XC3000 Implementation

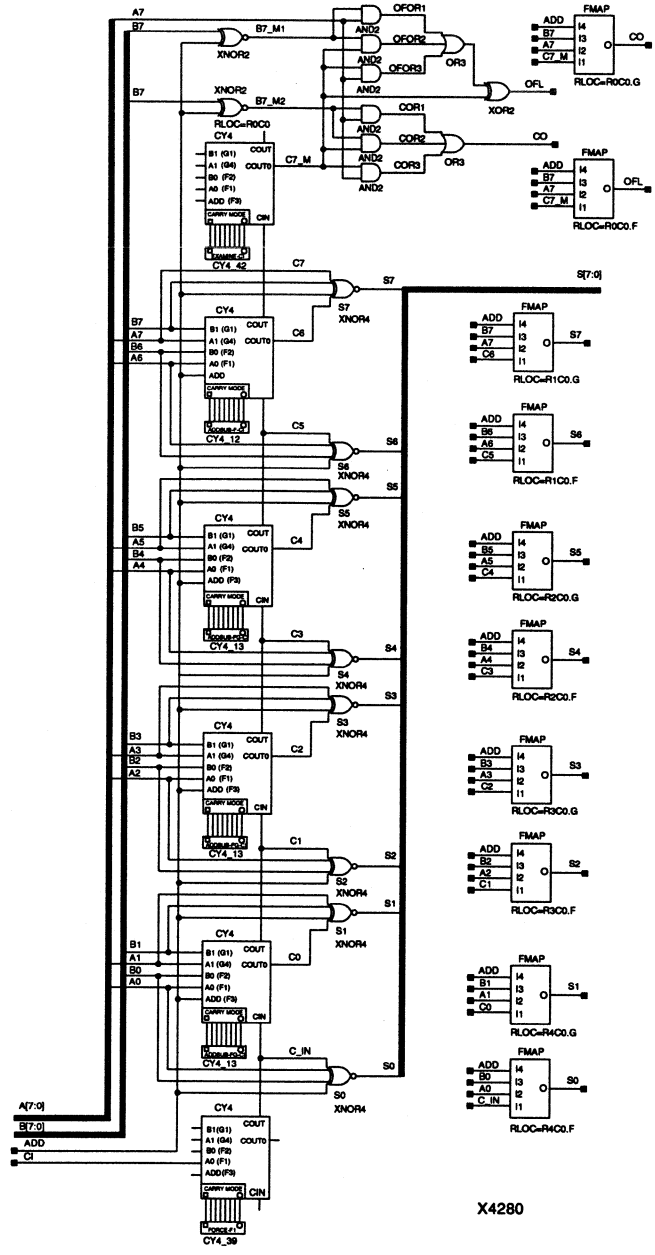
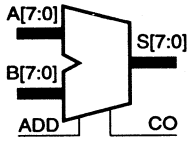


Figure 3-17 ADSU8 XC4000 Implementation

ADSU8X1

8-Bit Cascadable Adder/Subtracter with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

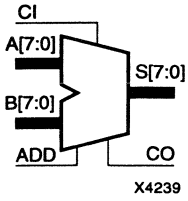
ADSU8X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU8X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order adders through its CO output. When the ADD input is High, two 8-bit words (A7 – A0 and B7 – B0) are added, producing an 8-bit output (S7 – S0) and carry-out (CO). When the ADD input is Low, B7 – B0 is subtracted from A7 – A0, producing a result (S7 – S0) and borrow (CO). In add mode, CO represents a carry-out and is active-High. In subtract mode, CO represents a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ADSU8X2 for the least-significant adder/subtracter and connect an ADD1X1 to its CI input. Connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the adder/subtracter.

Refer to "ADSU1" for truth table derivation.

ADSU8X2

8-Bit Cascadable Adder/Subtracter with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

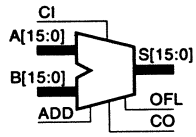
ADSU8X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU8X2 is a high-order adder component cascaded to lower-order adders through its CI input. When the ADD input is High, two 8-bit words (A7 – A0 and B7 – B0) are added with a carry-in (CI), producing an 8-bit output (S7 – S0) and carry-out (CO). When the ADD input is Low, B7 – B0 is subtracted from A7 – A0, producing a result (S7 – S0) and borrow (CO). In add mode, CO represents a carry-out, and CO and CI are active-High. In subtract mode, CO represents a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out.

Refer to "ADSU1" for truth table derivation.

ADSU16

16-Bit Cascadable Adder/Subtracter with Carry-In, Carry-Out, and Overflow



X4381

XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Macro*

* not supported for XC7336 designs

ADSU16 is implemented in the XC4000 family using carry logic and relative location constraints, which assure most efficient logic placement. When the ADD input is High, two 16-bit words (A15 – A0 and B15 – B0) are added with a carry-in (CI), producing a 16-bit sum (S15 – S0) and carry-out (CO) or overflow (OFL). When the ADD input is Low, B15 – B0 is subtracted from A15 – A0, producing a 16-bit difference (S15 – S0) and CO or OFL. In add mode, CO and CI are active-High. In subtract mode, CO and CI are active-Low. OFL is active-High in add and subtract modes. For cascadable EPLD symbols, refer to "ADSU16X1" and "ADSU16X2." ADSU16 CI and CO pins do not use the EPLD carry chain.

Unsigned Binary Versus Twos-Complement

ADSU16 can operate on either 16-bit unsigned binary numbers or 16-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when "overflow" occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when "overflow" occurs.

With adder/subtracters, either unsigned binary or twos-complement operations cause an overflow. If the result crosses the overflow boundary, an overflow is generated. Similarly, when the result crosses the carry-out boundary, a carry-out is generated. The following figure shows the ADSU carry-out and overflow boundaries.

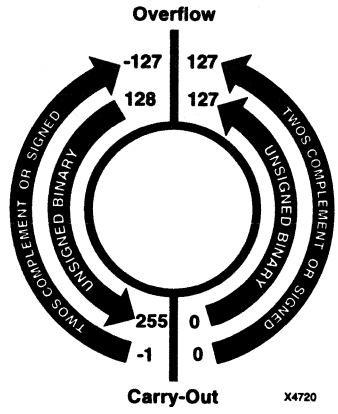


Figure 3-18 ADSU Carry-Out and Overflow Boundaries

Unsigned Binary Operation

For unsigned binary operation, ADSU16 can represent numbers between 0 and 65535, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtracter. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds.

An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows.

```
unsigned overflow = CO XOR ADD
```

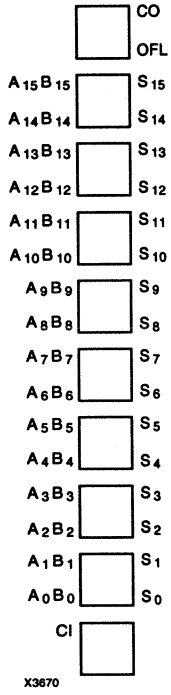
OFL is ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ADSU16 can represent numbers between -32768 and +32767, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High.

CO is ignored in twos-complement operation.

XC4000 Topology



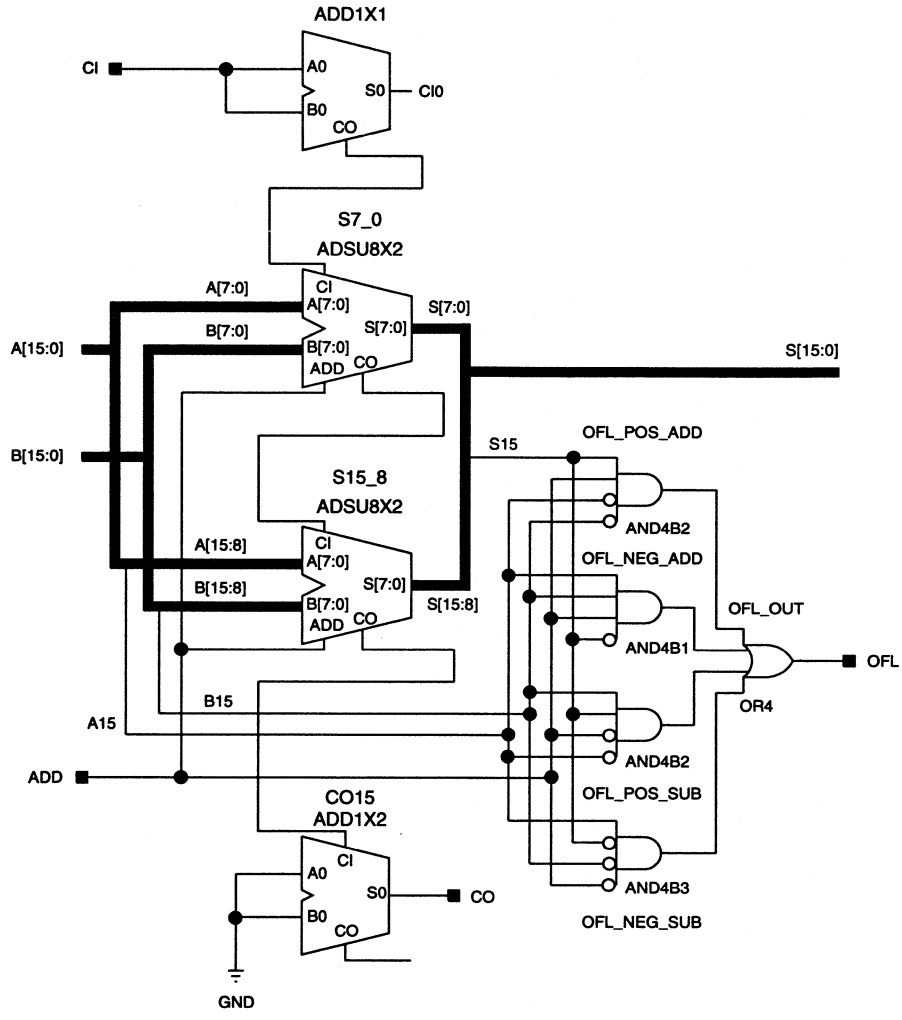
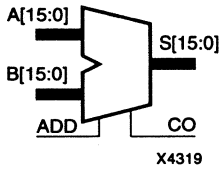


Figure 3-19 ADSU16 XC7000 Implementation

ADSU16X1

16-Bit Cascadable Adder/Subtracter with Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Macro*

* not supported for XC7336 designs

ADSU16X1 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU16X1 is a low-order adder component, which can be used as a stand-alone or cascaded with high-order adders through its CO output. When the ADD input is High, two 16-bit words (A15 – A0 and B15 – B0) are added, producing a 16-bit output (S15 – S0) and carry-out (CO). When the ADD input is Low, B15 – B0 is subtracted from A15 – A0, producing a result (S15 – S0) and borrow (CO). In add mode, CO represents a carry-out and is active-High. In subtract mode, CO represents a borrow and is active-Low.

The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out. If a carry-in is required from general-purpose logic, use an ADSU16X2 for the least-significant adder/subtracter and connect an ADD1X1 to its CI input. Connect your carry-in signal to both the A and B inputs of the ADD1X1 (the S output is not used) to generate a carry into the carry chain for the first bit of the adder/subtracter.

Refer to "ADSU1" for truth table derivation.

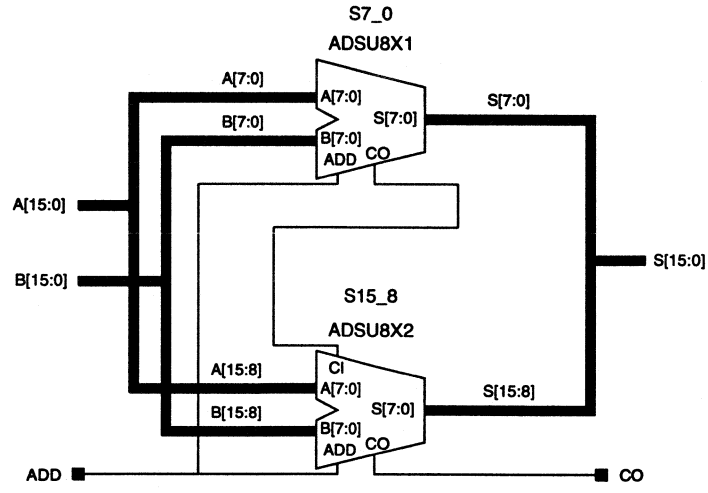
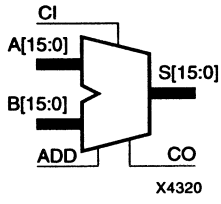


Figure 3-20 ADSU16X1 XC7000 Implementation

ADSU16X2

16-Bit Cascadable Adder/Subtracter with Carry-In and Carry-Out for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Macro*

* not supported for XC7336 designs

ADSU16X2 is implemented using the EPLD arithmetic carry-logic chain for high-speed ripple-carry addition. ADSU16X2 is a high-order adder component cascaded to lower-order adders through its CI input. When the ADD input is High, two 16-bit words (A15 – A0 and B15 – B0) are added with a carry-in (CI), producing a 16-bit output (S15 – S0) and carry-out (CO). When the ADD input is Low, B15 – B0 is subtracted from A15 – A0, producing a result (S15 – S0) and borrow (CO). In add mode, CO represents a carry-out, and CO and CI are active-High. In subtract mode, CO represents a borrow, and CO and CI are active-Low.

The CI input is taken from the EPLD carry chain, and therefore must only be connected to the CO output of another EPLD-specific arithmetic component. The CO output is passed into the EPLD carry chain, and therefore can only be connected to the CI input of another EPLD-specific arithmetic component. To generate a carry-out signal for general-purpose logic, connect an ADD1X2 to the CO output of the adder/subtracter and tie its A and B inputs to GND; the S output becomes the carry-out.

Refer to "ADSU1" for truth table derivation.

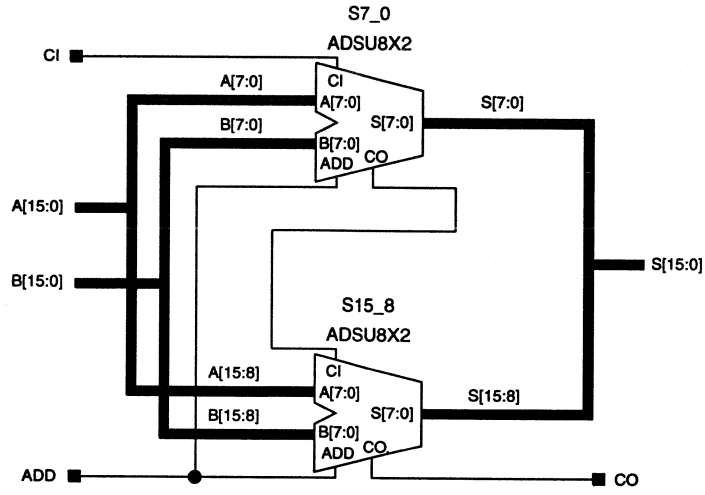


Figure 3-21 ADSU16X2 XC7000 Implementation

AND

2- to 9-Input AND Gates with Inverted and Non-Inverted Inputs

Name	XC2000	XC3000	XC4000	XC7000
AND2 – AND4B4	Primitive	Primitive	Primitive	Primitive
AND5 – AND5B5	Macro	Primitive	Primitive	Primitive
AND6, AND7, AND8, AND9	Macro	Macro	Macro	Primitive

The AND function is performed in the Configurable Logic Block (CLB) function generators for XC2000, XC3000, and XC4000 architectures. AND functions of up to five inputs are available in any combination of inverting and non-inverting inputs. AND functions of six to nine inputs are available with only non-inverting inputs. To make some or all inputs inverting, use external inverters. Because each input uses a CLB resource in FPGAs, replace functions with unused inputs with functions having the appropriate number of inputs.

Available AND gates are shown in the following figure. Refer to the Overview chapter for the combinatorial/AND gate naming convention.

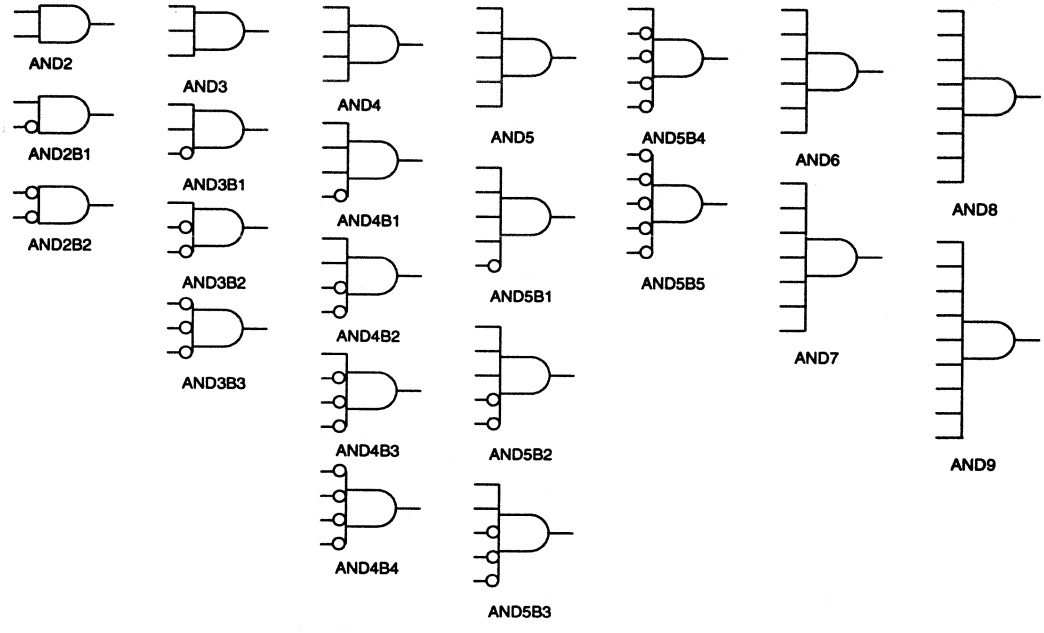
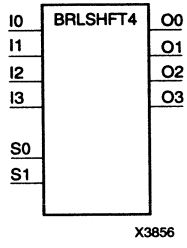


Figure 3-22 AND Gate Representations

BRLSHFT4

4-Bit Barrel Shifter



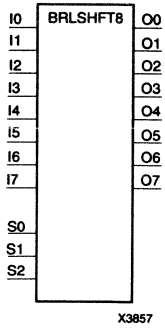
XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Primitive

BRLSHFT4, a 4-bit barrel shifter, can rotate four inputs (I3 – I0) up to four places. The control inputs (S1 and S0) determine the number of positions, from one to four that the data is rotated. The four outputs (O3 – O0) reflect the shifted data inputs.

Inputs						Outputs			
S1	S0	I0	I1	I2	I3	O0	O1	O2	O3
0	0	a	b	c	d	a	b	c	d
0	1	a	b	c	d	b	c	d	a
1	0	a	b	c	d	c	d	a	b
1	1	a	b	c	d	d	a	b	c

BRLSHFT8

8-Bit Barrel Shifter



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Primitive

BRLSHFT8, an 8-bit barrel shifter, can rotate the eight inputs, I7 – I0, up to eight places. The control inputs (S2 – S0) determine the number of positions, from one to eight that the data is rotated. The eight outputs (O7 – O0) reflect the shifted data inputs.

Inputs											Output							
S2	S1	S0	I0	I1	I2	I3	I4	I5	I6	I7	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	a	b	c	d	e	f	g	h	a	b	c	d	e	f	g	h
0	0	1	a	b	c	d	e	f	g	h	b	c	d	e	f	g	h	a
0	1	0	a	b	c	d	e	f	g	h	c	d	e	f	g	h	a	b
0	1	1	a	b	c	d	e	f	g	h	d	e	f	g	h	a	b	c
1	0	0	a	b	c	d	e	f	g	h	e	f	g	h	a	b	c	d
1	0	1	a	b	c	d	e	f	g	h	f	g	h	a	b	c	d	e
1	1	0	a	b	c	d	e	f	g	h	g	h	a	b	c	d	e	f
1	1	1	a	b	c	d	e	f	g	h	h	a	b	c	d	e	f	g

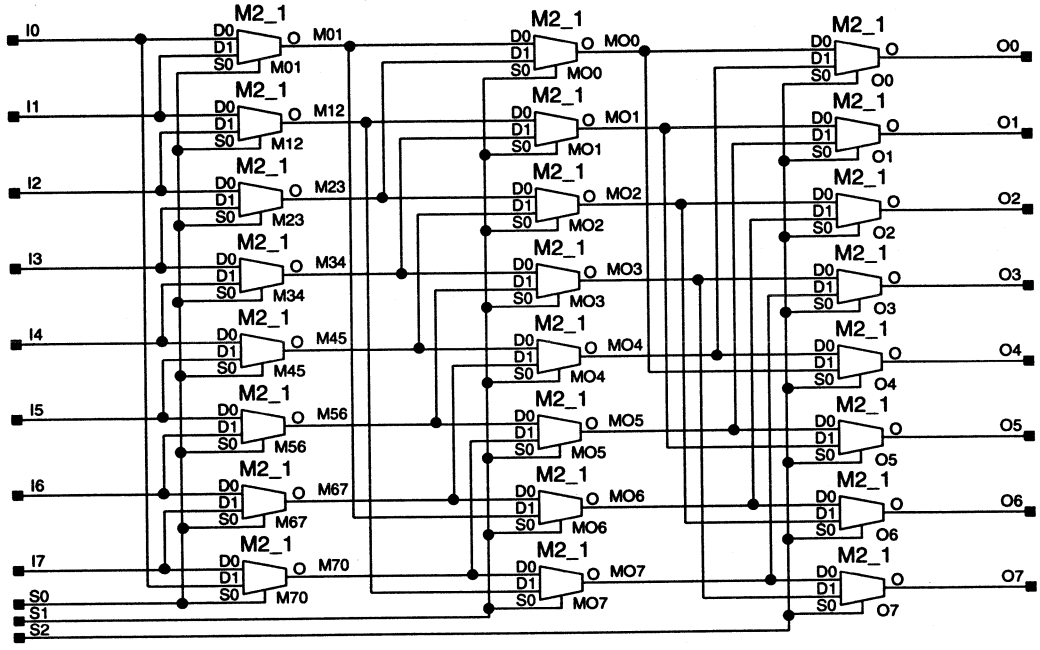
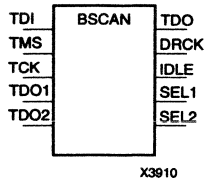


Figure 3-23 BRLSHFT8 XC3000/XC4000 Implementation

BSCAN

Boundary Scan Logic Control Circuit

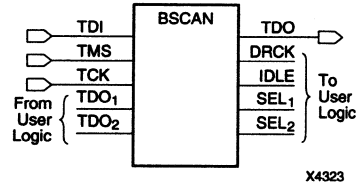


XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The BSCAN symbol indicates that boundary scan logic should be enabled after PLD configuration is complete. It also provides optional access to some special features of the XC4000 boundary scan logic. An overview of the boundary scan interface follows, for complete details, refer to the application note "Boundary Scan in XC4000 Devices" in the Programmable Logic Data Book.

To indicate that boundary scan remain enabled after configuration, connect the BSCAN symbol to the TDI, TMS, TCK, and TDO pads shown in the following figure. The other pins on BSCAN do not need to be connected, unless those special functions are needed. A signal on the TDO_1 input is passed to the external TDO output when the USER1 instruction is executed; the SEL_1 output goes High to indicate that the USER1 instruction is active. The TDO_2 and SEL_2 pins perform a similar function for the USER2 instruction. The DRCK output provides access to the data register clock (generated by the TAP controller). The IDLE output provides access to a version of the TCK input, which is only active while the TAP controller is in the Run-Test-Idle state.

If boundary scan is used only before configuration is complete, do not include the BSCAN symbol in the design, so the TDI, TMS, TCK, and TDO pins can be used for user functions.



BUF, BUF4, BUF8, and BUF16

General-Purpose Buffers

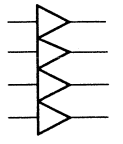
Name	XC2000	XC3000	XC4000	XC7000
BUF	Primitive	Primitive	Primitive	Primitive
BUF4, BUF8, BUF16	N/A	N/A	N/A	Primitive


X3830

BUF is a general purpose, non-inverting buffer. In FPGA architecture, BUF is usually not necessary and is removed by the partitioning software (XNFMap for XC2000/XC3000 and PPR for XC4000). The BUF element can be preserved for reducing the delay on a high fan-out net, for example, by splitting the net and reducing capacitive loading. In this case, the buffer is preserved by attaching an X (explicit) attribute to both the input and output nets of the BUF.

In EPLD architecture, BUF is usually removed, unless you inhibit optimization by applying the OPT=OFF attribute to the BUF symbol or by using the LOGIC_OPT=OFF global attribute.

BUF4


X4614

BUF8

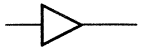

X4615

BUF16


X4616

BUFCE

Global Clock-Enable Buffer for EPLD



X4209

XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7236, XC7272, or XC7336 designs

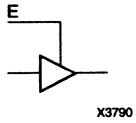
BUFCE, an EPLD-specific global buffer, distributes global clock-enable signals throughout the input-pad registers of an EPLD device. Global clock-enable pins are available on most XC7300 series devices; consult device data sheets for applicability.

BUFCE always acts as an input buffer. To use it in a schematic, connect the input of the BUFCE symbol to an IPAD or an IOPAD that represents the clock-enable signal source. Clock-enable signals generated on-chip must be passed through an OBUF-type buffer before they are connected to a BUFCE. The output of a BUFCE can only be connected to the CE input of an EPLD-specific input-pad register symbol, IFDX1. Each BUFCE can drive any number of IFDX1 registers in a design. The CE input of IFDX1 is active-Low and cannot be inverted.

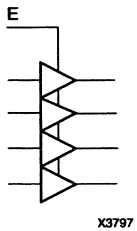
BUFE, BUFE4, BUFE8, and BUFE16

Internal 3-State Buffers

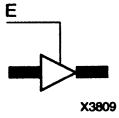
BUFE



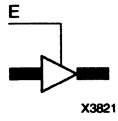
BUFE4



BUFE8



BUFE16



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	Primitive

BUFE, BUFE4, BUFE8, and BUFE16 are single or multiple 3-state buffers with inputs I, I3 – I0, I7 – I0, and so forth; outputs O, O3 – O0, O7 – O0, and so forth; and active-High output enable (E). When E is High, data on the inputs of the buffers is transferred to the corresponding outputs. When E is Low, the output is high impedance (Z state or off). The outputs of the buffers are connected to horizontal longlines in FPGA architectures.

The outputs of separate BUFE symbols can be tied together to form a bus or a multiplexer. Make sure that only one E is High at one time. If none of the E inputs is active-High, a "weak-keeper" circuit (FPGA) keeps the output bus from floating, but does not guarantee that the bus remains at the last value driven onto it.

The E in XC3000/XC4000 BUFE macros is implemented by using a BUFT with an inverter on the active-Low enable (T) pin. This inverter can add an extra level of logic to the data path. Pull-up resistors can be used to establish a High logic level if all BUFE elements are off. Pull-up resistors are always assumed for EPLD designs. The following figure shows BUFE XC3000/XC4000 implementation.

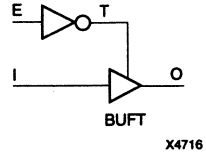


Figure 3-24 BUFE XC3000/XC4000 Implementation

Inputs		Outputs
E	I	O
0	X	Z
1	1	1
1	0	0

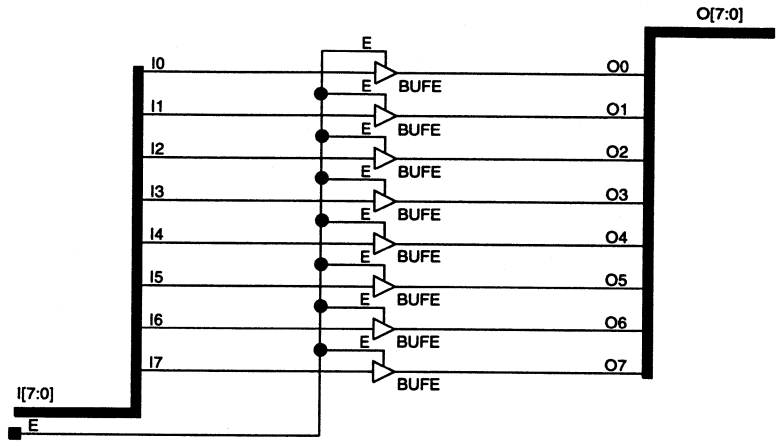


Figure 3-25 BUFE8 XC3000/4000 Implementation

BUFFOE

Global Fast Output Enable Buffer for EPLD

X4210

XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7272 designs

BUFFOE, an EPLD-specific global buffer, distributes global output-enable signals throughout the output pad drivers of an EPLD device. Global Fast Output Enable (FOE) pins are available on most XC7000 architecture devices; consult device data sheets for applicability.

BUFFOE always acts as an input buffer. To use it in a schematic, connect the input of the BUFFOE symbol to an IPAD or an IOPAD representing the FOE signal source. FOE signals generated on-chip must be passed through an OBUF-type buffer before they are connected to the BUFFOE. The output of a BUFFOE can only connect to the E input of an EPLD-specific 3-state output buffer symbol, OBUFEX1. Each BUFFOE can drive any number of OBUFEX1 buffers in a design. The E input of OBUFEX1 is active-High and cannot be inverted.

BUFG

Global Clock Buffer

X3831

XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	Primitive	Primitive

BUFG, an architecture-independent global buffer, distributes high fan-out clock signals throughout a PLD device. The Xilinx implementation software converts each BUFG to an appropriate type of global buffer for the target PLD device (GCLK or ACLK for XC2000 and XC3000, BUFGP or BUFGS for XC4000, and FastCLK for XC7000).

For an XC2000 or XC3000 design, you can use a maximum of two BUFG symbols (assuming that no specific GCLK or ACLK buffer is specified). For an XC4000 design, you can use a maximum of eight BUFG symbols (assuming that no specific BUFGP or BUFGS buffers are specified). For an XC7000 design, consult the device date sheet for the number of available FastCLK pins.

To use a BUFG in a schematic, connect the input of the BUFG symbol to the clock source. Depending on the target PLD family, the clock source can be an external PAD symbol, an IBUF symbol, or internal logic. For XC2000 designs, the BUFG cannot be sourced directly from the PAD symbol; an IBUF must be included between the PAD and BUFG. For a negative-edge clock input, insert an INV (inverter) symbol between the BUFG output and the clock input. The inversion is implemented at the Configurable Logic Block (CLB) or Input Output Block (IOB) clock pin.

Note: For XC2000 and XC3000 designs, XNFPrep always selects an ACLK, then a GCLK. For XC4000 designs, it always selects a BUFGS before a BUFGP. If you want to use a specific type of buffer, manually instantiate it.

For XC7000 designs, BUFG always acts as an input buffer. Connect the input of BUFG to an IPAD or an IOPAD that represents the FastCLK signal source. FastCLK signals generated on-chip must be passed through an OBUF-type buffer before connecting to BUFG. Each BUFG can drive any number of register clocks (or ILD latch-enable inputs) in a design. All clock inputs driven by BUFG are active-High and cannot be inverted.

BUFGP

Primary Global Buffer for Driving Clocks or Longlines (Four per PLD Device)

X3902

XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	Primitive

BUFGP, a primary global buffer, is used to distribute high fan-out clock or control signals throughout PLD devices. In XC7000 EPLD designs, BUFGP is treated like BUFG. A BUFGP provides direct access to Configurable Logic Block (CLB) and Input Output Block (IOB) clock pins and limited access to other CLB inputs. Four BUFGPs are available on each XC4000 device, one in each corner. The input to a BUFGP comes only from a dedicated IOB.

Alongside each column of CLBs in an XC4000 device are four global vertical lines, which are in addition to the standard vertical longlines. Each one of the four global vertical lines can drive the CLB clock (K) pin directly. In addition, one of the four lines can drive the F3 pin, a second line can drive the G1 pin, a third can drive the C3 pin, and a fourth can drive the C1 pin. Each of the four BUFGPs drives one of these global vertical lines. These same vertical lines are also used for the secondary global buffers (refer to "BUFGS" for more information).

Because of its structure, a BUFGP can always access a clock pin directly. However, it can access only one of the F3, G1, C3, or C1 pins, depending on the corner in which the BUFGP is placed. When the required pin cannot be accessed directly from the vertical line, PPR feeds the signal through another CLB and uses general purpose routing to access the load pin.

To use a BUFGP in a schematic, connect the input of the BUFGP element directly to the PAD symbol. Do not use any IBUFs, because the signal comes directly from a dedicated IOB. The output of the BUFGP is then used throughout the schematic. For a negative-edge clock, insert an INV (inverter) element between the output of the BUFGP and the clock input. This inversion is performed inside each CLB or IOB.

A BUFGP can be sourced by an internal signal, but PPR must use the dedicated IOB to drive the BUFGP, which means that the IOB is not available for use by other signals. If possible, use a BUFGS instead, because they can be sourced internally without using an IOB.

The dedicated inputs for BUFGPs are identified by the names PGCK1 through PGCK4 in XC4000 pinouts. The package pin that drives the BUFGP depends on which corner the BUFGP is placed by PPR.

BUFGS

Secondary Global Buffer for Driving Clocks or Longlines (Four per PLD Device)

X3904

XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	Primitive

BUFGS, a secondary global buffer, distributes high fan-out clock or control signals throughout a PLD device. In XC7000 EPLD designs, BUFGS is treated like BUFG. BUFGS provides direct access to Configurable Logic Block (CLB) clock pins and limited access to other CLB inputs. Four BUFGSs are available on each XC4000 device, one in each corner. The input to a BUFGS comes either from a dedicated Input Output Block (IOB) or from an internal signal.

Alongside each column of CLBs in an XC4000 device are four global vertical lines, which are in addition to the standard vertical longlines. Each one of the four global vertical lines can drive the CLB clock (K) pin directly. In addition, one of the four lines can drive the F3 pin, a second line can drive the G1 pin, a third can drive the C3 pin, and a fourth can drive the C1 pin. Each of the four BUFGSs can drive any of these global vertical lines and are also used as the primary global buffers (refer also to BUFGP for information).

Because of its structure, a BUFGS can always access a clock pin directly. Because the BUFGS is more flexible than the BUFGP, it can use additional global vertical lines to access the F3, G1, C3, and C1 pins, but requires multiple vertical lines in the same column. If the vertical lines in a given column are already used for BUFGPs or another BUFGS, PPR might have to feed signals through other CLBs to reach the load pins.

To use a BUFGS in a schematic, connect the input of the BUFGS element either directly to the PAD symbol (for an external input) or to an internally sourced net. For an external signal, do not use any IBUFs, because the signal comes directly from the dedicated IOB. The output of the BUFGS is then used throughout the schematic. For a negative-edge clock, insert an INV (inverter) element between the output of the BUFGS and the clock input. This inversion is performed inside each CLB or IOB.

The dedicated inputs for BUFGSs are identified by the names SGCK1 through SGCK4 in XC4000 pinouts. The package pin that drives the BUFGS depends on which corner the BUFGS is placed by PPR.

BUFOD

Open-Drain Buffer

X3903

XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

BUFOD is a buffer with input (I) and open-drain output (O). When the input is Low, the output is Low. When the input is High, the output is off. To establish an output High level, a pull-up resistors is tied to output O. One pull-up resistor uses the least power; two pull-up resistors achieve the fastest Low-to-High speed.

To indicate two pull-up resistors, append a DOUBLE parameter to the pull-up symbol attached to the output (O) node. Refer to the appropriate CAE tool interface user guide for details.

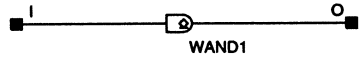


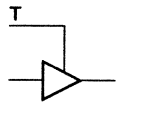
Figure 3-26 BUFOD XC4000 Implementation

BUFT, BUFT4, BUFT8, and BUFT16

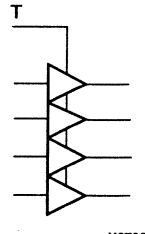
Internal 3-State Buffers

Name	XC2000	XC3000	XC4000	XC7000
BUFT	N/A	Primitive	Primitive	Primitive
BUFT4, BUFT8, BUFT16	N/A	Macro	Macro	Primitive

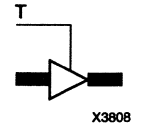
BUFT



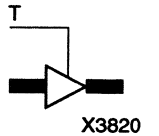
BUFT4



BUFT8



BUFT16



BUFT, BUFT4, BUFT8, and BUFT16 are single or multiple 3-state buffers with inputs I, I3 – I0, I7 – I0, and so forth; outputs O, O3 – O0, O7 – O0, and so forth; and active-Low output enable (T). When T is Low, data on the inputs of the buffers is transferred to the corresponding outputs. When T is High, the output is high impedance (Z state or off). The outputs of the buffers are connected to horizontal longlines in FPGA architectures.

The outputs of separate BUFT symbols can be tied together to form a bus or a multiplexer. Make sure that only one T is Low at one time. If none of the T inputs is active (Low), a "weak-keeper" circuit (FPGAs) prevents the output bus from floating, but does not guarantee that the bus remains at the last value driven onto it. Pull-up resistors can be used to establish a High logic level if all BUFT elements are off. Pull-up resistors are always assumed for EPLD designs.

Inputs		Outputs
T	I	O
1	X	Z
0	1	1
0	0	0

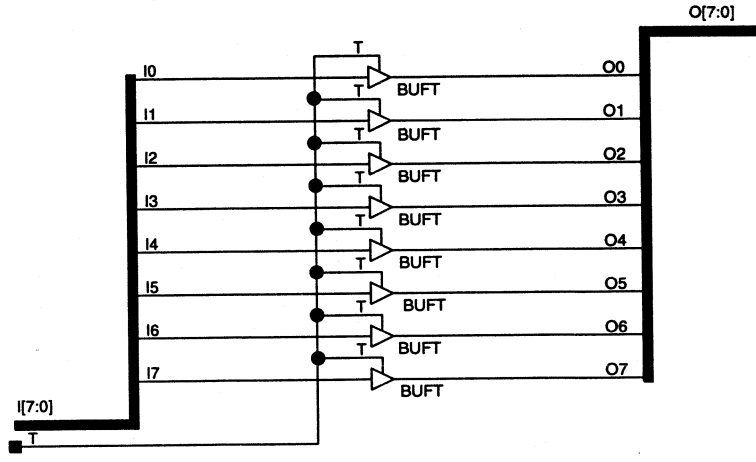
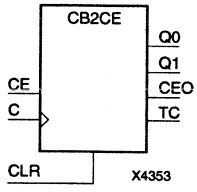


Figure 3-27 BUFT8 XC3000/4000 Implementation

CB2CE

2-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB2CE is a 2-stage, 2-bit, synchronous, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q1 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q1 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when both Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

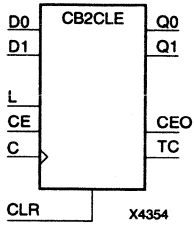
Inputs			Outputs		
CLR	CE	C	Q1 – Q0	TC	CEO
1	X	X	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q1•Q0)

CEO = (TC•CE)

CB2CLE

2-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB2CLE is a 2-stage, 2-bit, synchronous, loadable, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q1 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock (C) transition. The data on the D1 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock transition, independent of the state of clock enable (CE). The outputs (Q1 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when both Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
CLR	L	CE	C	D1 – D0	Q1 – Q0	TC	CEO
1	X	X	X	X	0	0	0
0	1	X	↑	D	d1 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

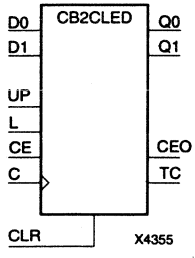
TC = (Q1•Q0)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

CB2CLED

2-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB2CLED is a 2-stage, 2-bit, synchronous, loadable, clearable, cascadable, bidirectional binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q1 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D1 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q1 – Q0) decrement when CE is High and UP is Low during the Low-to-High clock transition. The outputs (Q1 – Q0) increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the clock enable out (CEO) output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage. For EPLD designs, refer to "CB2X1" for high-performance cascadable, bidirectional counters.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs						Outputs		
CLR	L	CE	C	UP	D1 – D0	Q1 – Q0	TC	CEO
1	X	X	X	X	X	0	0	0
0	1	X	↑	X	D	d1 – d0	TC	CEO
0	0	0	X	X	X	No Chg	No Chg	0
0	0	1	↑	1	X	Inc	TC	CEO
0	0	1	↑	0	X	Dec	TC	CEO

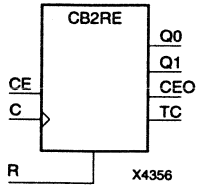
TC = (Q1•Q0•UP) + ($\overline{Q1}$•$\overline{Q0}$•$\overline{UP}$)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

CB2RE

2-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB2RE is a 2-stage, 2-bit, synchronous, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored and data (Q1 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q1 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when both Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

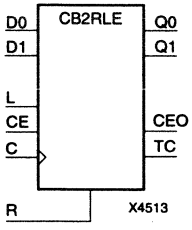
Inputs				Outputs	
R	CE	C	Q1 – Q0	TC	CEO
1	X	↑	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q1•Q0)

CEO = (TC•CE)

CB2RLE

2-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB2RLE is a 2-stage, 2-bit, synchronous, loadable, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input. The synchronous R, when High, overrides all other inputs and resets the Q1 – Q0, terminal count (TC), and clock enable out (CEO) outputs to Low on the Low-to-High clock (C) transition.

The data on the D1 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of CE. The outputs (Q1 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High. The CEO output is High when all Q outputs and CE are High to allow direct cascading of counters. Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and by connecting the C, L, and R inputs in parallel. The maximum length of the counter is determined by the accumulated CE-to-CEO propagation delays versus the clock period.

The counter is asynchronously reset, output Low, when power is applied or when global reset or master reset is active. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
R	L	CE	C	D1 – D0	Q1 – Q0	TC	CEO
1	X	X	↑	X	0	0	0
0	1	X	↑	D	d1 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

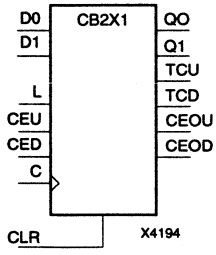
TC = Q1•Q0

CEO = TC•CE

dn = state of referenced input one set-up time prior to active clock transition

CB2X1

2-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB2X1 is a 2-stage, 2-bit, synchronous, loadable, clearable, bidirectional binary counter. CB2X1 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the EPLD architecture.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored, data outputs (Q1 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, independent of clock transitions. The data on the D1 – D0 inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The outputs (Q1 – Q0) increment when CEU is High, provided CLR and L are Low, during the Low-to-High clock transition. The outputs (Q1 – Q0) decrement when CED is High, provided CLR and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The clock, L and CLR inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component, resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced. The counter is initialized to zero (TCU Low and TCD High) when the device is powered-up or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
CLR	**L**	**CEU**	**CED**	**C**	**D1 – D0**	**Q1 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	X	X	0	0	1	0	CEOD
0	1	X	X	↑	D	d1 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	CEOU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	CEOD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q1•Q0

TCD = $\overline{Q1}$•$\overline{Q0}$

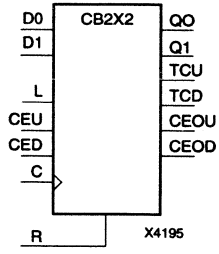
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CB2X2

2-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB2X2 is a 2-stage, 2-bit, synchronous, loadable, resettable, bidirectional binary counter. CB2X2 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the EPLD architecture.

The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored, data outputs (Q1 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, on the Low-to-High clock (C) transition. The data on the D1 – D0 inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The outputs (Q1 – Q0) increment when CEU is High, provided R and L are Low during the Low-to-High clock transition. The outputs (Q1 – Q0) decrement when CED is High, provided R and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The C, L, and R inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component, resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced. The counter is initialized to zero (TCU Low and TCD High) when power is applied or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
R	**L**	**CEU**	**CED**	**C**	**D1 – D0**	**Q1 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	↑	X	0	0	1	0	CEOD
0	1	X	X	↑	D	d1 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	CEOU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	CEOD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q1•Q0

TCD = $\overline{Q1}$•$\overline{Q0}$

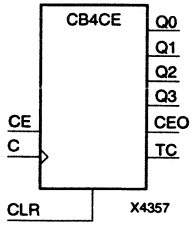
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CB4CE

4-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB4CE is a 4-stage, 4-bit, synchronous, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q3 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q3 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

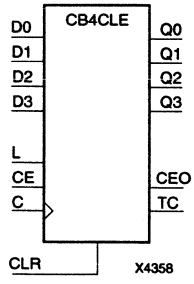
Inputs			Outputs		
CLR	CE	C	Q3 – Q0	TC	CEO
1	X	X	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q3•Q2•Q1•Q0)

CEO = (TC•CE)

CB4CLE

4-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB4CLE is a 4-stage, 4-bit, synchronous, loadable, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q3 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D3 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q3 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
CLR	L	CE	C	D3 – D0	Q3 – Q0	TC	CEO
1	X	X	X	X	0	0	0
0	1	X	↑	D	d3 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

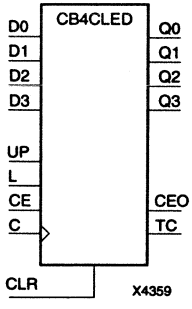
TC = (Q3•Q2•Q1•Q0)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

CB4CLED

4-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB4CLED is a 4-stage, 4-bit, synchronous, loadable, clearable, cascadable, bidirectional binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q3 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D3 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q3 – Q0) decrement when CE is High and UP is Low during the Low-to-High clock transition. The outputs (Q3 – Q0) increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the count enable out (CEO) output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage. For EPLD designs, refer to "CB4X1" for high-performance cascadable, bidirectional counters.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs						Outputs		
CLR	L	CE	C	UP	D3 – D0	Q3 – Q0	TC	CEO
1	X	X	X	X	X	0	0	0
0	1	X	↑	X	D	d3 – d0	TC	CEO
0	0	0	X	X	X	No Chg	No Chg	0
0	0	1	↑	1	X	Inc	TC	CEO
0	0	1	↑	0	X	Dec	TC	CEO

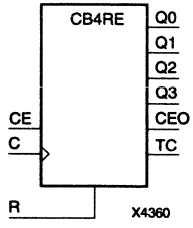
$TC = (Q3 \bullet Q2 \bullet Q1 \bullet Q0 \bullet UP) + (\overline{Q3} \bullet \overline{Q2} \bullet \overline{Q1} \bullet \overline{Q0} \bullet \overline{UP})$

$CEO = (TC \bullet CE)$

dn = state of referenced clock one set-up time prior to active clock transition

CB4RE

4-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB4CE is a 4-stage, 4-bit, synchronous, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored and data (Q3 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock (C) transition. The outputs (Q3 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

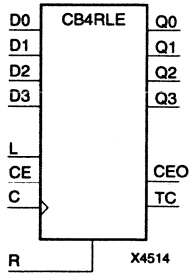
Inputs			Outputs		
R	CE	C	Q3 – Q0	TC	CEO
1	X	↑	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q3•Q2•Q2•Q0)

CEO = (TC•CE)

CB4RLE

4-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB4RLE is a 4-stage, 4-bit, synchronous, loadable, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input. The synchronous R, when High, overrides all other inputs and resets the Q3 – Q0, TC, and CEO outputs to Low on the Low-to-High clock (C) transition. The data on the D3 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of CE. The outputs (Q3 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High. The CEO output is High when all Q outputs and CE are High to allow direct cascading of counters.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and by connecting the C, L, and R inputs in parallel. The maximum length of the counter is determined by the accumulated CE-to-CEO propagation delays versus the clock period.

The counter is asynchronously reset, output Low, when power is applied or when global reset or master reset is active. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
R	L	CE	C	D3 – D0	Q3 – Q0	TC	CEO
1	X	X	↑	X	0	0	0
0	1	X	↑	D	d3 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	TC

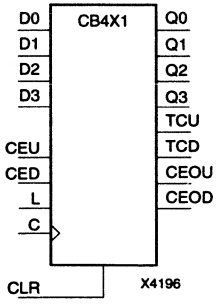
TC = Q3•Q2•Q1•Q0

CEO = TC•CE

dn = state of referenced input one set-up time prior to active clock transition

CB4X1

4-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB4X1 is a 4-stage, 4-bit, synchronous, loadable, clearable, bidirectional binary counter. CB4X1 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions, to support high-speed cascading in the EPLD architecture.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored, data outputs (Q3 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, independent of clock transitions. The data on the D3 – D0 inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The outputs (Q3 – Q0) increment when CEU is High, provided CLR and L are Low, during the Low-to-High clock transition. The outputs (Q3 – Q0) decrement when CED is High, provided CLR and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The clock, L, and CLR inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component, resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced. The counter is initialized to zero (TCU Low and TCD High) when the device is powered-up or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
CLR	**L**	**CEU**	**CED**	**C**	**D3 – D0**	**Q3 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	X	X	0	0	1	0	CEOD
0	1	X	X	↑	D	d3 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	CEOU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	CEOD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q3•Q2•Q1•Q0

TCD = $\overline{Q3}$•$\overline{Q2}$•$\overline{Q1}$•$\overline{Q0}$

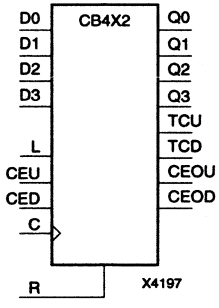
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CB4X2

4-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB4X2 is a 4-stage, 4-bit, synchronous, loadable, resettable, bidirectional binary counter. CB4X2 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the EPLD architecture.

The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored, data outputs (Q3 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, on the Low-to-High clock (C) transition. The data on the D3 – D0 inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The outputs (Q3 – Q0) increment when CEU is High, provided R and L are Low, during the Low-to-High clock transition. The outputs (Q3 – Q0) decrement when CED is High, provided R and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The C, L, and R inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced. The counter is initialized to zero (TCU Low and TCD High) when the device is powered-up or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
R	**L**	**CEU**	**CED**	**C**	**D3 – D0**	**Q3 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	↑	X	0	0	1	0	CED
0	1	X	X	↑	D	d3 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	TCU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	TCD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q3•Q2•Q1•Q0

TCD = $\overline{Q3}$•$\overline{Q2}$•$\overline{Q1}$•$\overline{Q0}$

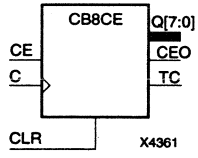
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CB8CE

8-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB8CE is an 8-stage, 8-bit, synchronous, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q7 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs		
CLR	CE	C	Q7 – Q0	TC	CEO
1	X	X	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q7•Q6•Q5•Q4•...•Q0)

CEO = (TC•CE)

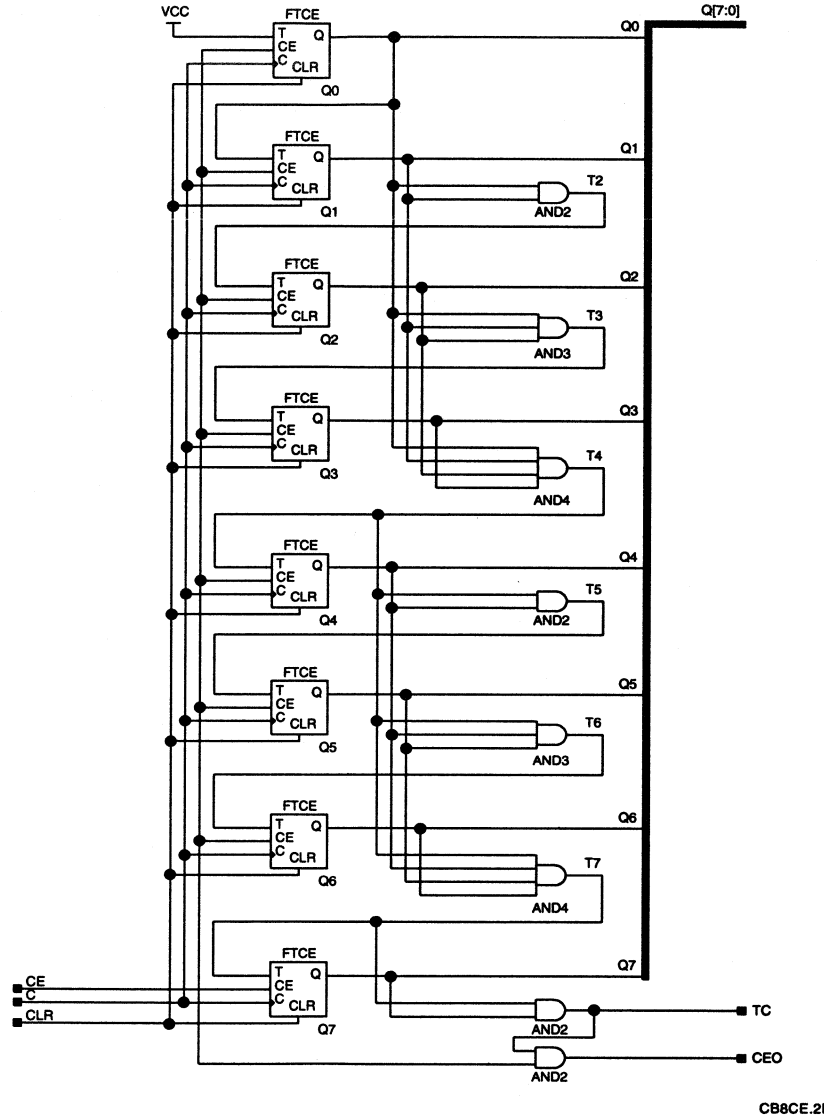


Figure 3-28 CB8CE XC2000 Implementation

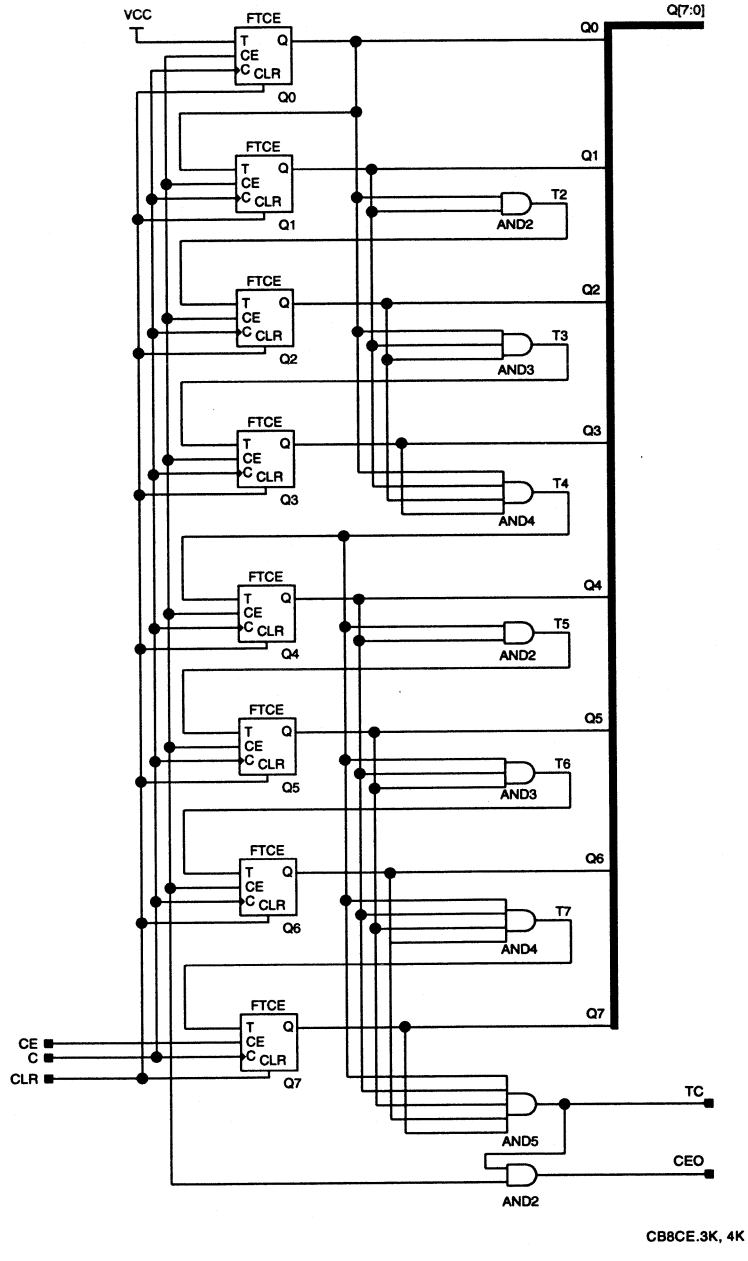
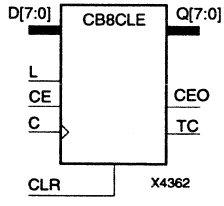


Figure 3-29 CB8CE XC3000/XC4000 Implementation

CB8CLE

8-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB8CLE is an 8-stage, 8-bit, synchronous, loadable, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D7 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q7 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and by connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
CLR	L	CE	C	D7 – D0	Q7 – Q0	TC	CEO
1	X	X	X	X	0	0	0
0	1	X	↑	D	d7 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

TC = (Q7•Q6•Q5•Q4•...•Q0)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

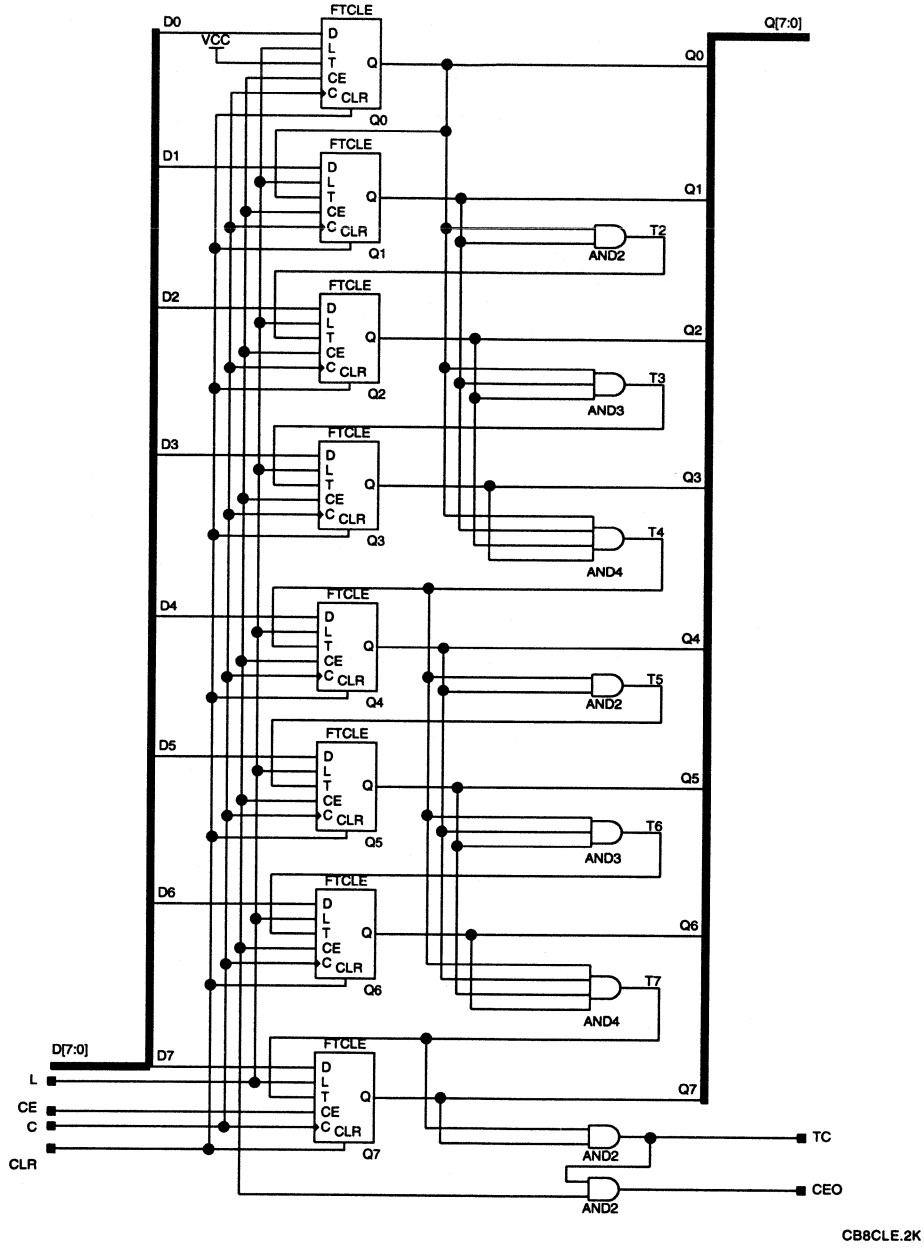


Figure 3-30 CB8CLE XC2000 Implementation

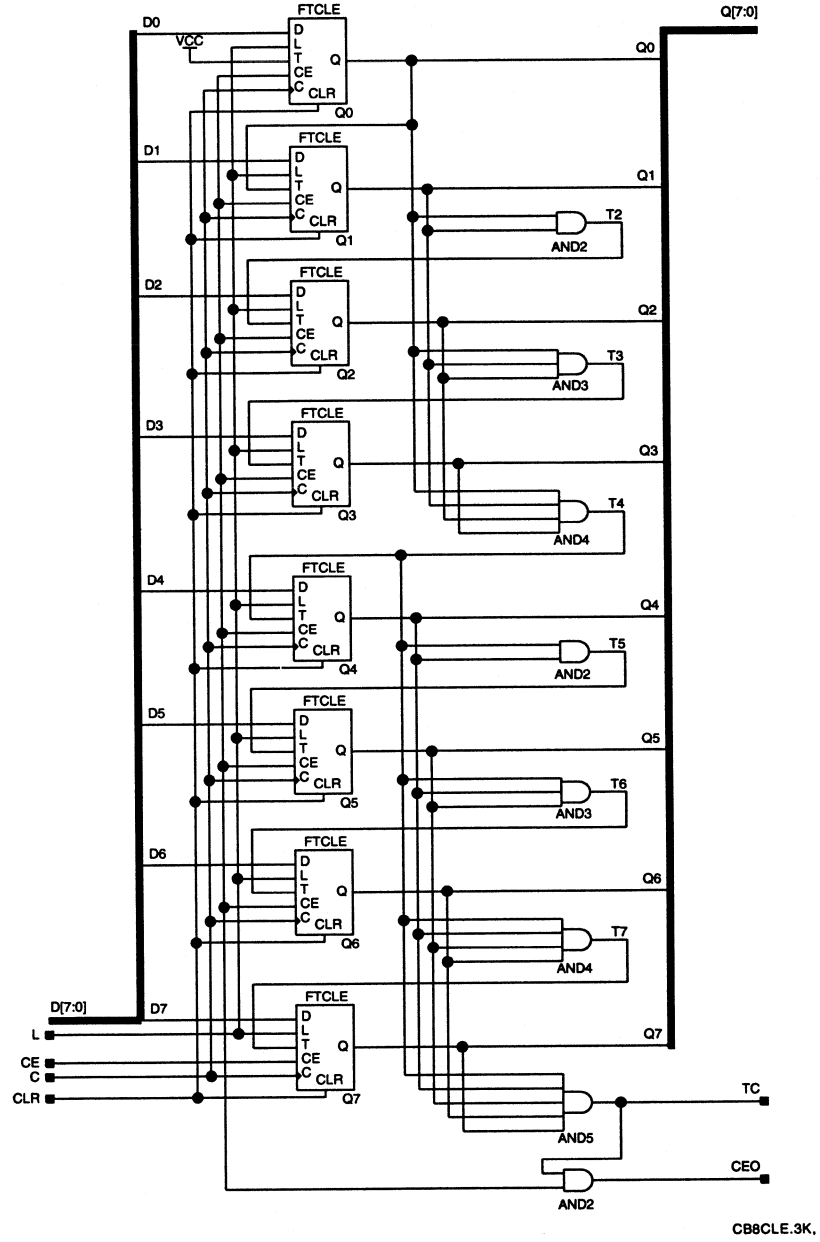
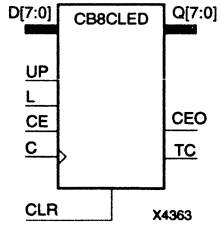


Figure 3-31 CB8CLE XC3000/4000 Implementation

CB8CLED

8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB8CLED is an 8-stage, 8-bit, synchronous, loadable, clearable, cascadable, bidirectional binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D7 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of CE. The outputs (Q7 – Q0) decrement when CE is High and UP is Low during the Low-to-High clock transition. The outputs (Q7 – Q0) increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the count enable out (CEO) output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage. For EPLD designs, refer to "CB8X1" for high-performance cascadable, bidirectional counters.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs						Outputs		
CLR	L	CE	C	UP	D7 – D0	Q7 – Q0	TC	CEO
1	X	X	X	X	X	0	0	0
0	1	X	↑	X	D	d7 – d0	TC	CEO
0	0	0	X	X	X	No Chg	No Chg	0
0	0	1	↑	1	X	Inc	TC	CEO
0	0	1	↑	0	X	Dec	TC	CEO

$TC = (Q7 \bullet Q6 \bullet Q5 \bullet ... \bullet Q0 \bullet UP) + (\overline{Q7} \bullet \overline{Q6} \bullet \overline{Q5} \bullet ... \bullet \overline{Q0} \bullet \overline{UP})$

$CEO = (TC \bullet CE)$

dn = state of referenced clock one set-up time prior to active clock transition

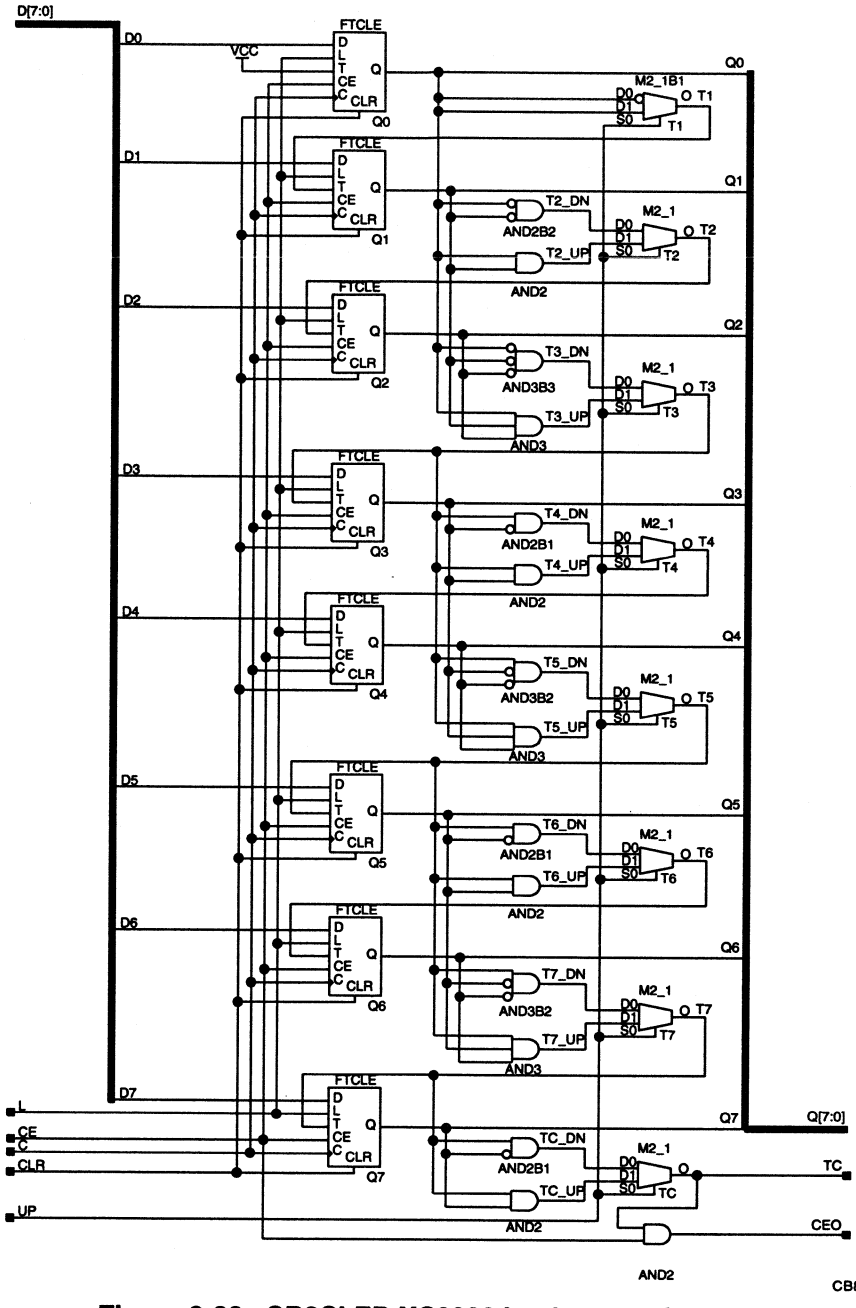


Figure 3-32 CB8CLED XC2000 Implementation

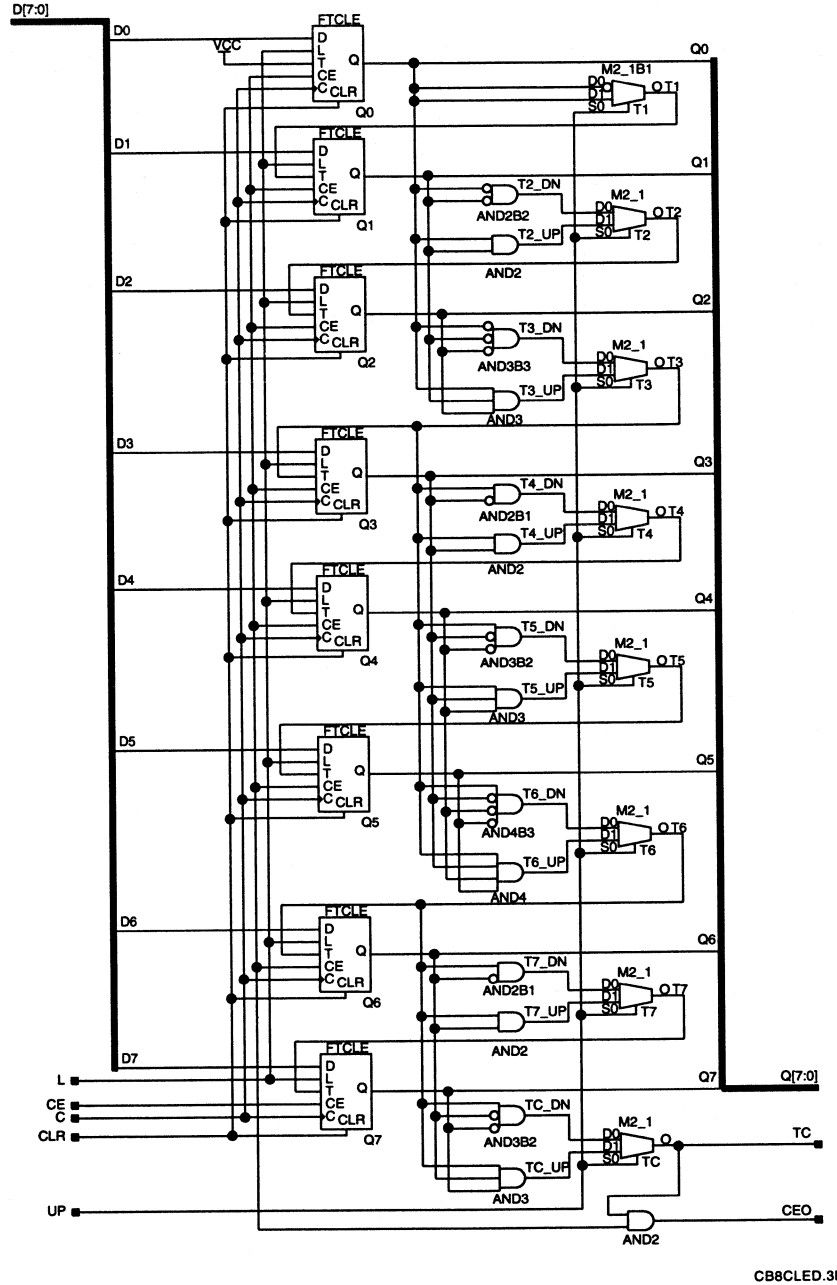


Figure 3-33 CB8CLED XC3000 Implementation

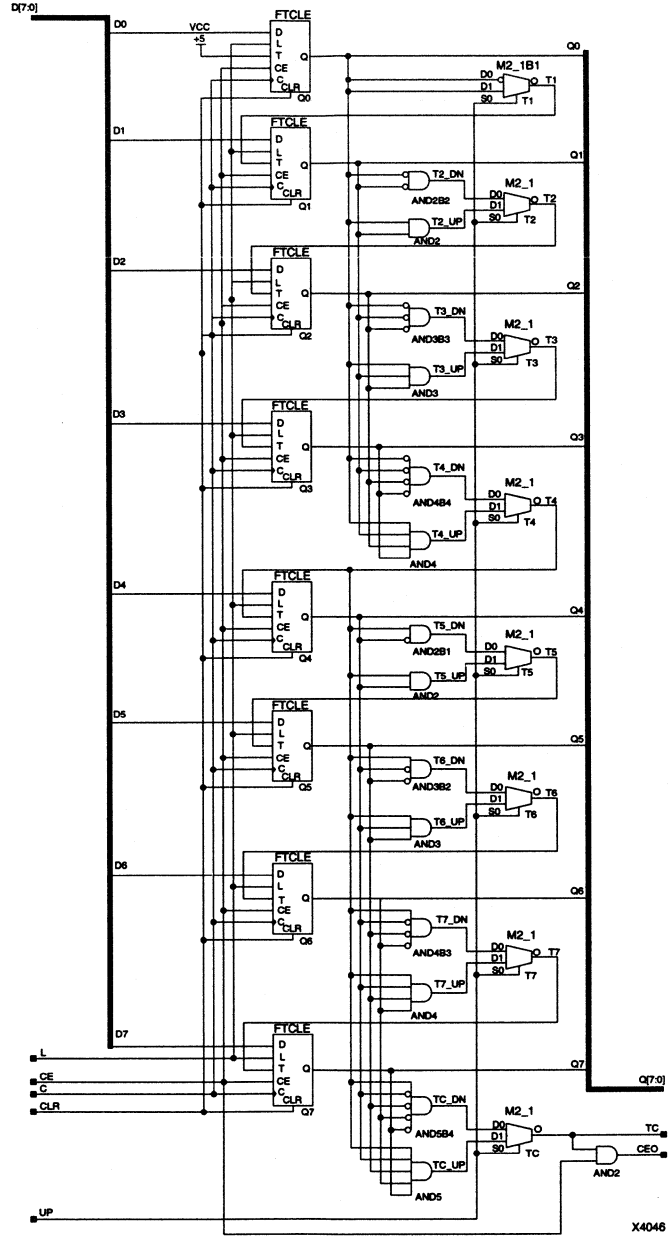
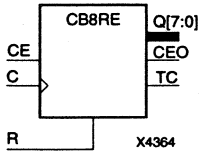


Figure 3-34 CB8CLED XC4000 Implementation

CB8RE

8-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB8CE is an 8-stage, 8-bit, synchronous, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock (C) transition. The outputs (Q7 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs		
R	**CE**	**C**	**Q7 – Q0**	**TC**	**CEO**
1	X	↑	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q7•Q6•Q5•...•Q0)

CEO = (TC•CE)

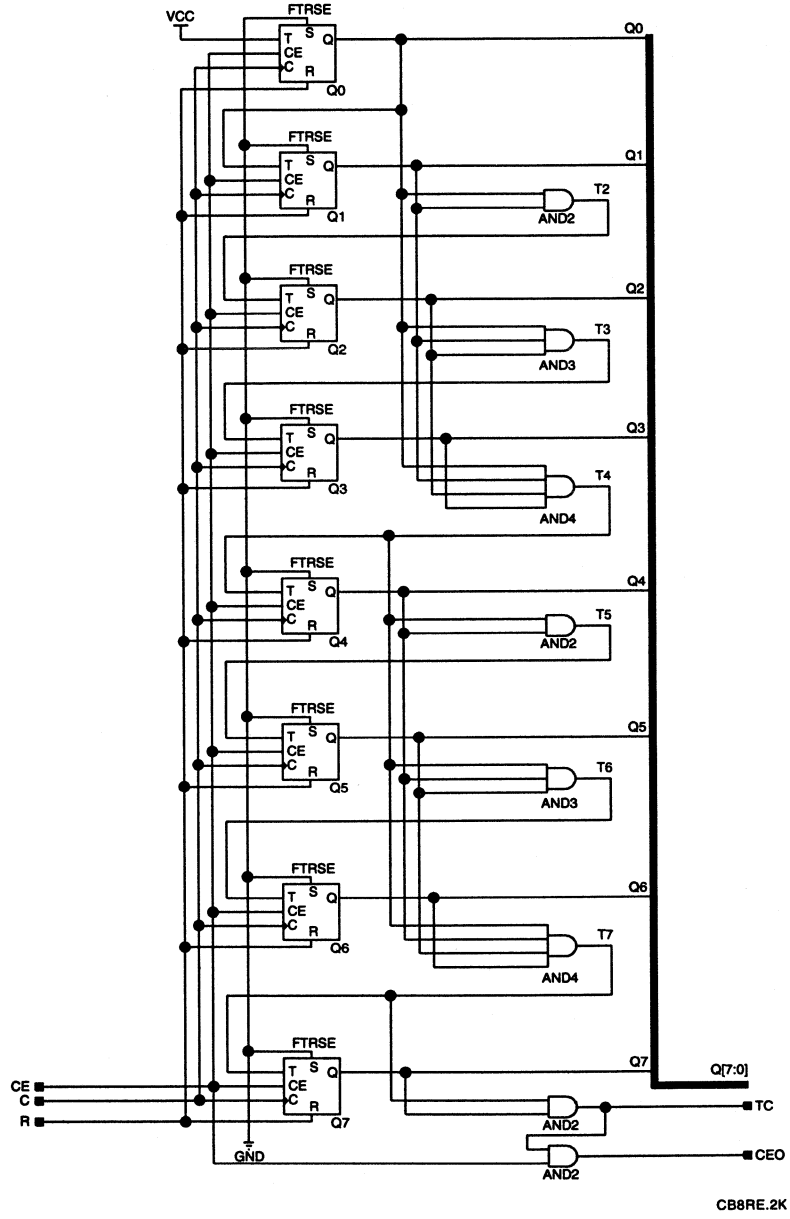


Figure 3-35 CB8RE XC2000 Implementation

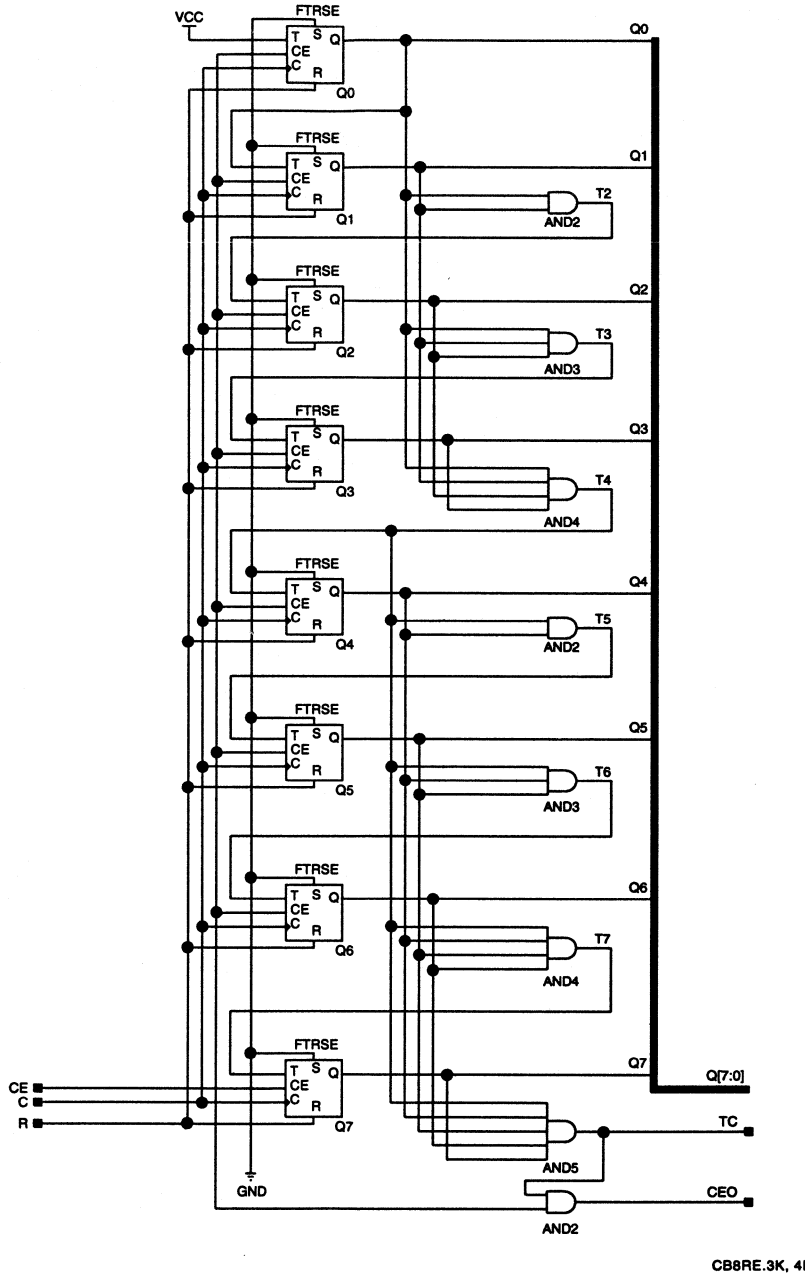
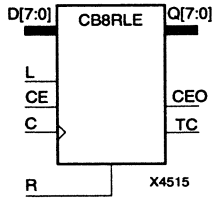


Figure 3-36 CB8RE XC3000/4000 Implementation

CB8RLE

8-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB8RLE is an 8-stage, 8-bit, synchronous, loadable, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input. The synchronous R, when High, overrides all other inputs and resets the Q7 – Q0, TC, and CEO outputs to Low on the Low-to-High clock (C) transition. The data on the D7 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of CE.

The outputs (Q7 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High. The CEO output is High when all Q outputs and CE are High, to allow direct cascading of counters.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C, L, and R inputs in parallel. The maximum length of the counter is determined by the accumulated CE-to-CEO propagation delays versus the clock period.

The counter is asynchronously reset, output Low, when power is applied or when global reset or master reset is active. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
R	**L**	**CE**	**C**	**D7 – D0**	**Q7 – Q0**	**TC**	**CEO**
1	X	X	↑	X	0	0	0
0	1	X	↑	D	d7 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

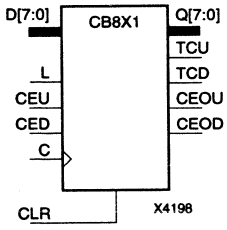
TC = Q7•Q6•...•Q0

CEO = TC•CE

dn = state of referenced input one set-up time prior to active clock transition

CB8X1

8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB8X1 is an 8-stage, 8-bit, synchronous, loadable, clearable, bidirectional binary counter. CB8X1 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the EPLD architecture.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored, data outputs (Q7 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, independent of clock transitions. The data on the D7 – D0 inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The outputs (Q7 – Q0) increment when CEU is High, provided CLR and L are Low, during the Low-to-High clock transition. The outputs (Q7 – Q0) decrement when CED is High, provided CLR and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The C, L, and CLR inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED. When cascading counters, the final terminal count signals can be

produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component, resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each output remain on-chip. Otherwise, a macrocell buffer delay is introduced.

The counter is initialized to zero (TCU Low and TCD High) when the device is powered-up or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
CLR	**L**	**CEU**	**CED**	**C**	**D7 – D0**	**Q7 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	X	X	0	0	1	0	CEOD
0	1	X	X	↑	D	d7 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	CEOU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	CEOD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q7•Q6•Q5•...Q0

TCD = $\overline{Q7}$•$\overline{Q6}$•$\overline{Q5}$•...$\overline{Q0}$

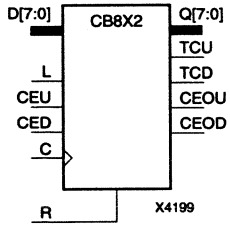
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CB8X2

8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB8X2 is an 8-stage, 8-bit, synchronous, loadable, resettable, bidirectional binary counter. CB8X2 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the EPLD architecture.

The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored, data outputs (Q7 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, on the Low-to-High clock (C) transition. The data on the D7 – D0 inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The outputs (Q7 – Q0) increment when CEU is High, provided R and L are Low, during the Low-to-High clock transition. The outputs (Q7 – Q0) decrement when CED is High, provided R and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The C, L, and R inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND-gates within the component, resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each output remain on-chip. Otherwise, a macrocell buffer delay is introduced.

The counter is initialized to zero (TCU Low and TCD High) when the device is powered-up or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
R	**L**	**CEU**	**CED**	**C**	**D7 – D0**	**Q7 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	↑	X	0	0	1	0	CEOD
0	1	X	X	↑	D	d7 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	CEOU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	CEOD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q7•Q6•Q5•...Q0

TCD = $\overline{Q7}$•$\overline{Q6}$•$\overline{Q5}$•...$\overline{Q0}$

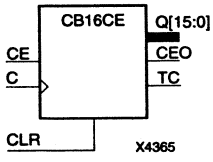
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CB16CE

16-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB16CE is a 16-stage, 16-bit, synchronous, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q15 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC is High and CE is High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

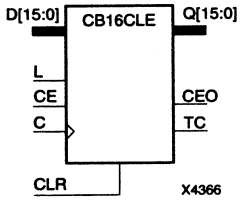
Inputs			Outputs		
CLR	**CE**	**C**	**Q15 – Q0**	**TC**	**CEO**
1	X	X	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q15•Q14•Q13•Q12...•Q0)

CEO = (TC•CE)

CB16CLE

16-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB16CLE is a 16-stage, 16-bit, synchronous, loadable, clearable, cascadable binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D15 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q15 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
CLR	**L**	**CE**	**C**	**D15 – D0**	**Q15 – Q0**	**TC**	**CEO**
1	X	X	X	X	0	0	0
0	1	X	↑	D	d15 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

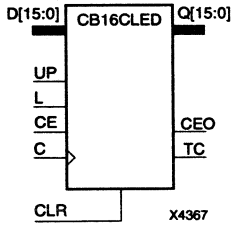
TC = (Q15•Q14•Q13•Q12...•Q0)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

CB16CLED

16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB16CLED is a 16-stage, 16-bit, synchronous, loadable, clearable, cascadable, bidirectional binary counter. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D15 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q15 – Q0) decrement when CE is High and UP is Low during the Low-to-High clock transition. The outputs (Q15 – Q0) increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the count enable out (CEO) output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage. For EPLD designs, refer to "CB16X1" for high-performance cascadable, bidirectional counters.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs						Outputs		
CLR	**L**	**CE**	**C**	**UP**	**D15 – D0**	**Q15 – Q0**	**TC**	**CEO**
1	X	X	X	X	X	0	0	0
0	1	X	↑	X	D	d15 – d0	TC	CEO
0	0	0	X	X	X	No Chg	No Chg	0
0	0	1	↑	1	X	Inc	TC	CEO
0	0	1	↑	0	X	Dec	TC	CEO

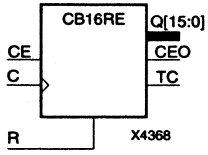
TC = (Q15•Q14•Q13...•Q0•UP) + ($\overline{Q15}$•$\overline{Q14}$•$\overline{Q13}$...•$\overline{Q0}$•$\overline{UP}$)

CEO = (TC•CE)

dn = state of referenced clock one set-up time prior to active clock transition

CB16RE

16-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CB16RE is a 16-stage, 16-bit, synchronous, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock (C) transition. The outputs (Q15 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

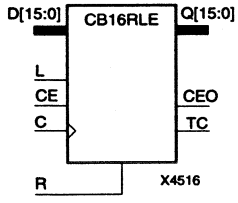
Inputs			Outputs		
R	**CE**	**C**	**Q15 – Q0**	**TC**	**CEO**
1	X	↑	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q15•Q14•Q13...•Q0)

CEO = (TC•CE)

CB16RLE

16-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB16RLE is a 16-stage, 16-bit, synchronous, loadable, resettable, cascadable binary counter. The synchronous reset (R) is the highest priority input.

The synchronous R, when High, overrides all other inputs and resets the Q15 – Q0, TC, and CEO outputs to Low on the Low-to-High clock (C) transition. The data on the D15 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of CE. The outputs (Q15 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low.

The TC output is High when all Q outputs are High. The CEO output is High when all Q outputs and CE are High, to allow direct cascading of counters. Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C, L, and R inputs in parallel. The maximum length of the counter is determined by the accumulated CE-to-CEO propagation delays versus the clock period.

The counter is asynchronously reset, output Low, when power is applied or when global reset or master reset is active. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs		
R	**L**	**CE**	**C**	**D15 – D0**	**Q15 – Q0**	**TC**	**CEO**
1	X	X	↑	X	0	0	0
0	1	X	↑	D	d15 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

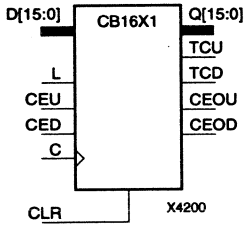
TC = Q15•Q14•...•Q0

CEO = TC•CE

dn = state of referenced input one set-up time prior to clock transition

CB16X1

16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB16X1 is a 16-stage, 16-bit, synchronous, loadable, clearable, bidirectional binary counter. CB16X1 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the EPLD architecture.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored, data outputs (Q15 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, independent of clock transitions. The data on the D15 – D0 inputs loads into the counter on the Low-to-High clock (C) transition, when the load enable input (L) is High, independent of the CE inputs.

The outputs (Q15 – Q0) increment when CEU is High, provided CLR and L are Low, during the Low-to-High clock transition. The outputs (Q15 – Q0) decrement when CED is High, provided CLR and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The clock, L, and CLR inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component, resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each output remain on-chip. Otherwise, a macrocell buffer delay is introduced. The counter is initialized to zero (TCU Low and TCD High) when the device is powered-up or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
CLR	**L**	**CEU**	**CED**	**C**	**D15 – D0**	**Q15 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	X	X	0	0	1	0	CEOD
0	1	X	X	↑	D	d15 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	CEOU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	CEOD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q15•Q14•Q13•...•Q0

TCD = $\overline{Q15}$•$\overline{Q14}$•$\overline{Q13}$•...•$\overline{Q0}$

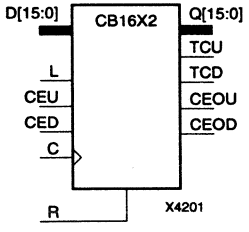
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CB16X2

16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

CB16X2 is a 16-stage, 16-bit, synchronous, loadable, resettable, bidirectional binary counter. CB16X2 has separate count-enable inputs and synchronous terminal-count outputs for up and down directions, to support high-speed cascading in the EPLD architecture.

The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored, data outputs (Q15 – Q0) go to logic level zero, and terminal count outputs TCU and TCD go to zero and one, respectively, on the Low-to-High clock (C) transition. The data on the D15 – D0 inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs. The outputs (Q15 – Q0) increment when CEU is High, provided R and L are Low, during the Low-to-High clock transition. The outputs (Q15 – Q0) decrement when CED is High, provided R and L are Low.

The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High. For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The C, L, and R inputs are connected in parallel.

In Xilinx EPLD devices, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component, resulting in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced. The counter is initialized to zero (TCU Low and TCD High) when the device is powered-up or when the device Master Reset pin is activated. The clock (C) input can be driven by either the EPLD FastCLK global net (represented by a BUFG symbol), an ordinary input, or other on-chip logic.

Inputs						Outputs				
R	**L**	**CEU**	**CED**	**C**	**D15 – D0**	**Q15 – Q0**	**TCU**	**TCD**	**CEOU**	**CEOD**
1	X	X	X	↑	X	0	0	1	0	CEOD
0	1	X	X	↑	D	d15 – d0	TCU	TCD	CEOU	CEOD
0	0	0	0	X	X	No Chg	No Chg	No Chg	0	0
0	0	1	0	↑	X	Inc	TCU	TCD	CEOU	0
0	0	0	1	↑	X	Dec	TCU	TCD	0	CEOD
0	0	1	1	↑	X	Inc	TCU	TCD	Invalid	Invalid

TCU = Q15•Q14•Q13•...•Q0

TCD = $\overline{Q15}$•$\overline{Q14}$•$\overline{Q13}$•...•$\overline{Q0}$

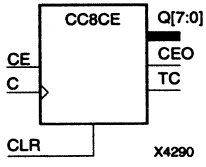
CEOU = TCU•CEU

CEOD = TCD•CED

dn = state of referenced input one set-up time prior to active clock transition

CC8CE

8-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC8CE is an 8-stage, 8-bit, synchronous, clearable, cascadable binary counter. The counter is implemented using carry logic with relative location restraints, which assures most efficient logic placement. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q7 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

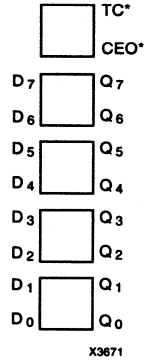
The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs		
CLR	**CE**	**C**	**Q7 – Q0**	**TC**	**CEO**
1	X	X	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q7•Q6•Q5•Q4•...•Q0)

CEO = (TC•CE)

XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.

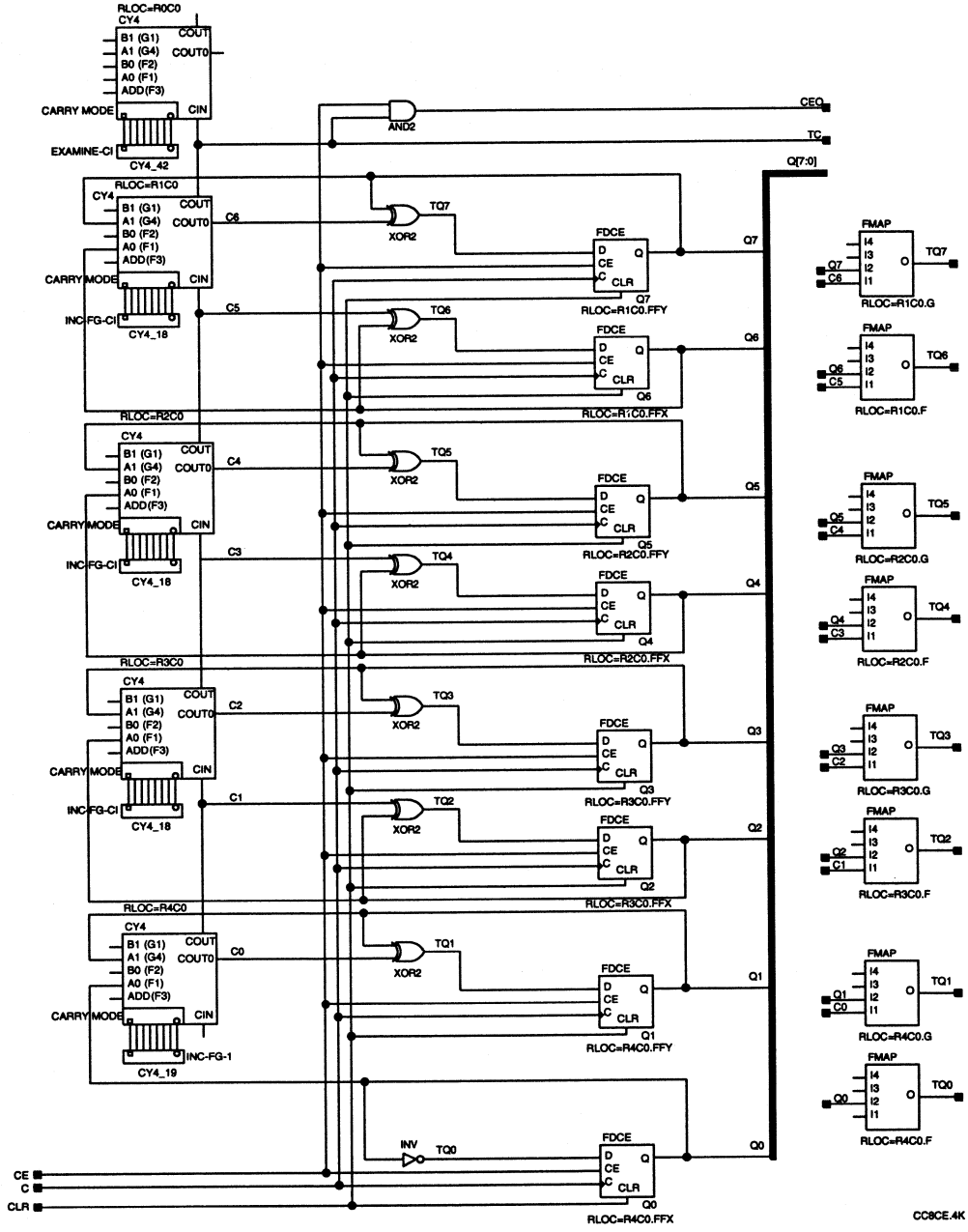
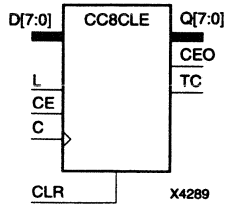


Figure 3-37 CC8CE XC4000 Implementation

CC8CLE

8-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC8CLE is an 8-stage, 8-bit, synchronous, loadable, clearable, cascadable binary counter. The counter is implemented using carry logic with relative location constraints, which assures most efficient logic placement.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D7 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q7 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

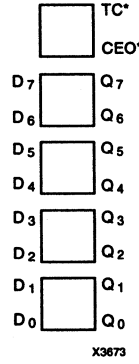
Inputs					Outputs		
CLR	L	CE	C	D7 – D0	Q7 – Q0	TC	CEO
1	X	X	X	X	0	0	0
0	1	X	X	D	d7 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

TC = (Q7•Q6•Q5•Q4•...•Q0)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.

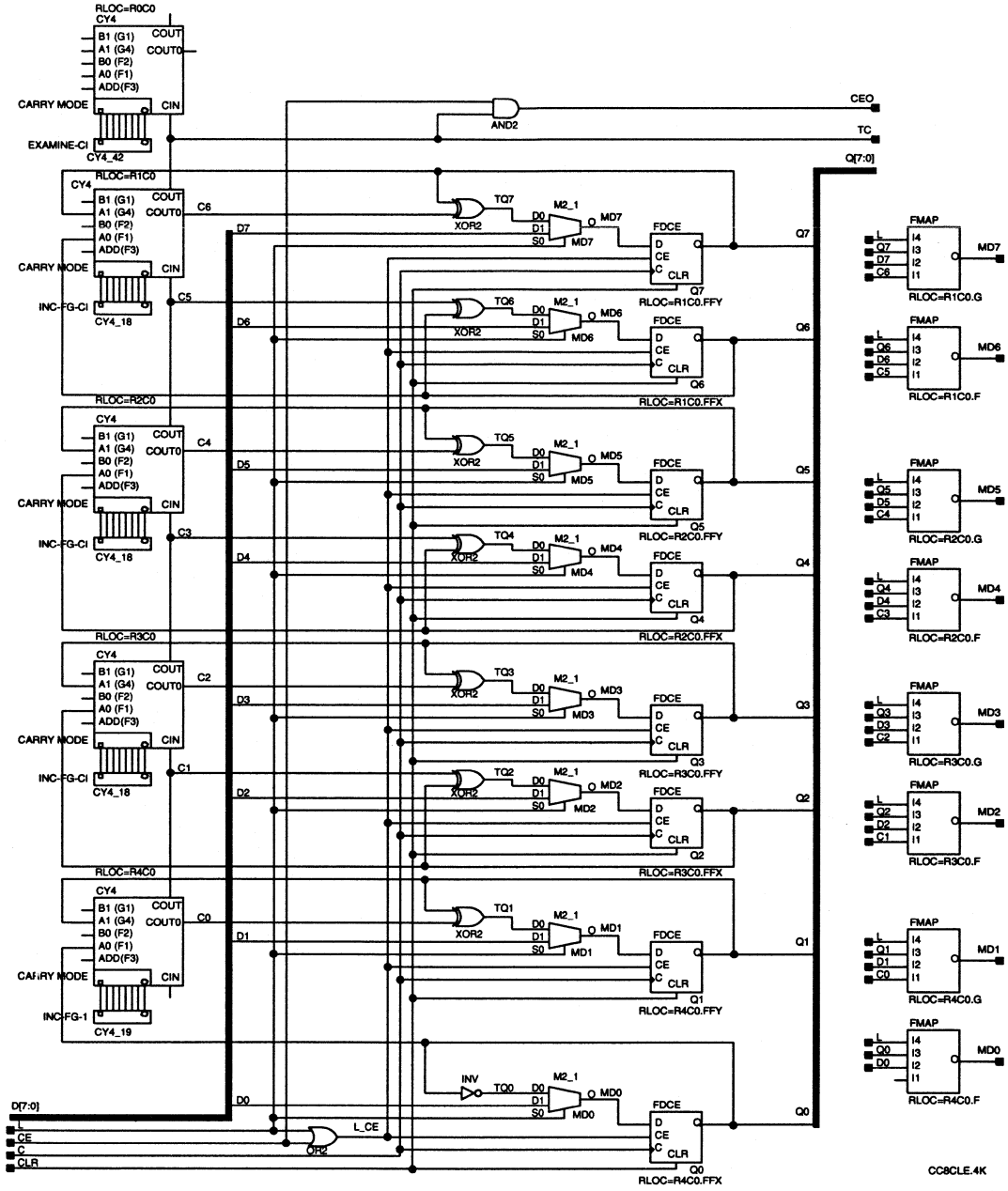
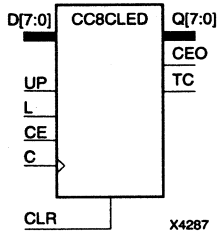


Figure 3-38 CC8CLE XC4000 Implementation

CC8CLED

8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC8CLED is an 8-stage, 8-bit, synchronous, loadable, clearable, cascadable, bidirectional binary counter. The counter is implemented using carry logic with relative location constraints, which assures most efficient logic placement.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D7 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q7 – Q0) decrement when CE is High and UP is Low during the Low-to-High clock transition. The outputs (Q7 – Q0) increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the count enable out (CEO) output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

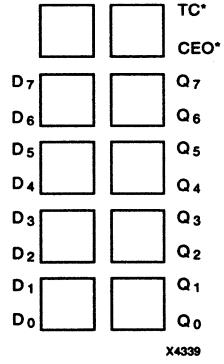
Inputs						Outputs		
CLR	**L**	**CE**	**C**	**UP**	**D7 – D0**	**Q7 – Q0**	**TC**	**CEO**
1	X	X	X	X	X	0	0	0
0	1	X	X	X	D	d7 – d0	TC	CEO
0	0	0	X	X	X	No Chg	No Chg	0
0	0	1	↑	1	X	Inc	TC	CEO
0	0	1	↑	0	X	Dec	TC	CEO

$TC = (Q7 \bullet Q6 \bullet Q5 \bullet ... \bullet Q0 \bullet UP) + (\overline{Q7} \bullet \overline{Q6} \bullet \overline{Q5} \bullet ... \bullet \overline{Q0} \bullet \overline{UP})$

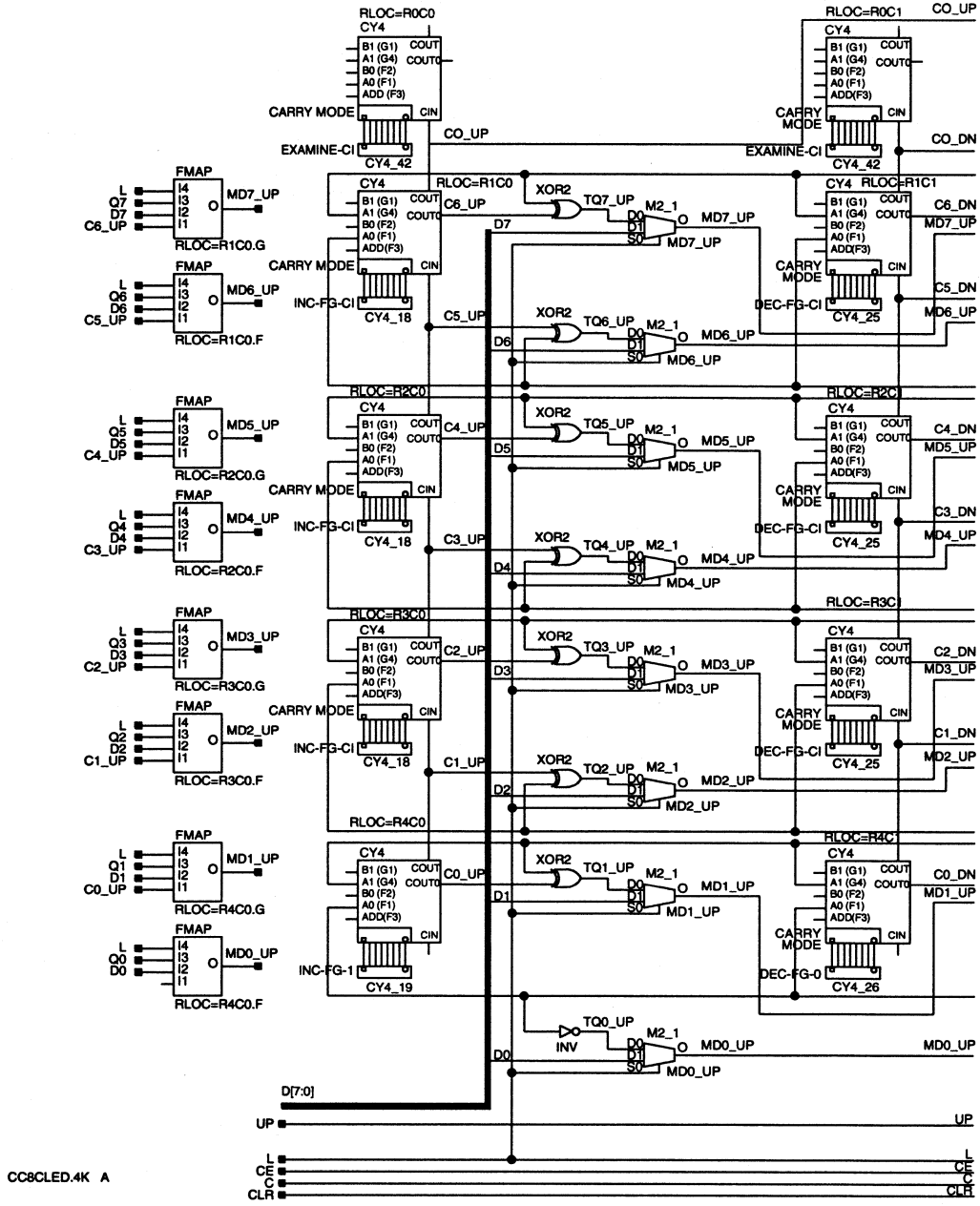
$CEO = (TC \bullet CE)$

dn = state of referenced clock one set-up time prior to active clock transition

XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.


RLOC=R0C0
RLOC=R0C1
CO_UP
CY4
CARRY MODE
EXAMINE-CI
CY4_42
CO_DN
RLOC=R1C0
RLOC=R1C1
XOR2
TQ7_UP
M2_1
MD7_UP
C6_UP
C6_DN
D7
CY4_18
CY4_25
C5_UP
C5_DN
TQ6_UP
MD6_UP
D6
RLOC=R2C0
RLOC=R2C1
TQ5_UP
C4_UP
C4_DN
MD5_UP
D5
TQ4_UP
C3_UP
C3_DN
MD4_UP
D4
RLOC=R3C0
RLOC=R3C1
TQ3_UP
C2_UP
C2_DN
MD3_UP
D3
TQ2_UP
C1_UP
C1_DN
MD2_UP
D2
RLOC=R4C0
RLOC=R4C1
TQ1_UP
C0_UP
C0_DN
MD1_UP
D1
CY4_19
CY4_26
TQ0_UP
INV
MD0_UP
D0
D[7:0]
UP
L
CE
C
CLR
FMAP
MD7_UP
MD6_UP
MD5_UP
MD4_UP
MD3_UP
MD2_UP
MD1_UP
MD0_UP
RLOC=R1C0.G
RLOC=R1C0.F
RLOC=R2C0.G
RLOC=R2C0.F
RLOC=R3C0.G
RLOC=R3C0.F
RLOC=R4C0.G
RLOC=R4C0.F
CC8CLED.4K A

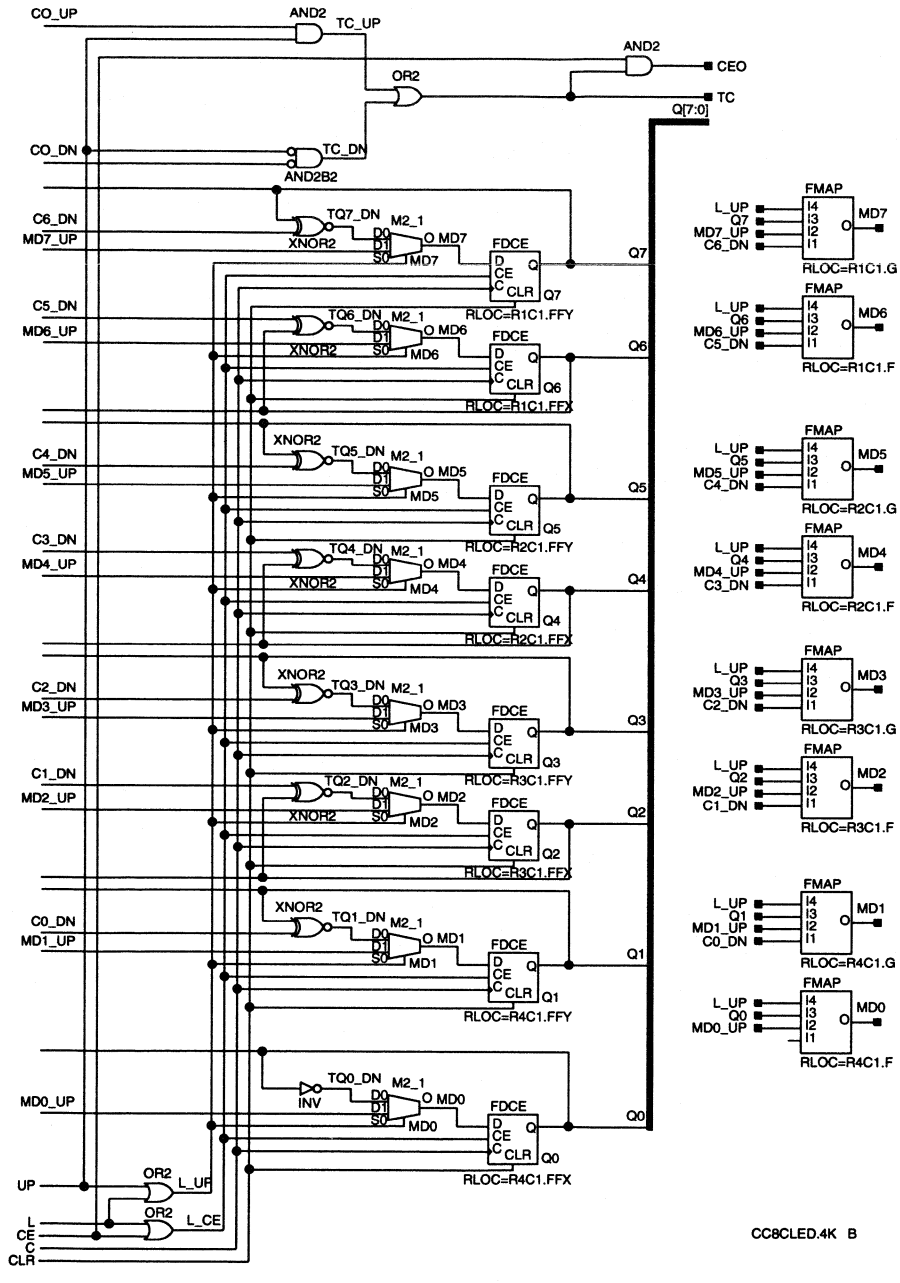
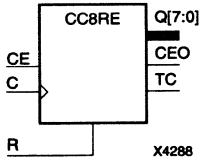


Figure 3-39 CC8CLED XC4000 Implementation

CC8RE

8-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC8RE is an 8-stage, 8-bit, synchronous, resettable, cascadable binary counter. The counter is implemented using carry logic with relative location constraints, which assures most efficient logic placement. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored and data (Q7 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock (C) transition. The outputs (Q7 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs and CE are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

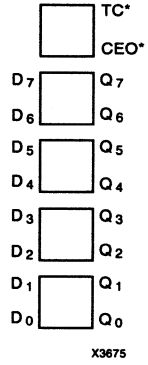
The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs		
R	**CE**	**C**	**Q7 – Q0**	**TC**	**CEO**
1	X	↑	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q7•Q6•Q5•...•Q0•CE)

CEO = (TC•CE)

XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.

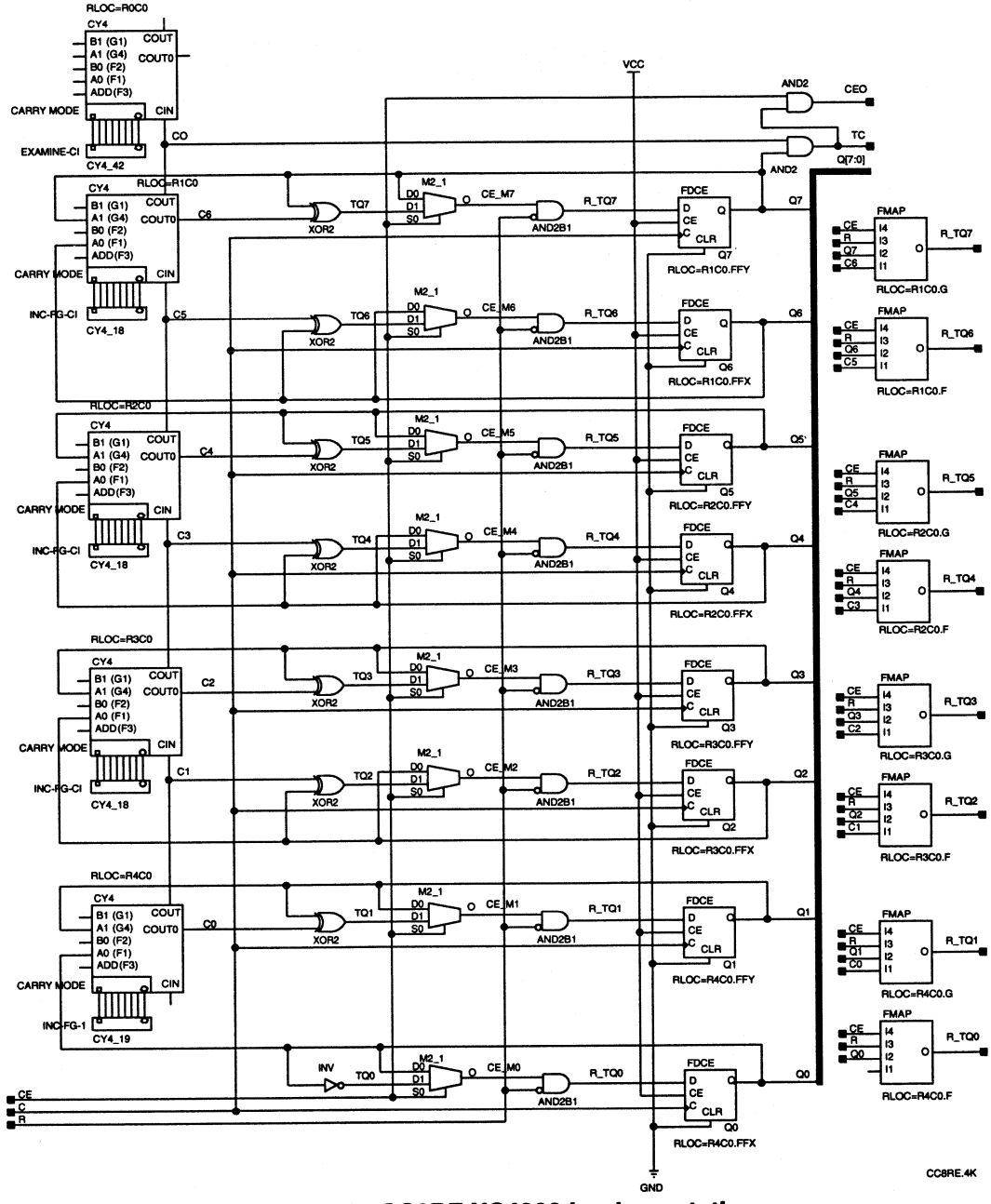
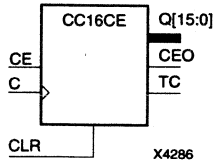


Figure 3-40 CC8RE XC4000 Implementation

CC16CE

16-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC16CE is a 16-stage, 16-bit, synchronous, clearable, cascadable binary counter. The counter is implemented using carry logic with relative location restraints, which assures the most efficient logic placement. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q15 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

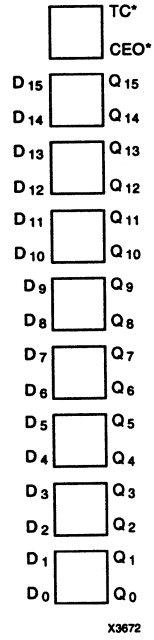
The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs		
CLR	CE	C	Q15 – Q0	TC	CEO
1	X	X	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q15•Q14•Q13•Q12•...•Q0)

CEO = (TC•CE)

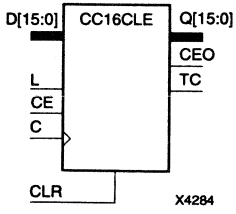
XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.

CC16CLE

16-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC16CLE is a 16-stage, 16-bit, synchronous, loadable, clearable, cascadable binary counter. The counter is implemented using carry logic with relative location constraints, which assures the most efficient logic placement.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D15 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q15 – Q0) increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

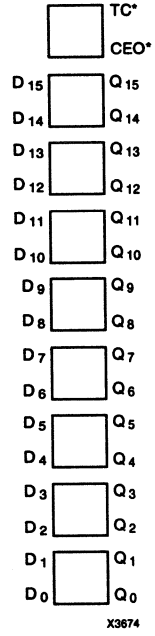
Inputs					Outputs		
CLR	L	CE	C	D15 – D0	Q15 – Q0	TC	CEO
1	X	X	X	X	0	0	0
0	1	X	X	D	d15 – d0	TC	CEO
0	0	0	X	X	No Chg	No Chg	0
0	0	1	↑	X	Inc	TC	CEO

TC = (Q15•Q14•Q13•Q12•...•Q0)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

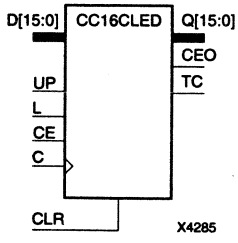
XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.

CC16CLED

16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC16CLED is a 16-stage, 16-bit, synchronous, loadable, clearable, cascadable, bidirectional binary counter. The counter is implemented using carry logic with relative location constraints, which assures most efficient logic placement.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D15 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The outputs (Q15 – Q0) decrement when CE is High and UP is Low during the Low-to-High clock transition. The outputs (Q15 – Q0) increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the count enable out (CEO) output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

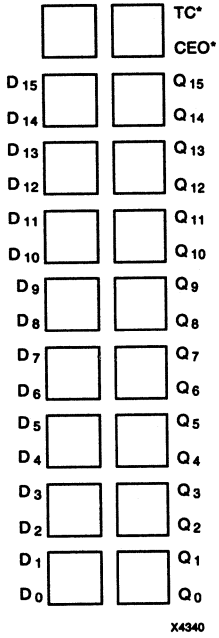
Inputs						Outputs		
CLR	L	CE	C	UP	D15 – D0	Q15 – Q0	TC	CEO
1	X	X	X	X	X	0	0	0
0	1	X	X	X	D	d15 – d0	TC	CEO
0	0	0	X	X	X	No Chg	No Chg	0
0	0	1	↑	1	X	Inc	TC	CEO
0	0	1	↑	0	X	Dec	TC	CEO

TC = (Q15•Q14•Q13•...•Q0•UP) + ($\overline{Q15}$•$\overline{Q14}$•$\overline{Q13}$•...•$\overline{Q0}$•$\overline{UP}$)

CEO = (TC•CE)

dn = state of referenced clock one set-up time prior to active clock transition

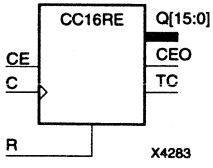
XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.

CC16RE

16-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

CC16RE is a 16-stage, 16-bit, synchronous, resettable, cascadable binary counter. The counter is implemented using carry logic with relative location constraints, which assures most efficient logic placement. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored and data (Q15 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock (C) transition. The outputs (Q15 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs and CE are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

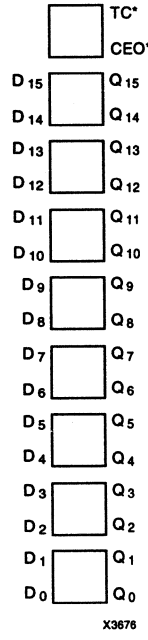
The counter is asynchronously reset, output Low, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs		
R	CE	C	Q15 – Q0	TC	CEO
1	X	↑	0	0	0
0	0	X	No Chg	No Chg	0
0	1	↑	Inc	TC	CEO

TC = (Q15•Q14•Q13•...•Q0)

CEO = (TC•CE)

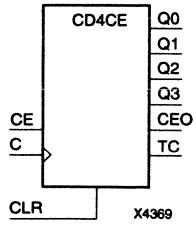
XC4000 Topology



In the process of combining the logic that loads CEO and TC, the place and route software might map the logic that generates CEO and TC to different function generators. If this mapping occurs, the CEO and TC logic cannot be placed in the uppermost CLB as indicated in the illustration.

CD4CE

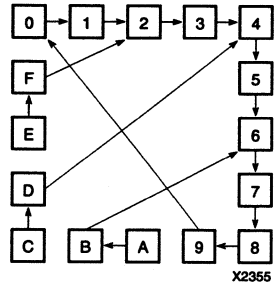
4-Bit Cascadable BCD Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CD4CE is a 4-stage, 4-bit, synchronous, clearable, cascadable binary-coded-decimal (BCD) counter. The asynchronous clear input (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and data (Q3 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The outputs (Q3 – Q0) increment when clock enable (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles for XC2000, XC3000, and XC4000 architectures, as shown in the following state diagram. For XC7000, the counter resets to zero or recovers within the first clock cycle.



Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the CLR and clock inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC), where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs					
CLR	CE	C	Q3	Q2	Q1	Q0	TC	CEO
1	X	X	0	0	0	0	0	0
0	1	↑	--------Increment--------				TC	CEO
0	0	X	--------No Change--------				TC	0
0	1	X	1	0	0	1	1	1

TC = (Q3•$\overline{Q2}$•$\overline{Q1}$•Q0)

CEO = (TC•CE)

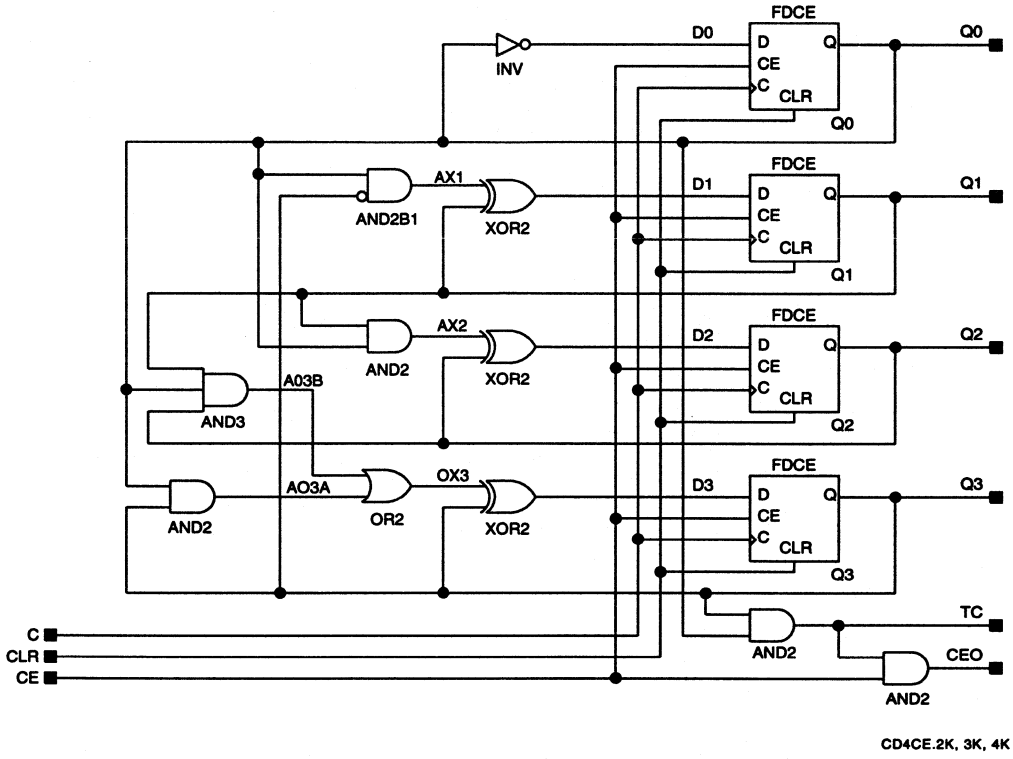
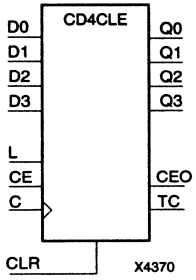


Figure 3-41 CD4CE XC2000/3000/4000 Implementation

CD4CLE

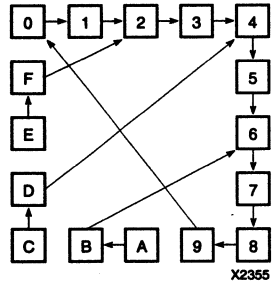
4-Bit Loadable Cascadable BCD Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CD4CLE is a 4-stage, 4-bit, synchronous, loadable, clearable, binary-coded-decimal (BCD) counter. The asynchronous clear input (CLR) is the highest priority input. When CLR is High, all other inputs are ignored and the data (Q3 – Q0) and terminal count (TC) outputs go to logic level zero, independent of clock transitions. The data on the D3 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition. The outputs (Q3 – Q0) increment when clock enable input (CE) is High during the Low- to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles for XC2000, XC3000, and XC4000, as shown in the following state diagram. For XC7000, the counter resets to zero or recovers within the first clock cycle.



Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the CLR, L, and C inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC),

where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs					
CLR	**L**	**CE**	**D3 – D0**	**C**	**Q3**	**Q2**	**Q1**	**Q0**	**TC**	**CEO**
1	X	X	X	X	0	0	0	0	0	0
0	1	X	D3 – D0	↑	d3	d2	d1	d0	TC	CEO
0	0	1	X	↑	---------Increment----------				TC	CEO
0	0	0	X	X	--------No Change--------				TC	0
0	0	1	X	X	1	0	0	1	1	1

TC = ($Q3 \bullet \overline{Q2} \bullet \overline{Q1} \bullet Q0$)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

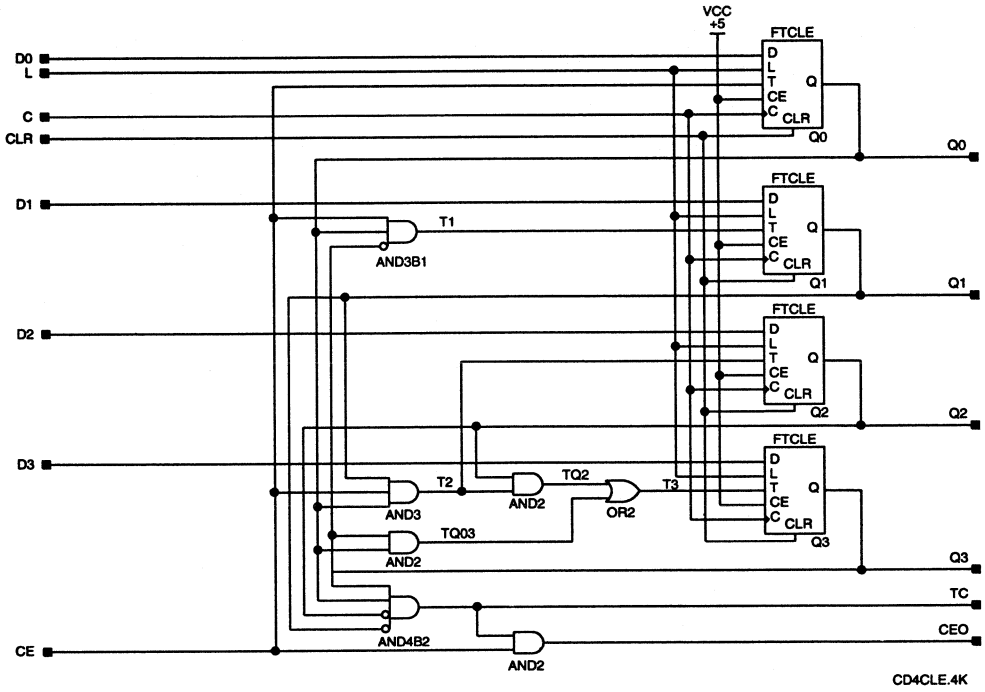
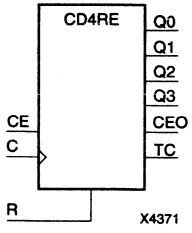


Figure 3-42 CD4CLE XC2000/3000/4000 Implementation

CD4RE

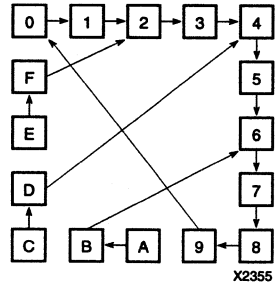
4-Bit Cascadable BCD Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CD4RE is a 4-stage, 4-bit, synchronous, resettable, cascadable binary-coded-decimal (BCD) counter. The synchronous reset input (R) is the highest priority input. When R is High, all other inputs are ignored and (Q3 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock (C) transition. The outputs (Q3 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles for XC2000, XC3000, and XC4000, as shown in the following state diagram. For XC7000, the counter resets to zero or recovers within the first clock cycle.



Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the R and clock inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC) where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs			Outputs					
R	CE	C	Q3	Q2	Q1	Q0	TC	CEO
1	X	↑	0	0	0	0	0	0
0	1	↑	---------Increment--------				TC	CEO
0	0	X	--------No Change--------				TC	0
0	1	X	1	0	0	1	1	1

TC = (Q3•$\overline{Q2}$•$\overline{Q1}$•Q0)

CEO = (TC•CE)

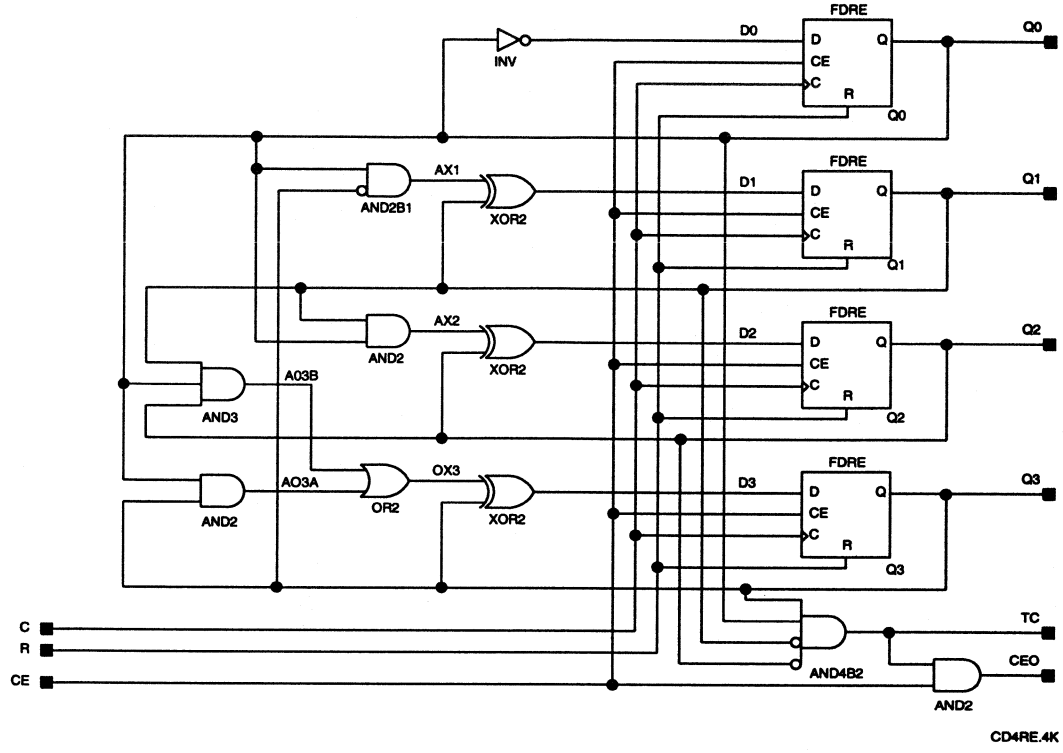
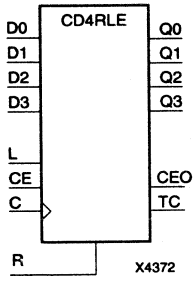


Figure 3-43 CD4RE XC2000/3000/4000 Implementation

CD4RLE

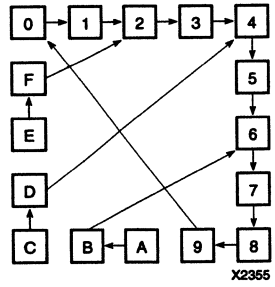
4-Bit Loadable Cascadable BCD Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CD4RLE is a 4-stage, 4-bit, synchronous, loadable, resettable, binary-coded-decimal (BCD) counter. The synchronous reset input (R) is the highest priority input. When R is High, all other inputs are ignored and the data (Q3 – Q0) and terminal count (TC) outputs go to logic level zero on the Low-to-High clock transitions. The data on the D3 – D0 inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition. The outputs (Q3 – Q0) increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles forXC2000, XC3000, and XC4000, as shown in the following state diagram. For XC7000, the counter resets to zero or recovers within the first clock cycle.



Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the R, L, and C inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than n(tCE-TC),

where "n" is the number of stages and "tCE-TC" is the CE-to-TC propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

Inputs					Outputs					
R	**L**	**CE**	**D3 – D0**	**C**	**Q3**	**Q2**	**Q1**	**Q0**	**TC**	**CEO**
1	X	X	X	↑	0	0	0	0	0	0
0	1	X	D3 – D0	↑	d3	d2	d1	d0	TC	CEO
0	0	1	X	↑	---------Increment---------				TC	CEO
0	0	0	X	X	--------No Change---------				TC	0
0	0	1	X	X	1	0	0	1	1	1

TC = ($Q3 \bullet \overline{Q2} \bullet \overline{Q1} \bullet Q0$)

CEO = (TC•CE)

dn = state of referenced input one set-up time prior to active clock transition

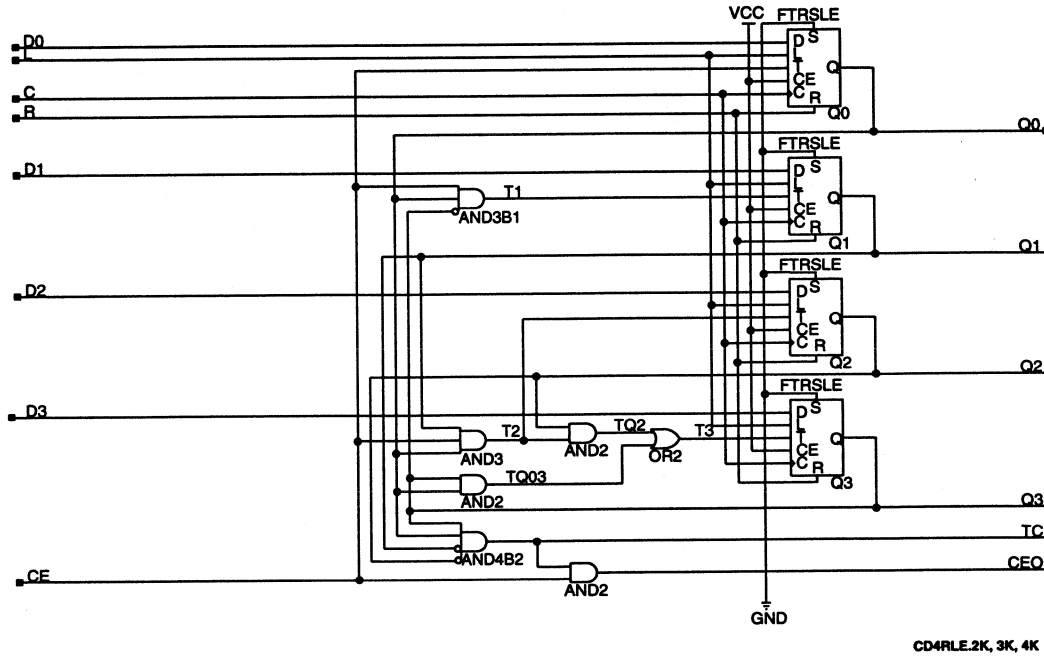
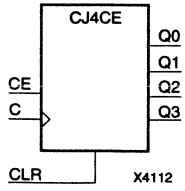


Figure 3-44 CD4RLE XC2000/3000/4000 Implementation

CJ4CE

4-Bit Johnson Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CJ4CE is a clearable Johnson/shift counter. The asynchronous clear (CLR) input, when High, overrides all other inputs and causes the data outputs (Q3 – Q0) to go to logic level zero, independent of clock (C) transitions. The counter increments (shifts Q0 to Q1, Q1 to Q2,and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low. The Q3 output is inverted and fed back to input Q0 to provide continuous counting operation.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs			
CLR	CE	C	Q0	Q1	Q2	Q3
1	X	X	0	0	0	0
0	0	X	--------No Change--------			
0	1	↑	$\overline{q3}$	q0	q1	q2

qn = state of referenced output one set-up time prior to active clock transition

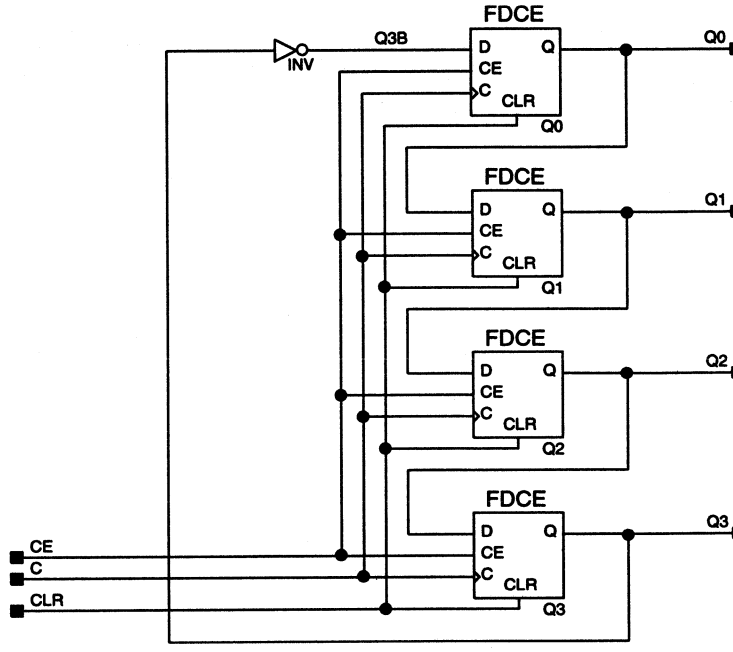
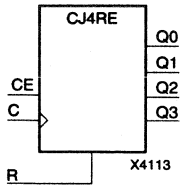


Figure 3-45 CJ4CE XC2000/3000/4000 Implementation

CJ4RE

4-Bit Johnson Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CJ4RE is a resettable Johnson/shift counter. The synchronous reset (R) input, when High, overrides all other inputs causes the data outputs (Q3 – Q0) to go to logic level zero during the Low-to-High clock (C) transition. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low. The Q3 output is inverted and fed back to input Q0 to provide continuous counting operation.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs			
R	CE	C	Q0	Q1	Q2	Q3
1	X	↑	0	0	0	0
0	0	X	--------No Change--------			
0	1	↑	$\overline{q3}$	q0	q1	q2

qn = state of referenced output one set-up time prior to active clock transition

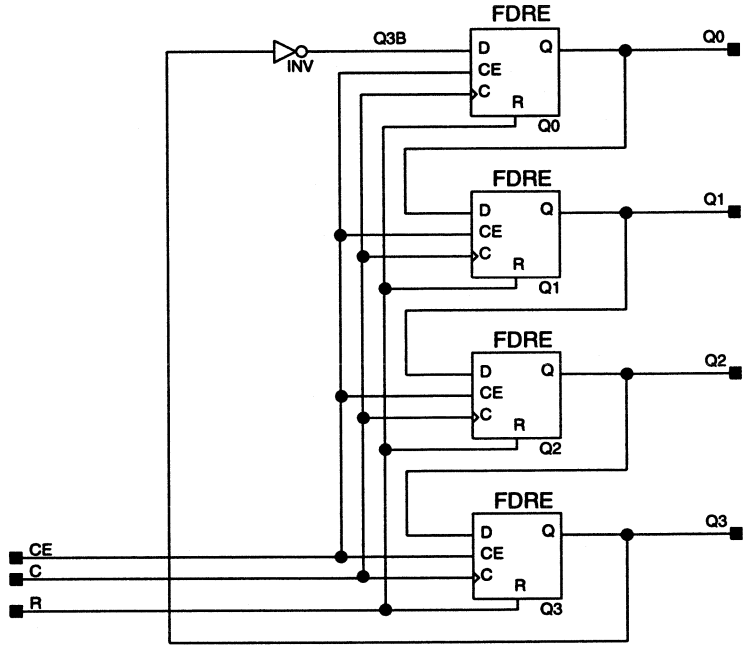
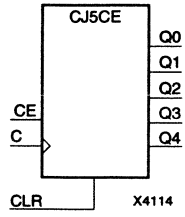


Figure 3-46 CJ4RE XC2000/3000/4000 Implementation

CJ5CE

5-Bit Johnson Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CJ5CE is a clearable Johnson/shift counter. The asynchronous clear (CLR) input, when High, overrides all other inputs and causes the data outputs (Q4 – Q0) to go to logic level zero, independent of clock (C) transitions. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low. The Q4 output is inverted and fed back to input Q0 to provide continuous counting operation.

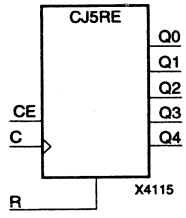
The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs				
CLR	CE	C	Q0	Q1	Q2	Q3	Q4
1	X	X	0	0	0	0	0
0	0	X	----------No Change----------				
0	1	↑	$\overline{q4}$	q0	q1	q2	q3

qn = state of referenced output one set-up time prior to active clock transition

CJ5RE

5-Bit Johnson Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CJ5RE is a resettable Johnson/shift counter. The synchronous reset (R) input, when High, overrides all other inputs and causes the data outputs (Q4 – Q0) to go to logic zero during the Low-to-High clock transition. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low. The Q4 output is inverted and fed back to input Q0 to provide continuous counting operation.

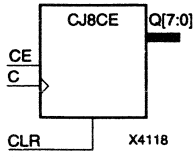
The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs				
R	CE	C	Q0	Q1	Q2	Q3	Q4
1	X	↑	0	0	0	0	0
0	0	X	----------No Change----------				
0	1	↑	$\overline{q4}$	q0	q1	q2	q3

qn = state of referenced output one set-up time prior to active clock transition

CJ8CE

8-Bit Johnson Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CJ8CE is a clearable Johnson/shift counter. The asynchronous clear (CLR) input, when High, overrides all other inputs and causes the data outputs (Q7 – Q0) to go to logic level zero, independent of clock (C) transitions. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low. The Q7 output is inverted and fed back to input Q0 to provide continuous counting operation.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs	
CLR	CE	C	Q0	Q1 – Q7
1	X	X	0	0
0	0	X	--No Change--	
0	1	↑	$\overline{q7}$	q0 – q6

qn = state of referenced output one set-up time prior to active clock transition

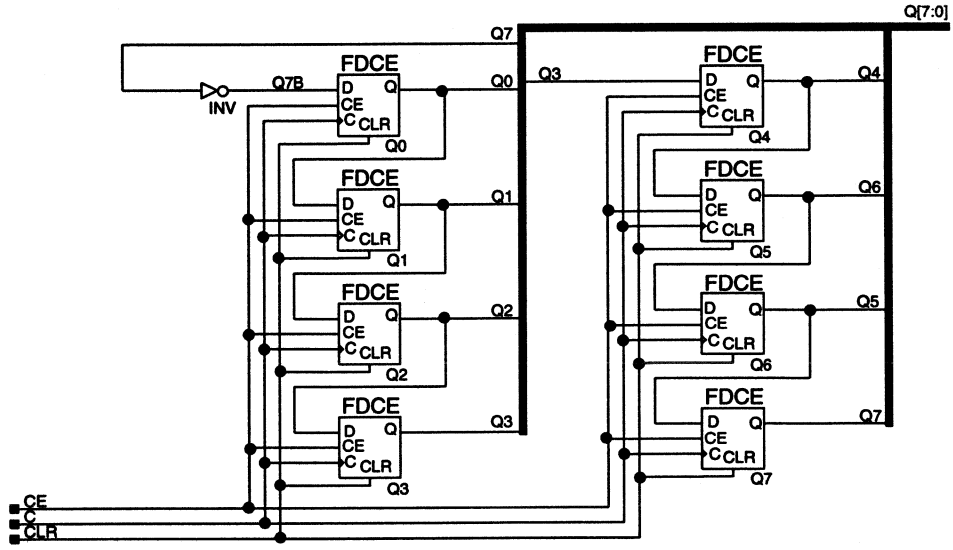
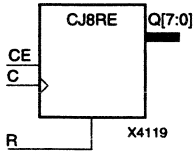


Figure 3-47 CJ8CE XC2000/3000/4000 Implementation

CJ8RE

8-Bit Johnson Counter with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CJ8RE is a resettable Johnson/shift counter. The synchronous reset (R) input, when High, overrides all other inputs and causes the data outputs (Q7 – Q0) to go to logic level zero during the Low-to-High clock transition. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low. The Q7 output is inverted and fed back to input Q0 to provide continuous counting operation.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs	
R	CE	C	Q0	Q1 – Q7
1	X	↑	0	0
0	0	X	--No Change--	
0	1	↑	$\overline{q7}$	q0 – q6

qn = state of referenced output one set-up time prior to active clock transition

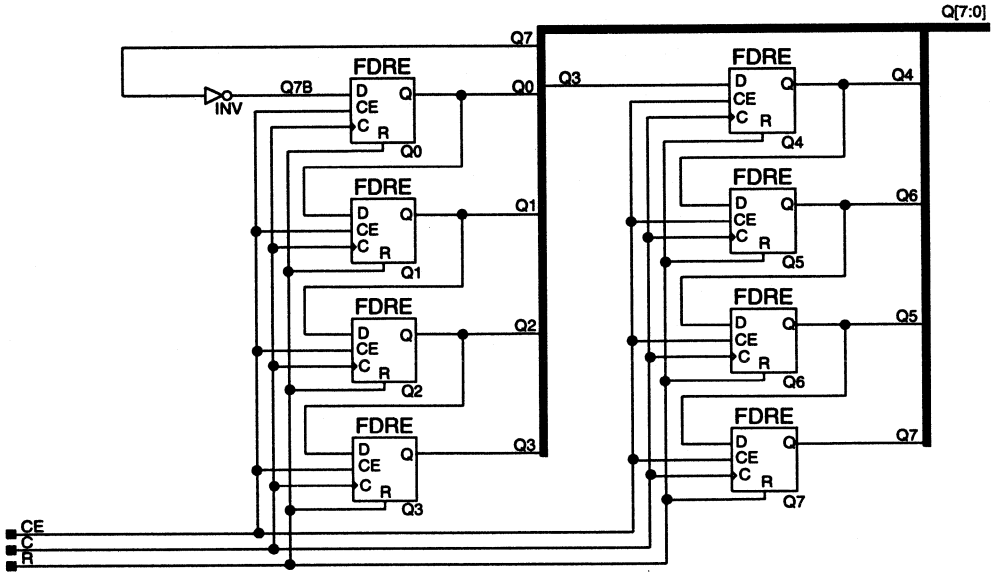


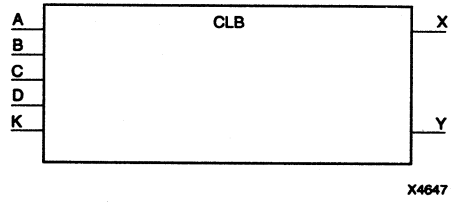
Figure 3-48 CJ8RE XC2000/3000/4000 Implementation

CLB

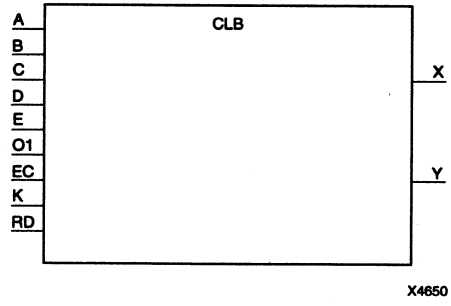
CLB Configuration Symbol

XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	N/A	N/A

The CLB symbol enables you to manually specify a CLB configuration. It allows you to enter portions of a logic design directly in terms of the physical CLB, rather than schematically. Using the CLB symbol provides precise partitioning control and requires knowledge of the CLB architecture. Use it in place of the equivalent captured logic and not in conjunction with it.



XC2000



XC3000

A blank XC2000 CLB primitive symbol and its corresponding configured CLB primitive and circuit are shown in the following figure.

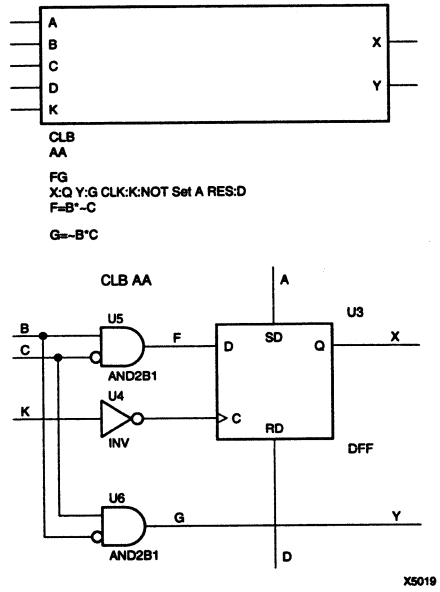


Figure 3-49 XC2000 CLB Primitive Example and Equivalent Circuit

A blank XC3000 CLB primitive symbol and its corresponding configured CLB primitive and circuit are shown in the following figure.

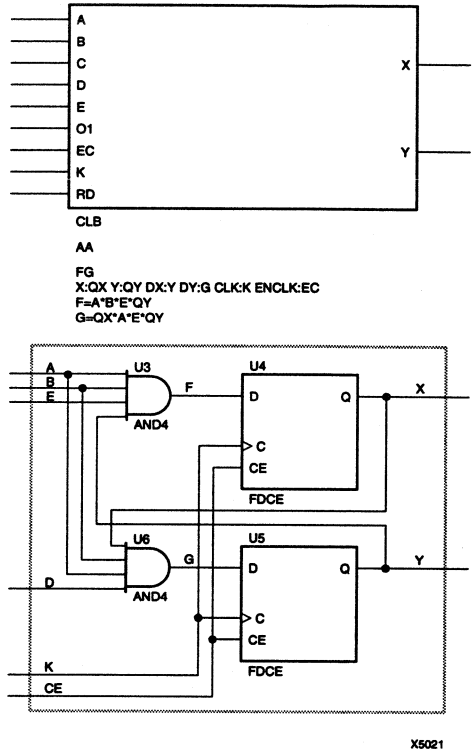


Figure 3-50 XC3000 CLB Primitive Example and Equivalent Circuit

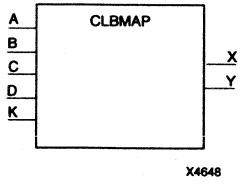
CLB symbol pins correspond to actual CLB pins. Signals connected to these pins in a schematic are connected to the corresponding CLB pins in the design. You must specify the BASE, CONFIG, and EQUATE commands for the CLB. These commands are entered on the schematic and the translator puts them into the CFG records in the LCA Xilinx netlist file. It is not necessary for the translator program to parse the commands specifying the CLB configuration. The mapping program from the LCA Xilinx netlist to the LCA design checks these commands for errors.

The configuration commands must be consistent with the connections. For example, if you use the A input in an equation, connect a signal to the A pin. Refer to the applicable CAE tool interface user guide for more information on specifying the CLB configuration commands in the schematic.

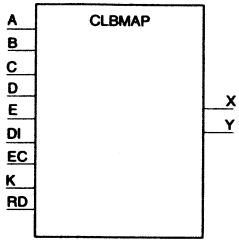
You can specify the location of a CLB on the device using the LOC attribute. When specifying the LOC attribute, a valid CLB name (AA, AB, and so forth) must be used. Refer to the "Attributes, Constraints, and Carry Logic" chapter for more information on the LOC attribute.

CLBMAP

Logic-Partitioning Control Symbol



XC2000



XC3000

XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	N/A	N/A

The CLBMAP symbol is used to control logic partitioning into XC2000 and XC3000 family CLBs. Unlike the CLB symbol, the CLBMAP symbol is not a substitute for logic. It is used in addition to combinatorial gates, latches, and flip-flops for mapping control.

At the schematic level, you can implement a portion of logic using gates, latches, and flip-flops and specify that the logic be grouped into a single CLB by using the CLBMAP symbol. You must name the signals that are the inputs and outputs of the CLB, then draw the signals to appropriate pins of the CLBMAP symbol or name the CLBMAP signals and logic signals correspondingly. The symbol can have unconnected pins, but all signals on the logic group to be mapped must be specified on a symbol pin.

CLBMAP primitives and equivalent circuits are shown for XC2000 and XC3000 families in the following figures.

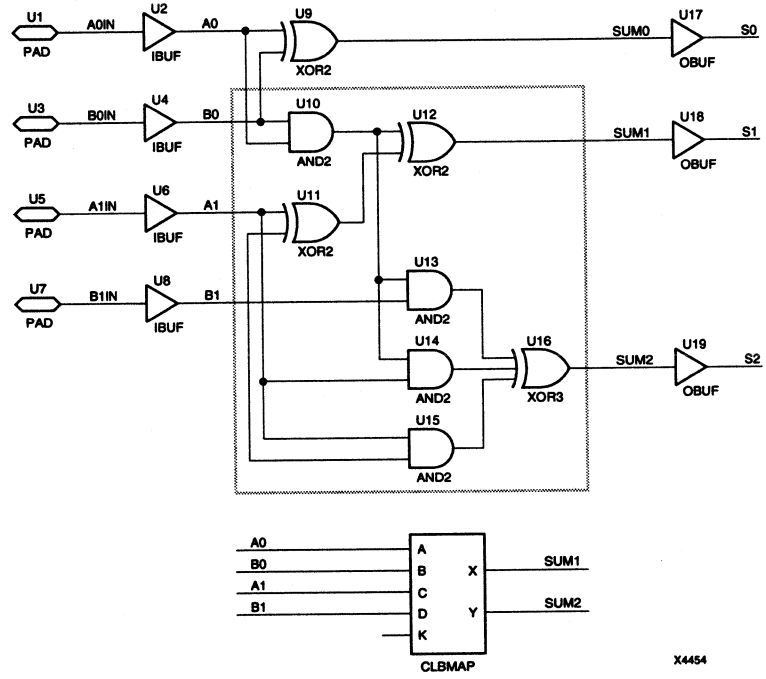


Figure 3-51 XC2000 CLBMAP Primitive Example and Equivalent

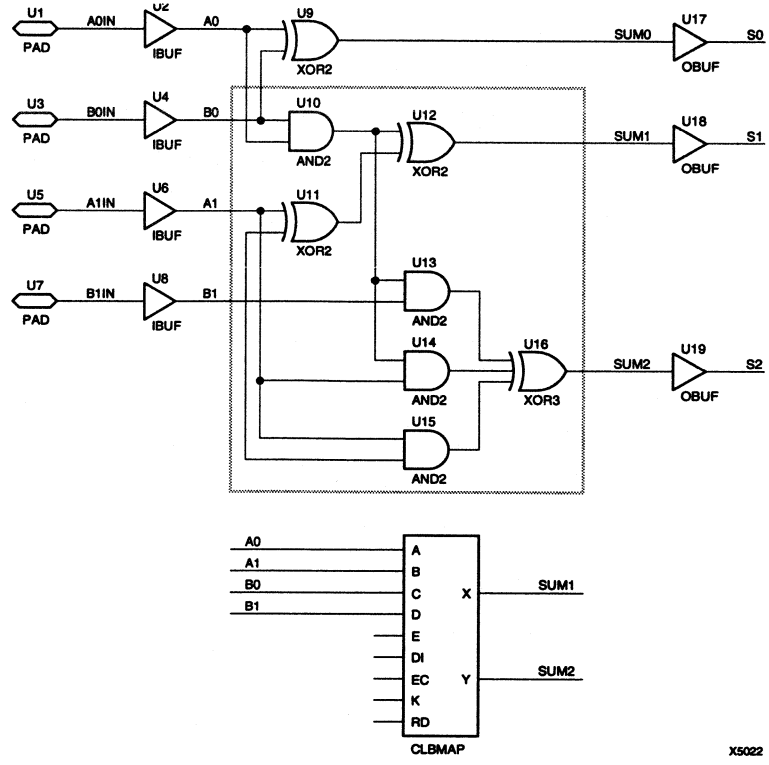


Figure 3-52 XC3000 CLBMAP Primitive Example and Equivalent

Use the MAP=*type* parameter with the CLBMAP symbol to further define how much latitude you want to give the mapping program. The following table shows MAP option characters and their meanings.

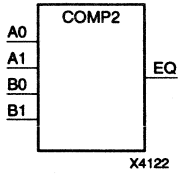
Character	Function
P	Pins
C	Closed – Adding logic to or removing logic from the CLB is not allowed.
L	Locked – Locking CLB pins
O	Open – Adding logic to or removing logic from the CLB is allowed.
U	Unlocked – No locking on CLB pins.

Possible types of MAP parameters for FMAP are: MAP=PUC, MAP=PLC, MAP=PLO, and MAP=PUO. The default parameter is PUC. If one of the "open" parameters is used (PLO or PUO), only the output signals must be specified.

You can lock individual pins using the "P" (Pin lock) parameter on the CLBMAP pin in conjunction with the PUC parameter. Refer to the appropriate CAE tool interface user guide for information on changing symbol parameters for your schematic editor.

COMP2

2-Bit Identity Comparator

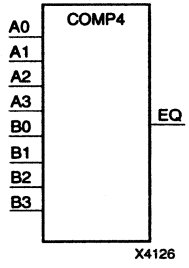


XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The equal output (EQ) of the COMP2 2-bit, identity comparator is High when the two words A1 – A0 and B1 – B0 are equal. Equality is determined by a bit comparison of the two words. When any two of the corresponding bits from each word are not the same, the EQ output is Low.

COMP4

4-Bit Identity Comparator

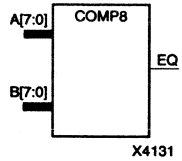


XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The equal output (EQ) of the COMP4 4-bit, identity comparator is High when the two words A3 – A0 and B3 – B0 are equal. Equality is determined by a bit comparison of the two words. When any two of the corresponding bits from each word are not the same, the EQ output is Low.

COMP8

8-Bit Identity Comparator



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The equal output (EQ) of the COMP8 8-bit, identity comparator is High when the two words A7 – A0 and B7 – B0 are equal. Equality is determined by a bit comparison of the two words. When any two of the corresponding bits from each word are not the same, the EQ output is Low.

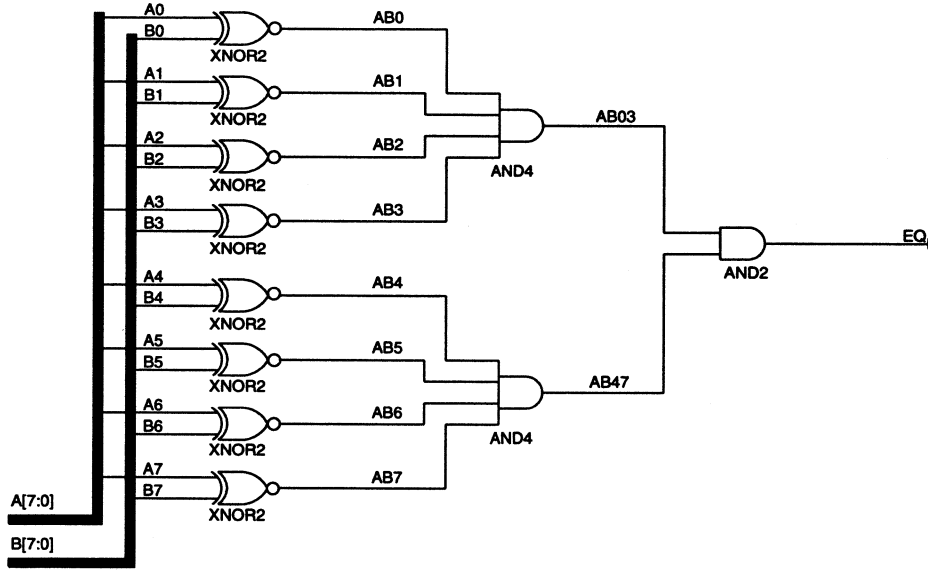
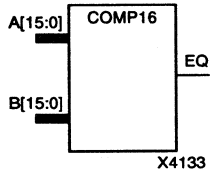


Figure 3-53 COMP8 XC2000/3000/4000 Implementation

COMP16

16-Bit Identity Comparator

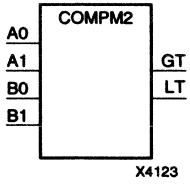


XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The equal output (EQ) of the COMP16 16-bit, identity comparator is High when the two words A15 – A0 and B15 – B0 are equal. Equality is determined by a bit comparison of the two words. When any two of the corresponding bits from each word are not the same, the EQ output is Low.

COMPM2

2-Bit Magnitude Comparator



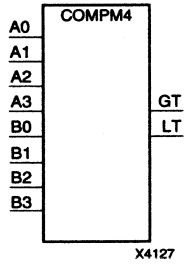
XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

COMPM2 is a 2-bit, magnitude comparator that compares two positive binary-weighted words A1 – A0 and B1 – B0, where A1 and B1 are the most significant bits. The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate.

Inputs				Outputs	
A1	**B1**	**A0**	**B0**	**GT**	**LT**
0	0	0	0	0	0
0	0	1	0	1	0
0	0	0	1	0	1
0	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	0
1	1	0	1	0	1
1	1	1	1	0	0
1	0	X	X	1	0
0	1	X	X	0	1

COMPM4

4-Bit Magnitude Comparator



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive*

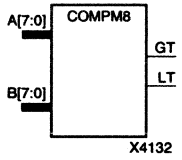
* not supported for XC7336 designs

COMPM4 is a 4-bit, magnitude comparator that compares two positive binary-weighted words A3 – A0 and B3 – B0, where A3 and B3 are the most significant bits. The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate.

Inputs				Outputs	
A3, B3	**A2, B2**	**A1, B1**	**A0, B0**	**GT**	**LT**
A3>B3	X	X	X	1	0
A3<B3	X	X	X	0	1
A3=B3	A2>B2	X	X	1	0
A3=B3	A2<B2	X	X	0	1
A3=B3	A2=B2	A1>B1	X	1	0
A3=B3	A2=B2	A1<B1	X	0	1
A3=B3	A2=A2	A1=B1	A0>B0	1	0
A3=B3	A2=B2	A1=B1	A0<B0	0	1
A3=B3	A2=B2	A1=B1	A0=B0	0	0

COMPM8

8-Bit Magnitude Comparator



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive*

* not supported for XC7336 designs

COMPM8 is an 8-bit, magnitude comparator that compares two positive binary-weighted words A7 – A0 and B7 – B0, where A7 and B7 are the most significant bits. The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate. Refer to the "COMPM4" section earlier in this chapter for a representative truth table.

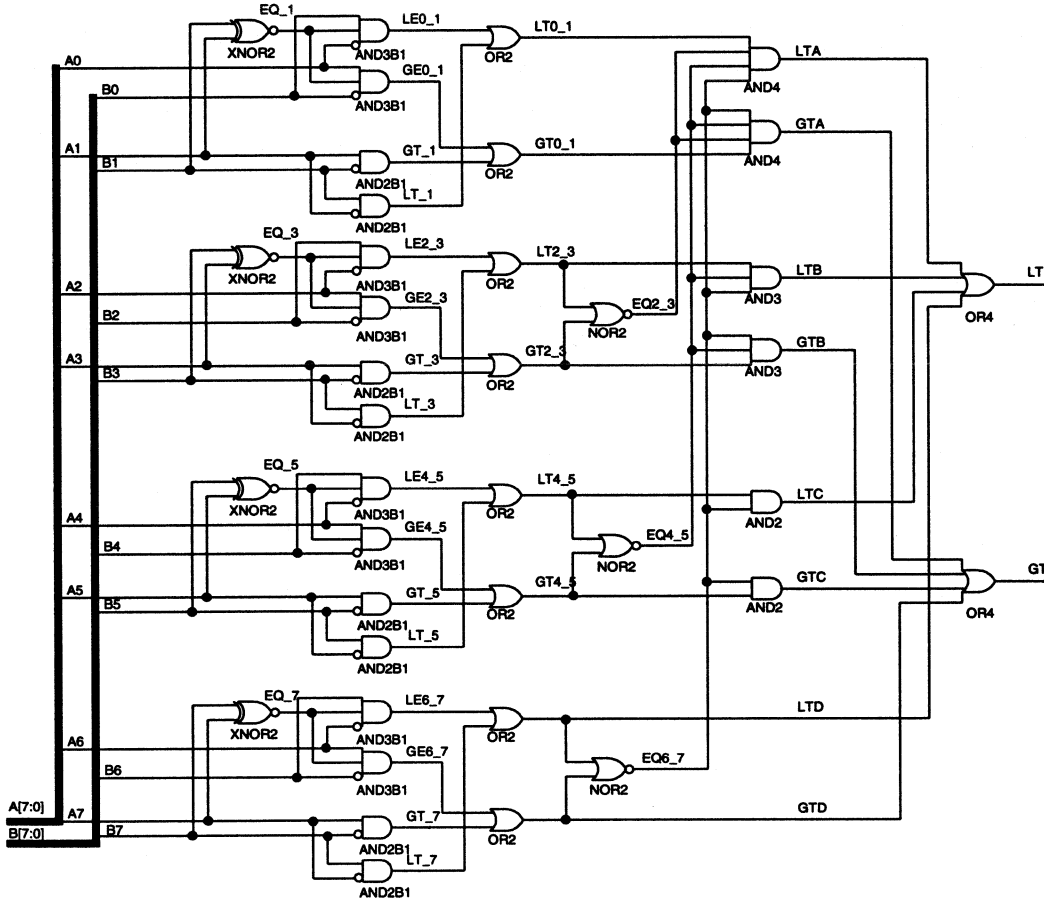
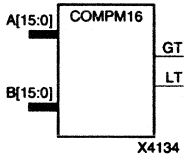


Figure 3-54 COMPM8 XC2000/3000/4000 Implementation

COMPM16

16-Bit Magnitude Comparator

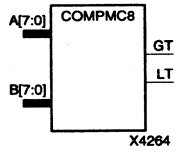


XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	N/A

COMPM16 is a 16-bit, magnitude comparator that compares two positive binary-weighted words A15 – A0 and B15 – B0, where A15 and B15 are the most significant bits. The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate. Refer to the "COMPM4" section earlier in this chapter for a representative truth table.

COMPMC8

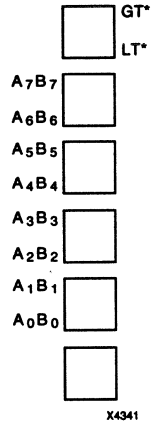
8-Bit Magnitude Comparator



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

COMPMC8 is an 8-bit, magnitude comparator that compares two positive binary-weighted words A7 – A0 and B7 – B0, where A7 and B7 are the most significant bits. The comparator is implemented using carry logic with relative location constraints, which assures most efficient logic placement. The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate. Refer to the "COMPM4" section earlier in this chapter for a representative truth table.

XC4000 Topology



In the process of combining the logic that loads GT and LT, the place and route software might map the logic that generates GT and LT to different function generators. If this mapping occurs, the GT and LT logic cannot be placed in the uppermost CLB, as indicated in the illustration.

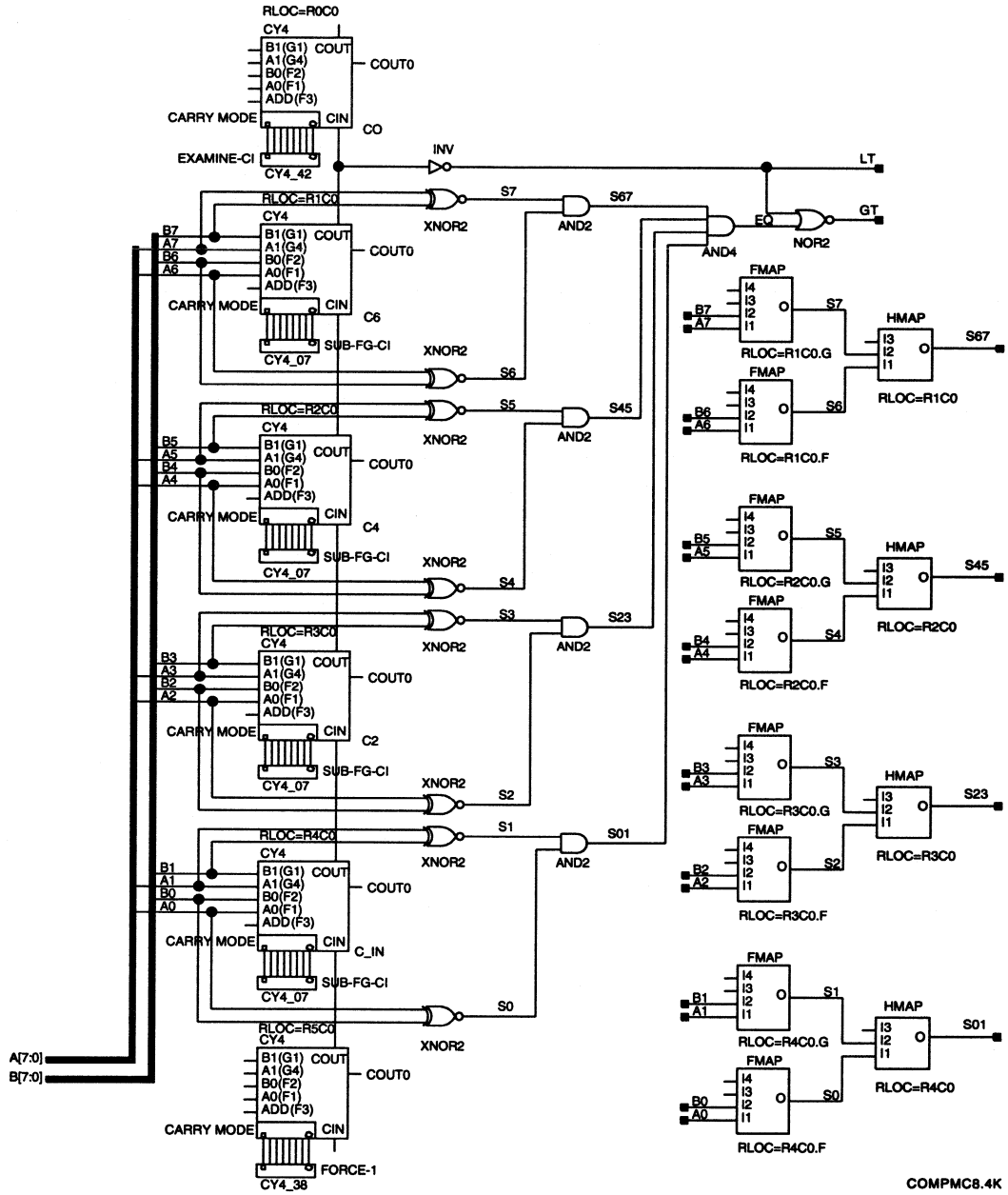
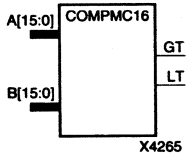


Figure 3-55 COMPMC8 XC4000 Implementation

COMPMC16

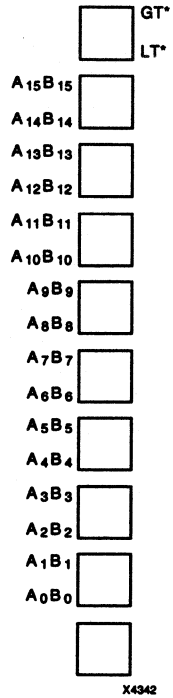
16-Bit Magnitude Comparator



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

COMPMC16 is a 16-bit, magnitude comparator that compares two positive binary-weighted words A15 – A0 and B15 – B0, where A15 and B15 are the most significant bits. The comparator is implemented using carry logic with relative location constraints, which assures most efficient logic placement. The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate. Refer to the "COMPM4" section earlier in this chapter for a representative truth table.

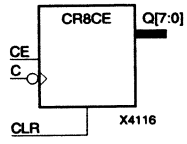
XC4000 Topology



In the process of combining the logic that loads GT and LT, the place and route software might map the logic that generates GT and LT to different function generators. If this mapping occurs, the GT and LT logic cannot be placed in the uppermost CLB, as indicated in the illustration.

CR8CE

8-Bit Negative-Edge Binary Ripple Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CR8CE is an 8-bit, cascadable, clearable, binary, ripple counter. The asynchronous clear (CLR), when High, overrides all other inputs and causes the outputs (Q7 – Q0) to go to logic level zero. The counter increments when the clock enable input (CE) is High during the High-to-Low clock (C) transition. The counter ignores clock transitions when CE is Low.

Larger counters can be created by connecting the Q7 output of the first stage to the clock input of the next stage. CLR and CE inputs are connected in parallel. The clock period is not affected by the overall length of a ripple counter. The overall clock-to-output propagation is n(TC - Q), where n is the number of stages and TC - Q is the C-to-Q7 propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. For XC7000, the clock (C) cannot be driven by a FastCLK (BUFG).

Inputs			Outputs
CLR	CE	C	Q7 – Q0
1	X	X	0
0	0	X	No Chg
0	1	↓	Inc

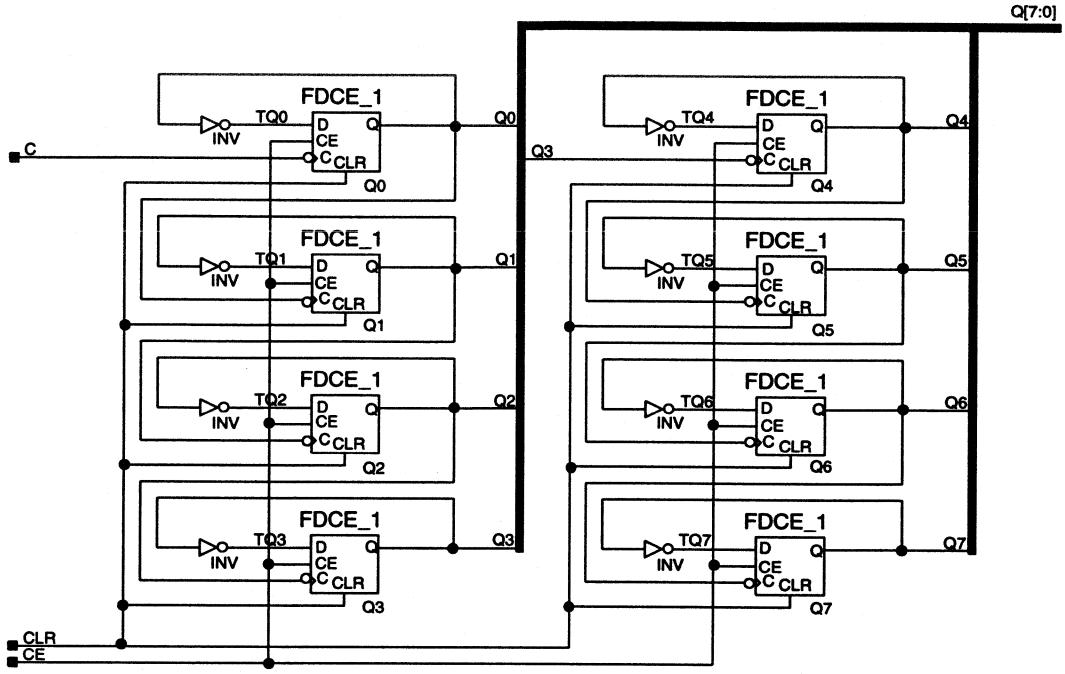
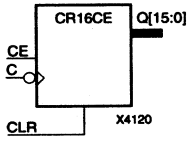


Figure 3-56 CR8CE XC2000/3000/4000 Implementation

CR16CE

16-Bit Negative-Edge Binary Ripple Counter with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

CR16CE is a 16-bit, cascadable, clearable, binary, ripple counter. The asynchronous clear (CLR), when High, overrides all other inputs and causes the outputs (Q15 – Q0) to go to logic level zero. The counter increments when the clock enable input (CE) is High during the High-to-Low clock (C) transition. The counter ignores clock transitions when CE is Low.

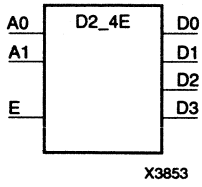
Larger counters can be created by connecting the Q15 output of the first stage to the clock input of the next stage. CLR and CE inputs are connected in parallel. The clock period is not affected by the overall length of a ripple counter. The overall clock-to-output propagation is *n*(TC - Q), where *n* is the number of stages and TC - Q is the C-to-Q15 propagation delay of each stage.

The counter is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. For XC7000, the clock (C) cannot be driven by a FastCLK (BUFG).

Inputs			Outputs
CLR	**CE**	**C**	**Q15 – Q0**
1	X	X	0
0	0	X	No Chg
0	1	↓	Inc

D2_4E

2- to 4-Line Decoder/Demultiplexer with Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the enable (EN) input of the D2_4E decoder/demultiplexer is High, one of four active-High outputs (D3 – D0) is selected with a 2-bit binary address (A1 – A0) input. The non-selected outputs are Low. Also, when the EN input is Low, all outputs are Low. In demultiplexer applications, the EN input is the data input.

Inputs			Outputs			
A1	A0	E	D3	D2	D1	D0
X	X	0	0	0	0	0
0	0	1	0	0	0	1
0	1	1	0	0	1	0
1	0	1	0	1	0	0
1	1	1	1	0	0	0

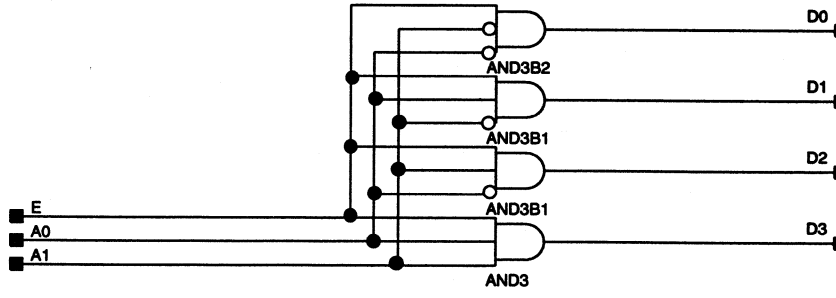
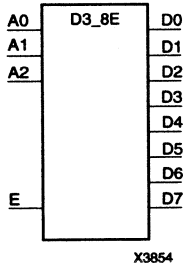


Figure 3-57 D2_4E XC2000/3000/4000 Implementation

D3_8E

3- to 8-Line Decoder/Demultiplexer with Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the enable (EN) input of the D3_8E decoder/demultiplexer is High, one of eight active-High outputs (D7 – D0) is selected with a 3-bit binary address (A2 – A0) input. The non-selected outputs are Low. Also, when the EN input is Low, all outputs are Low. In demultiplexer applications, the EN input is the data input.

Inputs				Outputs							
A2	A1	A0	E	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

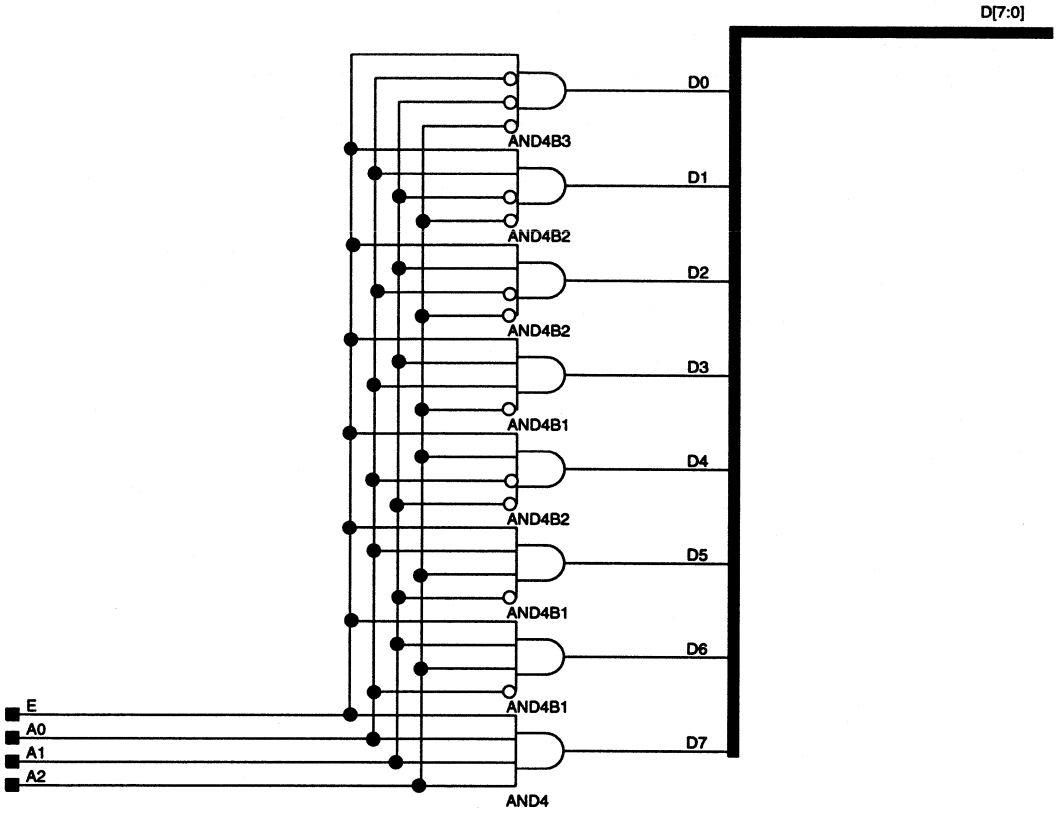
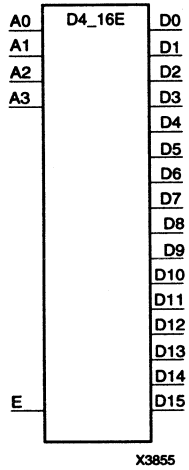


Figure 3-58 D3_8E XC2000/3000/4000 Implementation

D4_16E

4- to 16-Line Decoder/Demultiplexer with Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the enable (EN) input of the D4_16E decoder/demultiplexer is High, one of 16 active-High outputs (D15 – D0) is selected with a 4-bit binary address (A3 – A0) input. The non-selected outputs are Low. Also, when the EN input is Low, all outputs are Low. In demultiplexer applications, the EN input is the data input. Refer to "D3_8E" for truth table derivation.

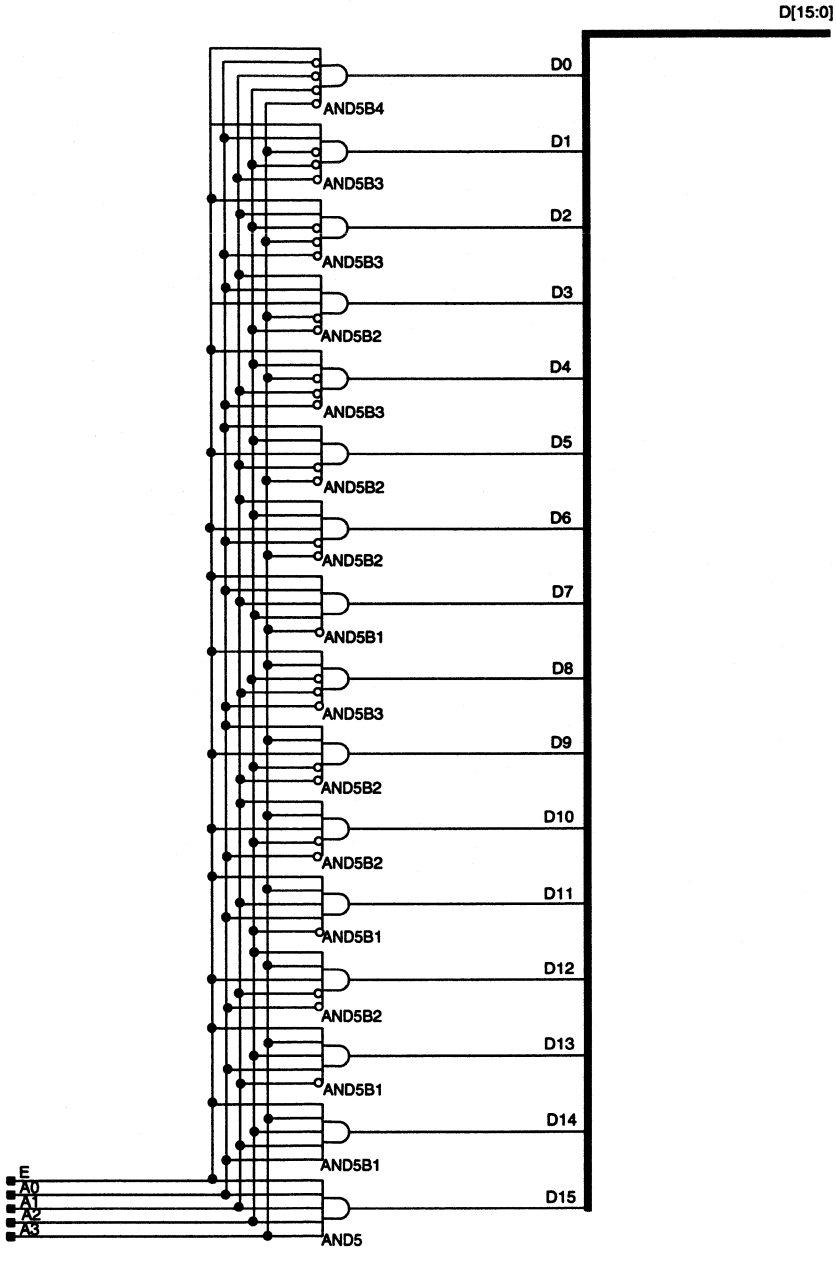
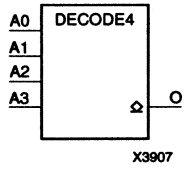
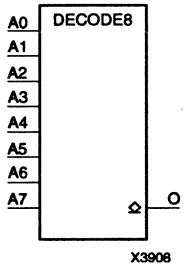


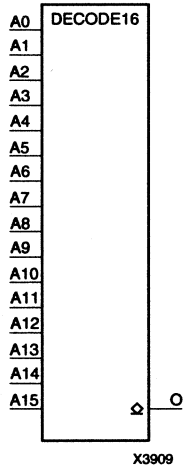
Figure 3-59 D4_16E XC2000/3000/4000 Implementation

DECODE4, DECODE8, and DECODE 16

4-, 8-, and 16-Bit Active-Low Edge Decoders







XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

These decoders are open-drain wired-AND gates. When one or more of the inputs (I) are Low, output (O) is Low. When all of the inputs are High, the output is High or "Off." A pull-up resistor must be connected to the output node to achieve a true logic High. A double pull-up resistor can be used to achieve faster performance but uses more power.

The XACT software implements these macros using the open-drain AND gates around the periphery of the XC4000 devices.

Note: Diamonds in library symbols indicate an open-drain output.

Inputs				Outputs
I0	I1	...	In – 1	O
1	1	1	1	1
0	X	X	X	0
X	0	X	X	0
X	X	X	0	0

A pull-up resistor must be connected to the output to establish High-level drive current.

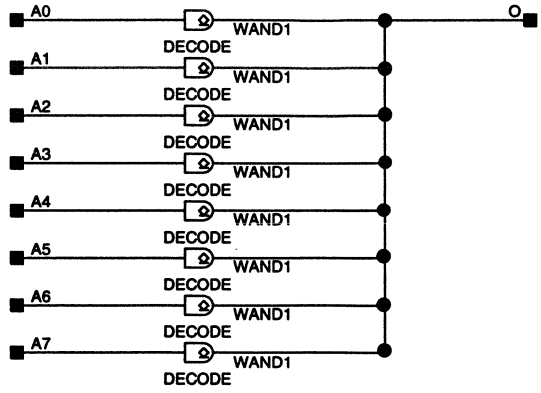
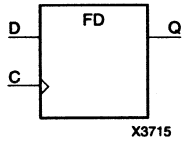


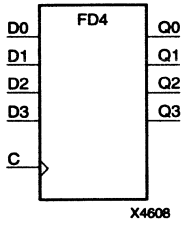
Figure 3-60 DECODE8 XC4000 Implementation

FD, FD4, FD8, and FD16

Single and Multiple D Flip-Flops

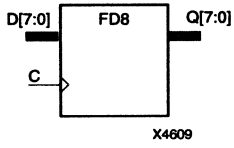


Element	XC2000	XC3000	XC4000	XC7000
FD	Macro	Macro	Macro	Primitive
FD4, FD8, FD16	N/A	N/A	N/A	Primitive

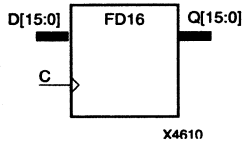


FD is a single D-type flip-flop with data input (D) and data output (Q). FD4, FD8, and FD16 are 4-bit, 8-bit, and 16-bit registers, each with a common clock (C). The data on the D inputs is loaded into the flip-flop during the Low-to-High clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.



Inputs		Outputs
D	C	Q
0	↑	0
1	↑	1



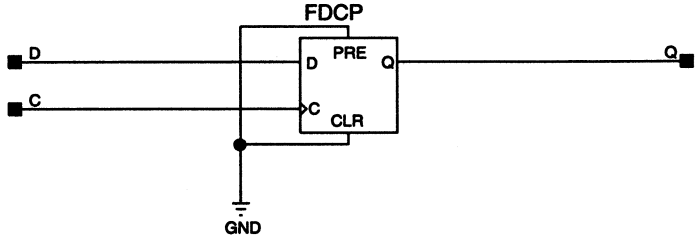


Figure 3-61 FD XC2000 Implementation

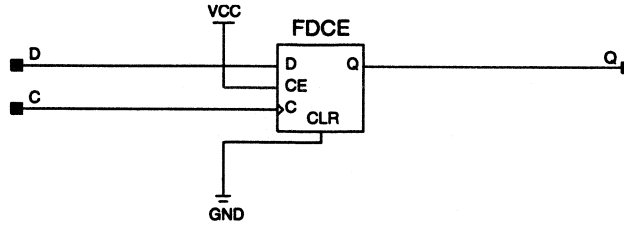
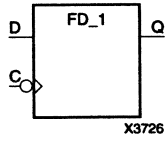


Figure 3-62 FD XC3000/4000 Implementation

FD_1

D Flip-Flop with Negative-Edge Clock



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	N/A

FD_1 is a single D-type flip-flop with data input (D) and data output (Q). The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs		Outputs
D	C	Q
0	↓	0
1	↓	1

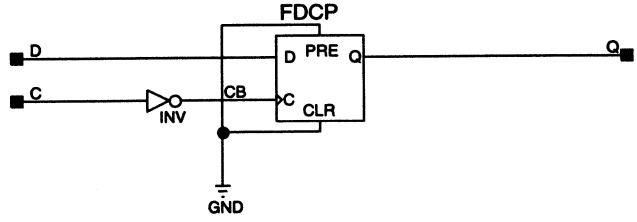


Figure 3-63 FD_1 XC2000 Implementation

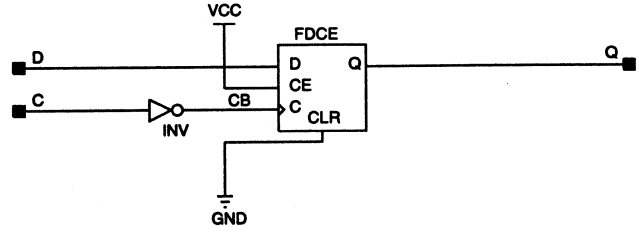
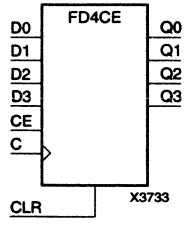


Figure 3-64 FD_1 XC3000/4000 Implementation

FD4CE

4-Bit Data Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When clock enable (CE) is High, and asynchronous clear (CLR) is Low, the data on the four data inputs (D3 – D0) of FD4CE is transferred to the corresponding data outputs (Q3 – Q0) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data outputs (Q3 – Q0) Low. When CE is Low, clock transitions are ignored.

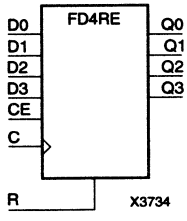
The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
CLR	CE	D3 – D0	C	Q3 – Q0
1	X	X	X	0
0	0	X	X	No Chg
0	1	Dn	↑	dn

dn = state of corresponding input one set-up time prior to active clock transition

FD4RE

4-Bit Data Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the clock enable (CE) input is High, and the synchronous reset (R) input is Low, the data on the four data inputs (D3 – D0) of FD4RE is transferred to the corresponding data outputs (Q3 – Q0) during the Low-to-High clock (C) transition. When R is High, it overrides all other inputs and resets the data outputs (Q3 – Q0) Low on the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

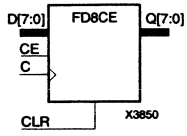
The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
R	CE	D3 – D0	C	Q3 – Q0
1	X	X	↑	0
0	0	X	X	No Chg
0	1	Dn	↑	dn

dn = state of corresponding input one set-up time prior to active clock transition

FD8CE

8-Bit Data Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When clock enable (CE) is High, and asynchronous clear (CLR) is Low, the data on the eight data inputs (D7 – D0) of FD8CE is transferred to the corresponding data outputs (Q7 – Q0) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data outputs (Q7 – Q0) Low. When CE is Low, clock transitions are ignored.

The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
CLR	**CE**	**D7 – D0**	**C**	**Q7 – Q0**
1	X	X	X	0
0	0	X	X	No Chg
0	1	Dn	↑	dn

dn = state of corresponding input one set-up time prior to active clock transition

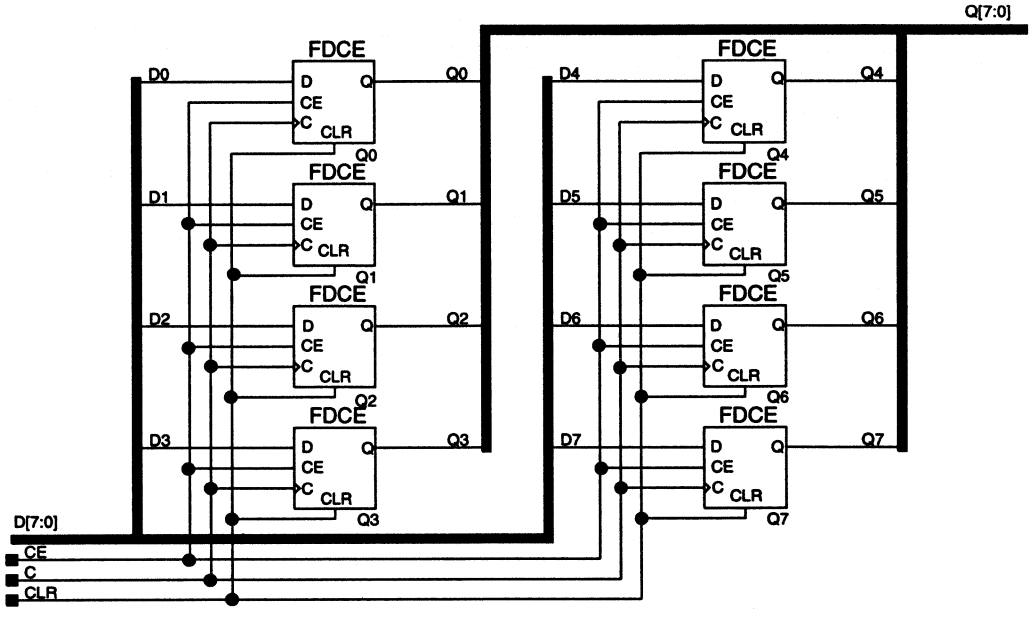
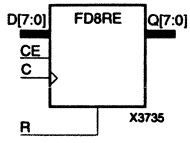


Figure 3-65 FD8CE XC2000/3000/4000 Implementation

FD8RE

8-Bit Data Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the clock enable (CE) input is High, and the synchronous reset (R) input is Low, the data on the eight data inputs (D7 – D0) of FD8RE is transferred to the corresponding data outputs (Q7 – Q0) during the Low-to-High clock (C) transition. When R is High, it overrides all other inputs and resets the data outputs (Q7 – Q0) Low on the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
R	**CE**	**D7 – D0**	**C**	**Q7 – Q0**
1	X	X	↑	0
0	0	X	X	No Chg
0	1	Dn	↑	dn

dn = state of corresponding input one set-up time prior to active clock transition

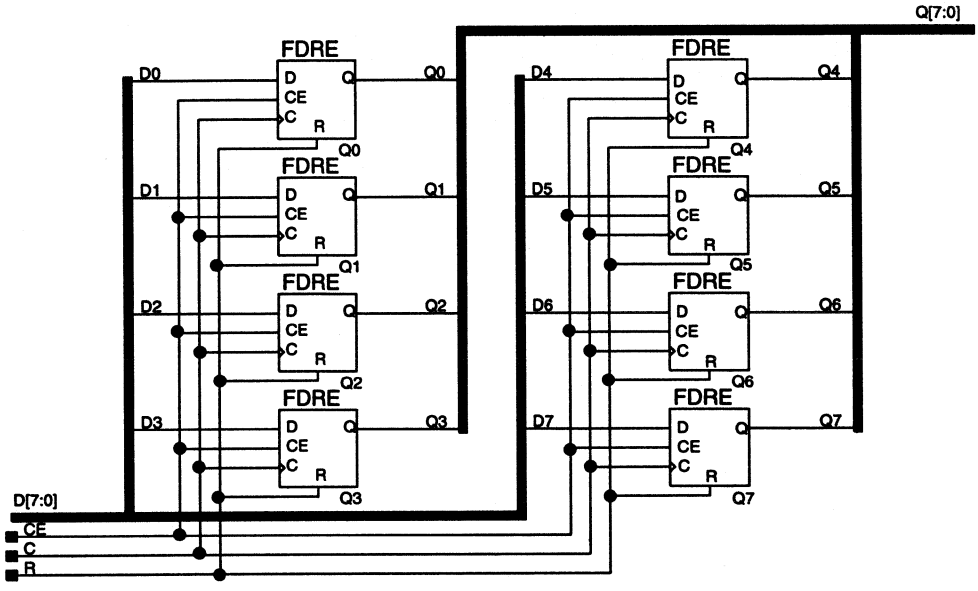
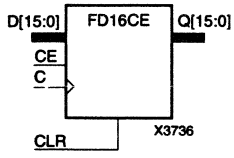


Figure 3-66 FD8RE XC2000/3000/4000 Implementation

FD16CE

16-Bit Data Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When clock enable (CE) is High, and asynchronous clear (CLR) is Low, the data on the 16 data inputs (D15 – D0) of FD16CE is transferred to the corresponding data outputs (Q15 – Q0) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data outputs (Q15 – Q0) Low. When CE is Low, clock transitions are ignored.

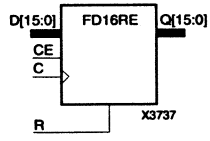
The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
CLR	CE	D15 – D0	C	Q15 – Q0
1	X	X	X	0
0	0	X	X	No Chg
0	1	Dn	↑	dn

dn = state of corresponding input one set-up time prior to active clock transition

FD16RE

16-Bit Data Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the clock enable (CE) input is High, and the synchronous reset (R) input is Low, the data on the 16 data inputs (D15 – D0) of FD16RE is transferred to the corresponding data outputs (Q15 – Q0) during the Low-to-High clock (C) transition. When R is High, it overrides all other inputs and resets the data outputs (Q15 – Q0) Low on the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

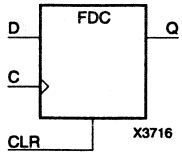
The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
R	CE	D15 – D0	C	Q15 – Q0
1	X	X	↑	0
0	0	X	X	No Chg
0	1	Dn	↑	dn

dn = state of corresponding input one set-up time prior to active clock transition

FDC

D Flip-Flop with Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDC is a single D-type flip-flop with data (D) and asynchronous clear (CLR) inputs and data output (Q). The asynchronous CLR, when High, overrides all other inputs and sets the Q output Low. The data on the D input is loaded into the flip-flop when CLR is Low on the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
CLR	D	C	Q
1	X	X	0
0	1	↑	1
0	0	↑	0

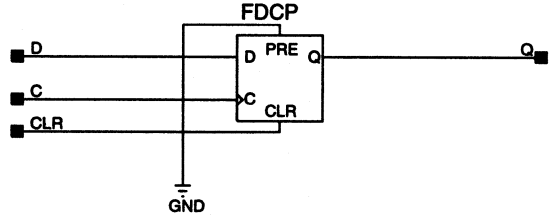


Figure 3-67 FDC XC2000 Implementation

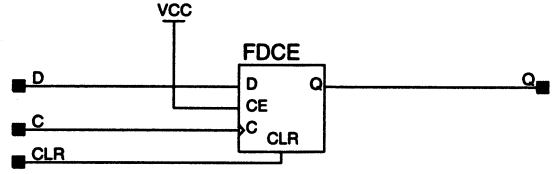
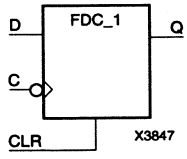


Figure 3-68 FDC XC3000/4000 Implementation

FDC_1

D Flip-Flop with Negative-Edge Clock and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	N/A

FDC_1 is a single D-type flip-flop with data input (D), asynchronous clear input (CLR) and data output (Q). The asynchronous CLR, when active, overrides all other inputs and sets the Q output Low. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
CLR	D	C	Q
1	X	X	0
0	1	↓	1
0	0	↓	0

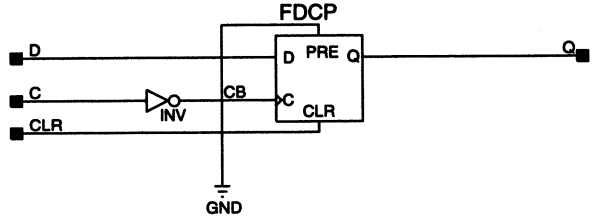


Figure 3-69 FDC_1 XC2000 Implementation

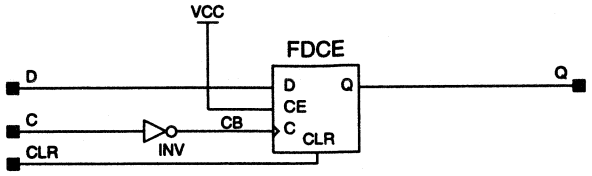
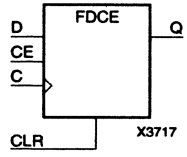


Figure 3-70 FDC_1 XC3000/4000 Implementation

FDCE

D Flip-Flop with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Primitive	Primitive	Primitive

When clock enable (CE) is High, and asynchronous clear (CLR) is Low, the data on the data input (D) of FDCE is transferred to the corresponding data output (Q) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data output (Q) Low. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
CLR	CE	D	C	Q
1	X	X	X	0
0	0	X	X	No Chg
0	1	1	↑	1
0	1	0	↑	0

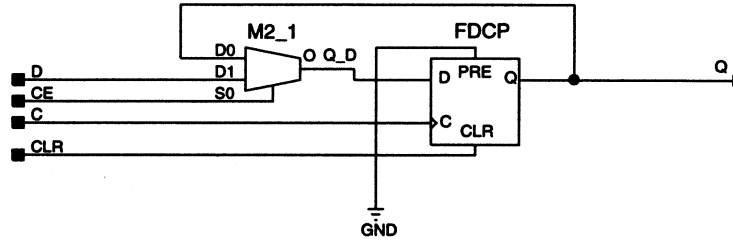
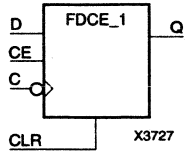


Figure 3-71 FDCE XC2000 Implementation

FDCE_1

D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	N/A

FDCE_1 is a single D-type flip-flop with data (D), clock enable (CE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous CLR input, when High, overrides all other inputs and sets the Q output Low. The data on the D input is loaded into the flip-flop when CLR is Low and CE is High on the High-to-Low clock (C) transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
CLR	**CE**	**D**	**C**	**Q**
1	X	X	X	0
0	0	X	↓	No Chg
0	1	1	↓	1
0	1	0	↓	0

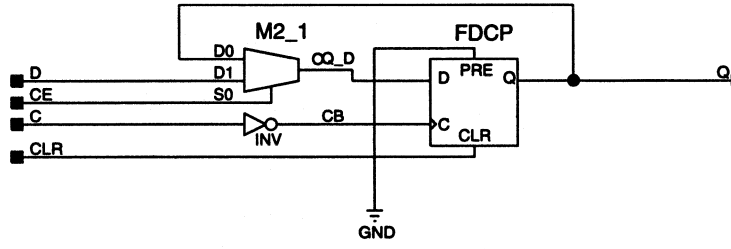


Figure 3-72 FDCE_1 XC2000 Implementation

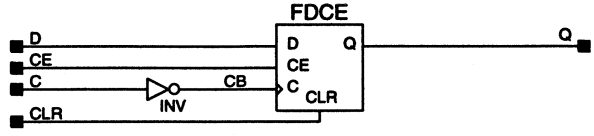
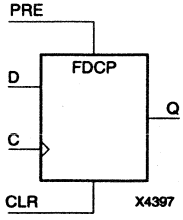


Figure 3-73 FDCE_1 XC3000/4000 Implementation

FDCP

D Flip-Flop with Asynchronous Preset and Clear



XC2000	XC3000	XC4000	XC7000
Primitive	N/A	N/A	Primitive*

* not supported for XC7336 designs

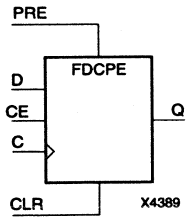
FDCP is a single D-type flip-flop with data (D), asynchronous preset (PRE) and clear (CLR) inputs, and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. When both PRE and CLR are active, the flip-flop is cleared. Data on the D input is loaded into the flip-flop when PRE and CLR are Low on the Low-to-High clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR) is active (Low).

Inputs				Outputs
CLR	PRE	D	C	Q
1	X	X	X	0
0	1	X	X	1
0	0	0	↑	0
0	0	1	↑	1

FDCPE

D Flip-Flop with Clock Enable and Asynchronous Preset and Clear



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

FDCPE is a single D-type flip-flop with data (D), clock enable (CE), asynchronous preset (PRE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. When both PRE and CLR are active, the flip-flop is cleared. Data on the D input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High clock (C) transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR) is active (Low).

Inputs					Outputs
CLR	PRE	CE	D	C	Q
1	X	X	X	X	0
0	1	X	X	X	1
0	0	0	X	X	No Chg
0	0	1	0	↑	0
0	0	1	1	↑	1

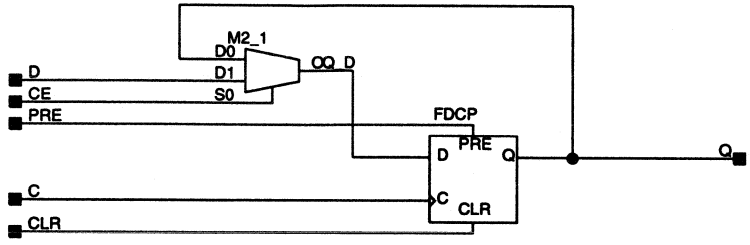
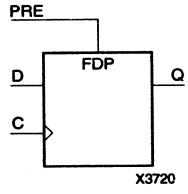


Figure 3-74 FDCPE XC2000 Implementation

FDP

D Flip-Flop with Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	Primitive

FDP is a single D-type flip-flop with data (D) and asynchronous preset (PRE) inputs, and data output (Q). The asynchronous PRE, when High, overrides all other inputs and presets the Q output High. The data on the D input is loaded into the flip-flop when PRE is Low on the Low-to-High clock (C) transition. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active; the GSR active level is programmable.

Inputs			Outputs
PRE	C	D	Q
1	X	X	1
0	↑	1	1
0	↑	0	0

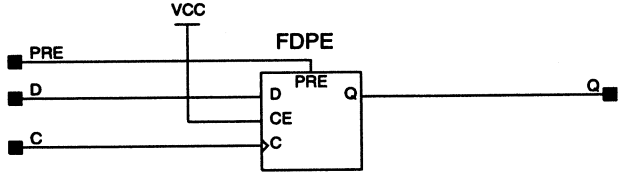
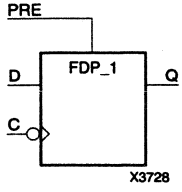


Figure 3-75 FDP XC4000 Implementation

FDP_1

D Flip-Flop with Negative-Edge Clock and Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

FDP_1 is a single D-type flip-flop with data (D) and asynchronous preset (PRE) inputs, and data output (Q). The asynchronous PRE, when High, overrides all other inputs and presets the Q output High. The data on the D input is loaded into the flip-flop when PRE is Low on the High-to-Low clock (C) transition. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active; the GSR active level is programmable.

Inputs			Outputs
PRE	C	D	Q
1	X	X	1
0	↓	1	1
0	↓	0	0

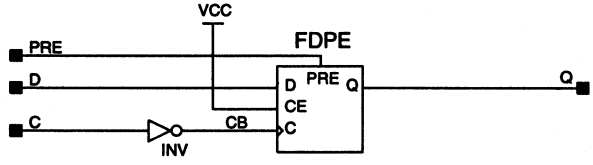
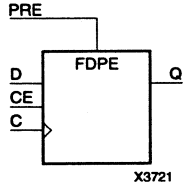


Figure 3-76 FDP_1 XC4000 Implementation

FDPE

D Flip-Flop with Clock Enable and Asynchronous Preset



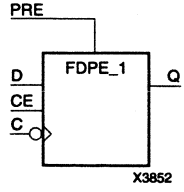
XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	Primitive

FDPE is a single D-type flip-flop with data (D), clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and sets the Q output High. Data on the D input is loaded into the flip-flop when PRE is Low and CE is High on the Low-to-High clock (C) transition. When CE is Low, the clock transitions are ignored. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active; the GSR active level is programmable.

Inputs				Outputs
PRE	CE	D	C	Q
1	X	X	X	1
0	0	X	X	No Chg
0	1	0	↑	0
0	1	1	↑	1

FDPE_1

D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

FDPE_1 is a single D-type flip-flop with data (D), clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and sets the Q output High. Data on the D input is loaded into the flip-flop when PRE is Low and CE is High on the High-to-Low clock (C) transition. When CE is Low, the clock transitions are ignored. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active; the GSR active level is programmable.

Inputs				Outputs
PRE	**CE**	**D**	**C**	**Q**
1	X	X	X	1
0	0	X	X	No Chg
0	1	1	↓	1
0	1	0	↓	0

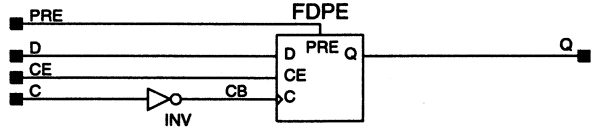
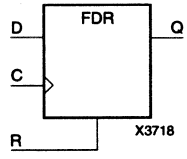


Figure 3-77 FDPE_1 XC4000 Implementation

FDR

D Flip-Flop with Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDR is a single D-type flip-flop with data (D) and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when R is Low during the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
R	D	C	Q
1	X	↑	0
0	1	↑	1
0	0	↑	0

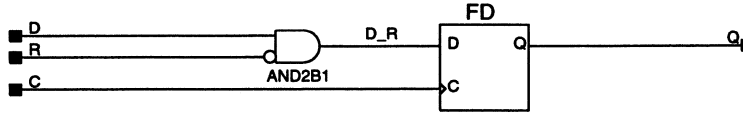
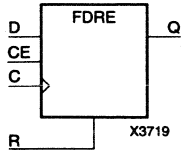


Figure 3-78 FDR XC2000/3000/4000 Implementation

FDRE

D Flip-Flop with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDRE is a single D-type flip-flop with data (D), clock enable (CE), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when R is Low and CE is High during the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
R	CE	D	C	Q
1	X	X	↑	0
0	0	X	X	No Chg
0	1	1	↑	1
0	1	0	↑	0

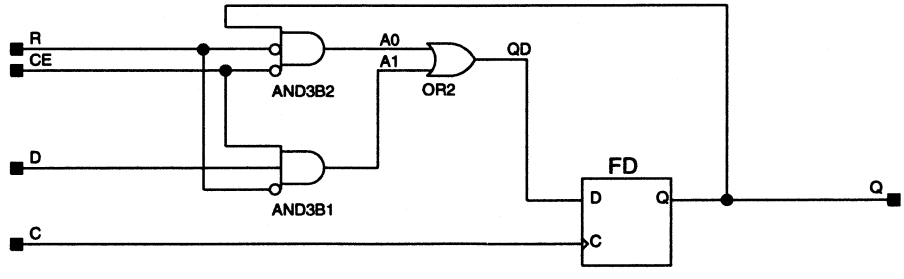
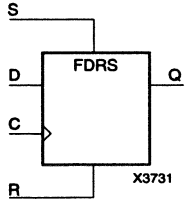


Figure 3-79 FDRE XC2000/3000/4000 Implementation

FDRS

D Flip-Flop with Synchronous Reset and Synchronous Set



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDRS is a single D-type flip-flop with data (D), synchronous set (S), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High clock (C) transition. (Reset has precedence over Set.) When S is High and R is Low, the flip-flop is set, output High, during the Low-to-High clock transition. When R and S are Low, data on the (D) input is loaded into the flip-flop during the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
R	S	D	C	Q
1	X	X	↑	0
0	1	X	↑	1
0	0	1	↑	1
0	0	0	↑	0

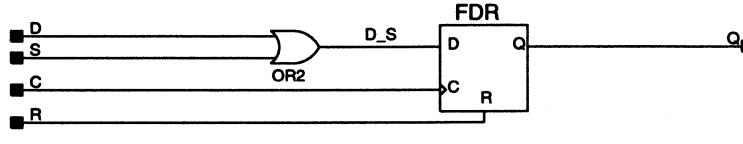
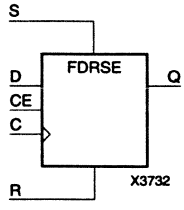


Figure 3-80 FDRS XC2000/3000/4000 Implementation

FDRSE

D Flip-Flop with Synchronous Reset and Set and Clock Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDRSE is a single D-type flip-flop with synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). The reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High clock transition. (Reset has precedence over Set.) When the set (S) input is High and R is Low, the flip-flop is set, output High, during the Low-to-High clock (C) transition. Data on the D input is loaded into the flip-flop when R and S are Low and CE is High during the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs					Outputs
R	S	CE	D	C	Q
1	X	X	X	↑	0
0	1	X	X	↑	1
0	0	0	X	X	No Chg
0	0	1	1	↑	1
0	0	1	0	↑	0

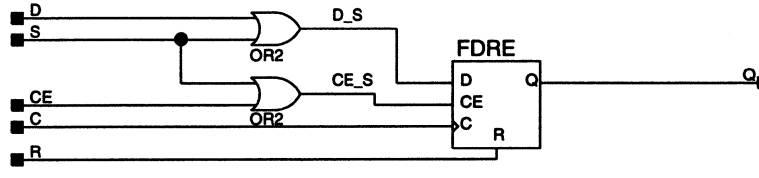
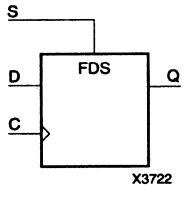


Figure 3-81 FDRSE XC2000/3000/4000 Implementation

FDS

D Flip-Flop with Synchronous Set



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDS is a single D-type flip-flop with data (D) and synchronous set (S) inputs and data output (Q). The synchronous set input, when High, sets the Q output High on the Low-to-high clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low during the Low-to-High clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
S	D	C	Q
1	X	↑	1
0	1	↑	1
0	0	↑	0

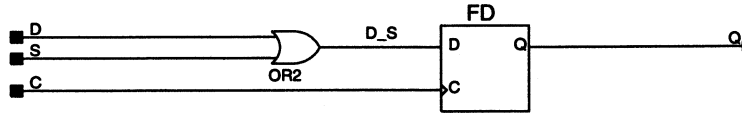
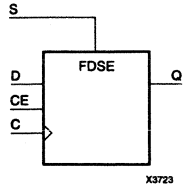


Figure 3-82 FDS XC2000/3000/4000 Implementation

FDSE

D Flip-Flop with Clock Enable and Synchronous Set



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDSE is a single D-type flip-flop with data (D), clock enable (CE), and synchronous set (S) inputs and data output (Q). The synchronous set (S) input, when High, overrides the clock enable (CE) input and sets the Q output High during the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low and CE is High during the Low-to-High clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
S	CE	D	C	Q
1	X	X	↑	1
0	0	X	X	No Chg
0	1	1	↑	1
0	1	0	↑	0

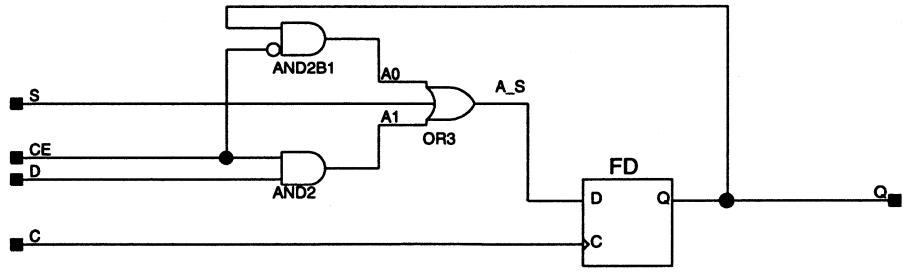
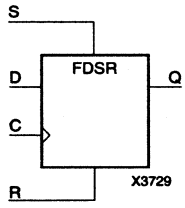


Figure 3-83 FDSE XC2000/3000/4000 Implementation

FDSR

D Flip-Flop with Synchronous Set and Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDSR is a single D-type flip-flop with data (D), synchronous reset (R) and synchronous set (S) inputs and data output (Q). When the set (S) input is High, it overrides all other inputs and sets the Q output High during the Low-to-High clock transition. (Set has precedence over Reset.) When reset (R) is High and S is Low, the flip-flop is reset, output Low, on the Low-to-High clock transition. Data on the D input is loaded into the flip-flop when S and R are Low on the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
S	R	D	C	Q
1	X	X	↑	1
0	1	X	↑	0
0	0	1	↑	1
0	0	0	↑	0

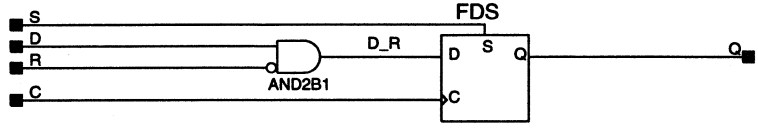
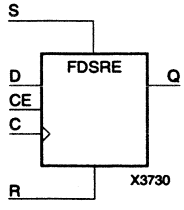


Figure 3-84 FDSR XC2000/3000/4000 Implementation

FDSRE

D Flip-Flop with Synchronous Set and Reset and Clock Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FDSRE is a single D-type flip-flop with synchronous set (S), synchronous reset (R), and clock enable (CE) inputs and data output (Q). When synchronous set (S) is High, it overrides all other inputs and sets the Q output High during the Low-to-High clock transition. (Set has precedence over Reset.) When synchronous reset (R) is High and S is Low, output Q is reset Low during the Low-to-High clock transition. Data is loaded into the flip-flop when S and R are Low and CE is High during the Low-to-high clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs					Outputs
S	R	CE	D	C	Q
1	X	X	X	↑	1
0	1	X	X	↑	0
0	0	0	X	X	No Chg
0	0	1	1	↑	1
0	0	1	0	↑	0

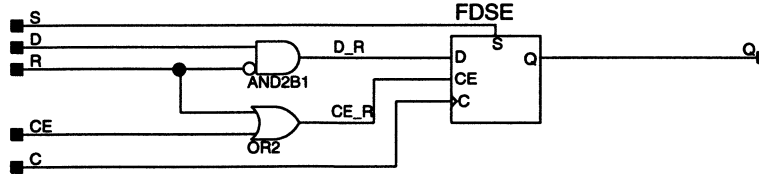
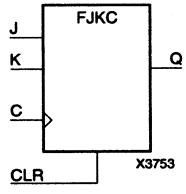


Figure 3-85 FDSRE XC2000/3000/4000 Implementation

FJKC

J-K Flip-Flop with Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FJKC is a single J-K-type flip-flop with J, K, and asynchronous clear (CLR) inputs and data output (Q). The asynchronous clear (CLR) input, when High, overrides all other inputs and resets the Q output Low. When CLR is Low, the output responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
CLR	J	K	C	Q
1	X	X	X	0
0	0	0	↑	No Chg
0	0	1	↑	0
0	1	0	↑	1
0	1	1	↑	Toggle

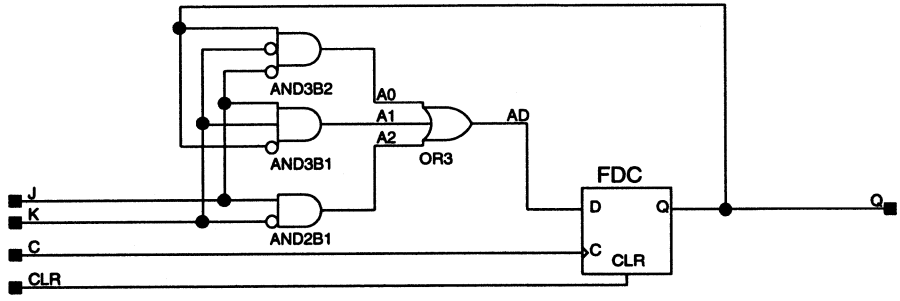
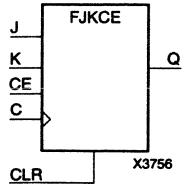


Figure 3-86 FJKC XC2000/3000/4000 Implementation

FJKCE

J-K Flip-Flop with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FJKCE is a single J-K-type flip-flop with J, K, clock enable (CE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous clear (CLR), when High, overrides all other inputs and resets the Q output Low during the Low-to-High clock (C) transition. When CLR is Low and CE is High, Q responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs					Outputs
CLR	CE	J	K	C	Q
1	X	X	X	X	0
0	0	X	X	X	No Chg
0	1	0	0	X	No Chg
0	1	0	1	↑	0
0	1	1	0	↑	1
0	1	1	1	↑	Toggle

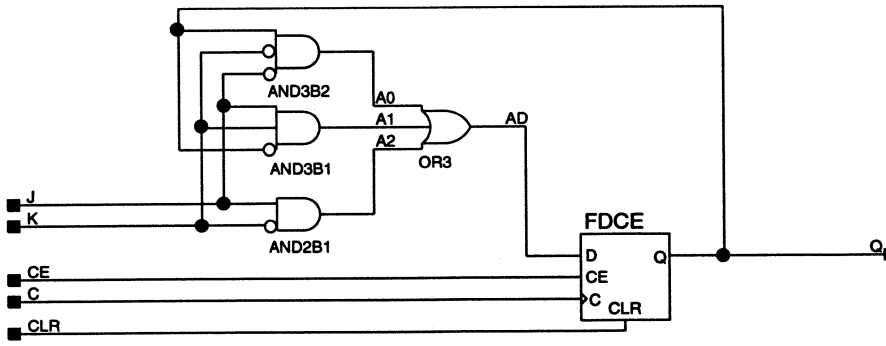
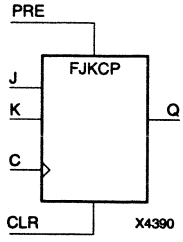


Figure 3-87 FJKCE XC2000/3000/4000 Implementation

FJKCP

J-K Flip-Flop with Asynchronous Clear and Preset



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

FJKCP is a single J-K-type flip-flop with J, K, asynchronous clear (CLR), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous clear input (CLR), when High, overrides all other inputs and resets the Q output Low on the High-to-Low clock (C) transition. The asynchronous preset (PRE) input, when High (provided CLR is Low), overrides all other inputs and sets the Q output High on the Low-to-High clock (C) transition. When CLR and PRE are Low, Q responds to the state of the J and K inputs during the Low-to-High clock transition, as shown in the following truth table.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR) is active (Low).

Inputs					Outputs
CLR	PRE	J	K	C	Q
1	X	X	X	X	0
0	1	X	X	X	1
0	0	0	0	X	No Chg
0	0	0	1	↑	0
0	0	1	0	↑	1
0	0	1	1	↑	Toggle

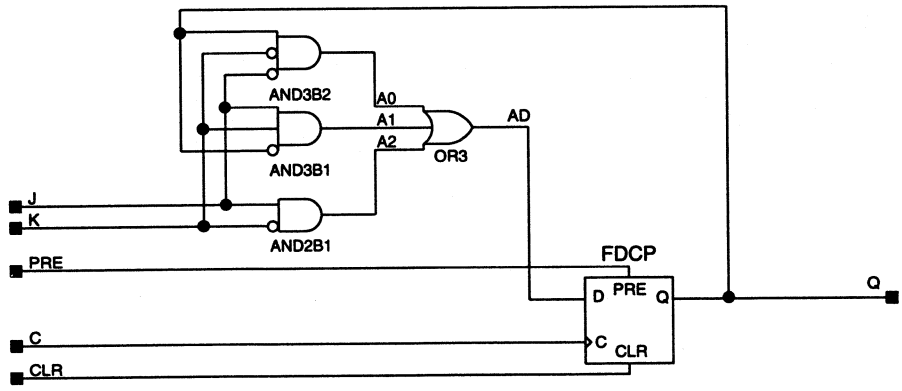
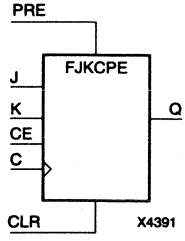


Figure 3-88 FJKCP XC2000 Implementation

FJKCPE

J-K Flip-Flop with Asynchronous Clear and Preset and Clock Enable



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

FJKCPE is a single J-K-type flip-flop with J, K, asynchronous clear (CLR), asynchronous preset (PRE), and clock enable (CE) inputs and data output (Q). The asynchronous clear input (CLR), when High, overrides all other inputs and resets the Q output Low on the High-to-Low clock (C) transition. The asynchronous preset (PRE) input, when High (provided CLR is Low), overrides all other inputs and sets the Q output High on the Low-to-High clock (C) transition. When CLR and PRE are Low and CE is High, Q responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock transition. Clock transitions are ignored when CE is Low.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR) is active (Low).

Inputs						Outputs
CLR	PRE	CE	J	K	C	Q
1	X	X	X	X	X	0
0	1	X	X	X	X	1
0	0	0	0	X	X	No Chg
0	0	1	0	0	X	No Chg
0	0	1	0	1	↑	0
0	0	1	1	0	↑	1
0	0	1	1	1	↑	Toggle

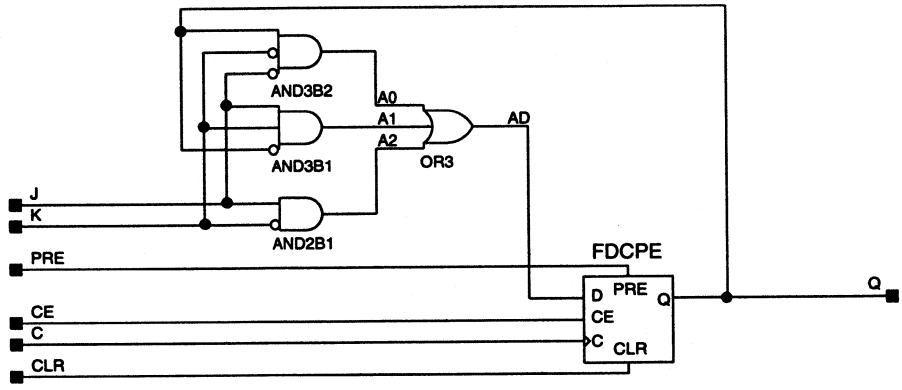
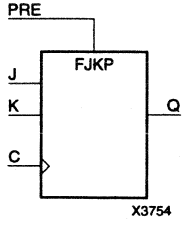


Figure 3-89 FJKCPE XC2000 Implementation

FJKP

J-K Flip-Flop with Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	Primitive

FJKP is a single J-K-type flip-flop with J, K, and asynchronous preset (PRE) inputs and data output (Q). The asynchronous preset (PRE) input, when High, overrides all other inputs and sets the Q output High on the Low-to-High clock (C) transition. When PRE is Low, the Q output responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock transition. The flip-flop is asynchronously set, output High, when global set/reset GSR) is active. The GSR active level is programmable.

Inputs				Outputs
PRE	J	K	C	Q
1	X	X	X	1
0	0	0	X	No Chg
0	0	1	↑	0
0	1	0	↑	1
0	1	1	↑	Toggle

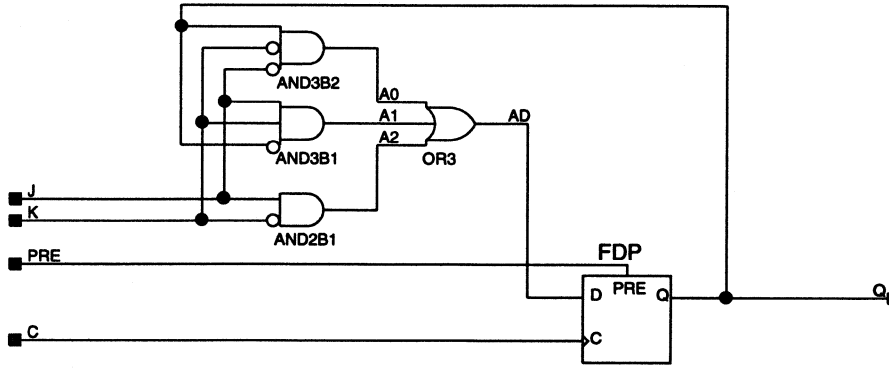
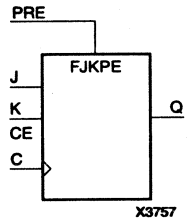


Figure 3-90 FJKP XC4000 Implementation

FJKPE

J-K Flip-Flop with Clock Enable and Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	Primitive

FJKPE is a single J-K-type flip-flop with J, K, clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous preset (PRE), when high, overrides all other inputs and sets the Q output High. When PRE is Low and CE is High, the Q output responds to the state of the J and K inputs, according to the following truth table, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs					Outputs
PRE	CE	J	K	C	Q
1	X	X	X	X	1
0	0	X	X	X	No Chg
0	1	0	0	X	No Chg
0	1	0	1	↑	0
0	1	1	0	↑	1
0	1	1	1	↑	Toggle

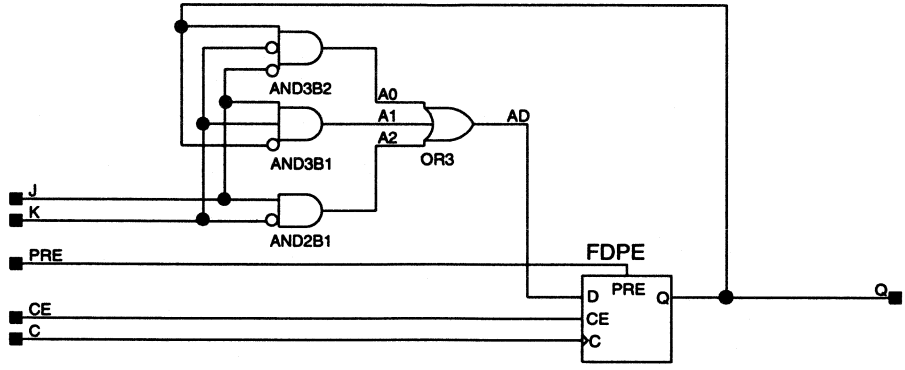
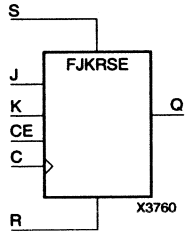


Figure 3-91 FJKPE XC4000 Implementation

FJKRSE

J-K Flip-Flop with Clock Enable and Synchronous Reset and Set



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FJKRSE is a single J-K-type flip-flop with J, K, synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). When synchronous reset (R) is High, all other inputs are ignored and output Q is reset Low. (Reset has precedence over Set.) When synchronous set (S) is High and R is Low, output Q is set High. When R and S are Low and CE is High, output Q responds to the state of the J and K inputs, according to the following truth table, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs
R	S	CE	J	K	C	Q
1	X	X	X	X	↑	0
0	1	X	X	X	↑	1
0	0	0	X	X	X	No Chg
0	0	1	0	0	X	No Chg
0	0	1	0	1	↑	0
0	0	1	1	0	↑	1
0	0	1	1	1	↑	Toggle

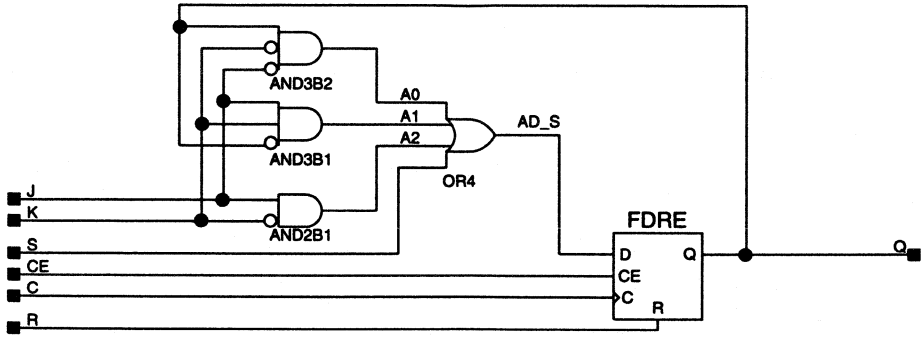
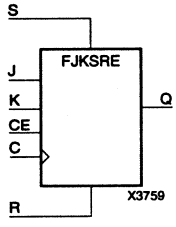


Figure 3-92 FJKRSE XC2000/3000/4000 Implementation

FJKSRE

J-K Flip-Flop with Clock Enable and Synchronous Set and Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FJKSRE is a single J-K-type flip-flop with J, K, synchronous set (S), synchronous reset (R), and clock enable (CE) inputs and data output (Q). When synchronous set (S) is High, all other inputs are ignored and output Q is set High. (Set has precedence over Reset.) When synchronous reset (R) is High and S is Low, output Q is reset Low. When S and R are Low and CE is High, output Q responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs
S	R	CE	J	K	C	Q
1	X	X	X	X	↑	1
0	1	X	X	X	↑	0
0	0	0	X	X	X	No Chg
0	0	1	0	0	X	No Chg
0	0	1	0	1	↑	0
0	0	1	1	0	↑	1
0	0	1	1	1	↑	Toggle

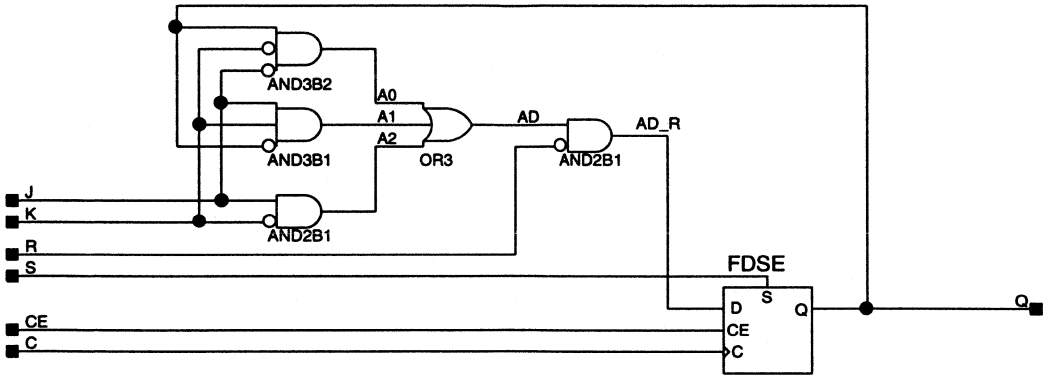
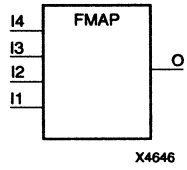


Figure 3-93 FJKSRE XC2000/3000/4000 Implementation

FMAP

F Function Generator Partitioning Control Symbol



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The FMAP symbol is used to control logic partitioning into XC4000 family 4-input function generators. The place and route software chooses an F or a G function generator as a default, unless you specify an F or G. Refer to the appropriate CAE tool interface user guide for information about specifying this attribute in your schematic design editor.

The FMAP symbol is usually used with the HMAP symbol, which partitions logic into the 3-input generator of the Configurable Logic Block (CLB). You can implement a portion of logic using gates, latches, and flip-flops, and specify the logic to be grouped into F, G, and H function generators by naming logic signals and FMAP/HMAP signals correspondingly. These symbols are used for mapping control in addition to the actual gates, latches, and flip-flops, not as a substitute for them.

The following figure gives an example of how logic can be placed using FMAP and HMAP symbols.

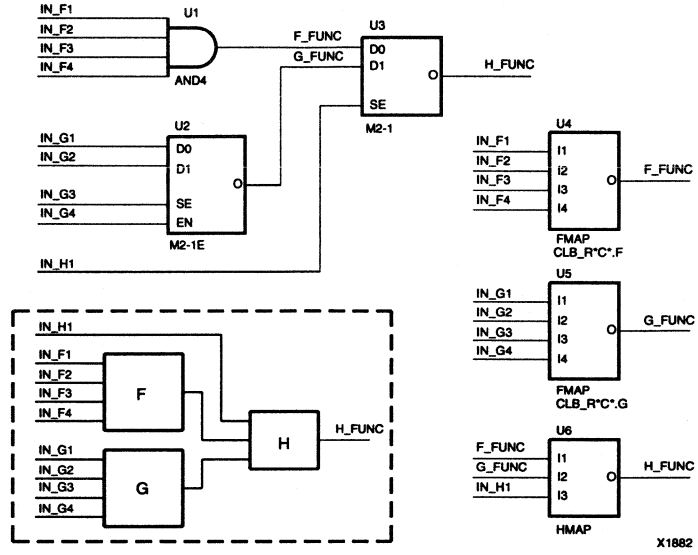


Figure 3-94 Partitioning Logic Using FMAP and HMAP Symbols

The MAP=*type* parameter can be used with the FMAP symbol to further define how much latitude you want to give the mapping program. The following table shows MAP option characters and their meanings.

Character	Function
P	Pins
C	Closed – Adding logic to or removing logic from the CLB is not allowed.
L	Locked – Locking CLB pins
O	Open – Adding logic to or removing logic from the CLB is allowed.
U	Unlocked – No locking on CLB pins.

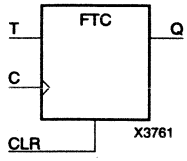
Possible types of MAP parameters for FMAP are: MAP=PUC, MAP=PLC, MAP=PLO, and MAP=PUO. The default parameter is

PUC. If one of the "open" parameters is used (PLO or PUO), only the output signals must be specified.

The FMAP symbol can be assigned to specific CLB locations using LOC attributes. Refer to the appropriate CAE tool interface user guide for more information on assigning LOC attributes.

FTC

Toggle Flip-Flop with Toggle Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

FTC is a synchronous, resettable toggle flip-flop. The asynchronous clear (CLR) input, when High, overrides all other inputs and resets the data output (Q) Low. The Q output toggles, or changes state, when the toggle enable (T) input is High and CLR is Low during the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
CLR	T	C	Q
1	X	X	0
0	0	X	No Chg
0	1	↑	Toggle

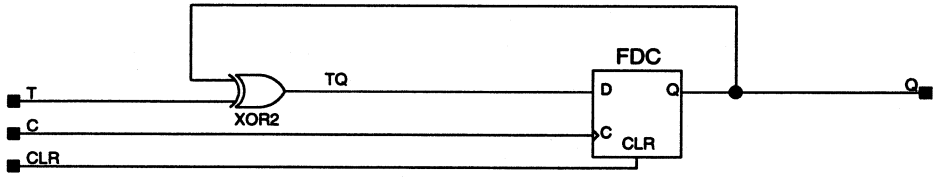
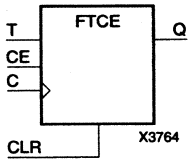


Figure 3-95 FTC XC2000/3000/4000 Implementation

FTCE

Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the asynchronous clear (CLR) input is High, all other inputs are ignored and the data output (Q) is reset Low. When CLR is Low and toggle enable (T) and clock enable (CE) are High, Q output toggles, or changes state, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs
CLR	CE	T	C	Q
1	X	X	X	0
0	0	X	X	No Chg
0	1	0	X	No Chg
0	1	1	↑	Toggle

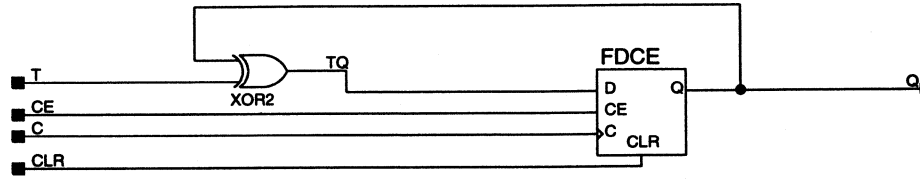
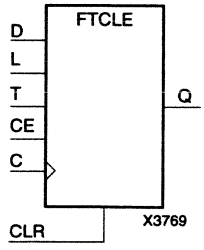


Figure 3-96 FTCE XC2000/3000/4000 Implementation

FTCLE

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the asynchronous clear input (CLR) is High, all other inputs are ignored and output Q is reset Low. When load enable input (L) is High and CLR is Low, clock enable (CE) is overridden and the data on data input (D) is loaded into the flip-flop during the Low-to-High clock (C) transition. When toggle enable (T) and CE are High and L and CLR are Low, output Q toggles, or changes state, during the Low- to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs
CLR	L	CE	T	D	C	Q
1	X	X	X	X	X	0
0	1	X	X	1	↑	1
0	1	X	X	0	↑	0
0	0	0	X	X	X	No Chg
0	0	1	0	X	X	No Chg
0	0	1	1	X	↑	Toggle

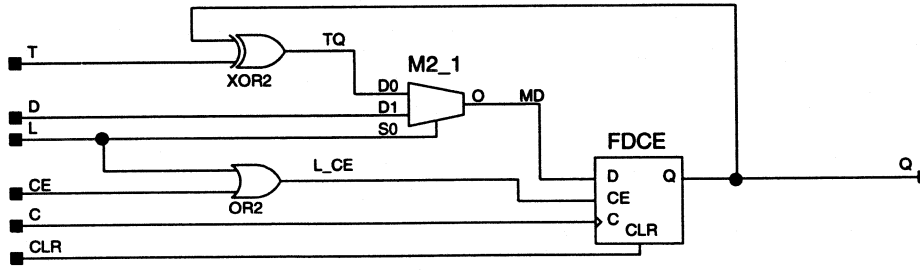
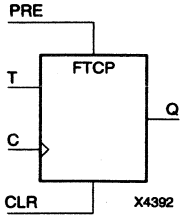


Figure 3-97 FTCLE XC2000/3000/4000 Implementation

FTCP

Toggle Flip-Flop with Toggle Enable and Asynchronous Clear and Preset



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

When the asynchronous clear (CLR) input is High, all other inputs are ignored and the output (Q) is reset Low. When the asynchronous preset (PRE) input is High (provided CLR is Low), all other inputs are ignored and Q is set High. When the toggle enable input (T) is High and CLR and PRE are Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR) is active (Low).

Inputs				Outputs
CLR	PRE	T	C	Q
1	X	X	X	0
0	1	X	X	1
0	0	0	X	No Chg
0	0	1	↑	Toggle

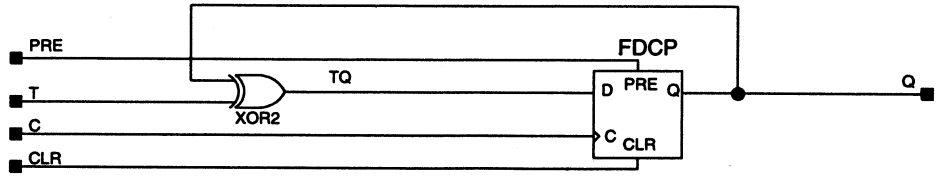
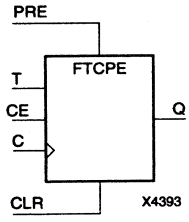


Figure 3-98 FTCP XC2000 Implementation

FTCPE

Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear and Preset



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

When the asynchronous clear (CLR) input is High, all other inputs are ignored and the output (Q) is reset Low. When the asynchronous preset (PRE) input is High (provided CLR is Low), all other inputs are ignored and Q is set High. When the toggle enable input (T) and the clock enable input (CE) are High and CLR and PRE are Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition. Clock transitions are ignored when CE is Low.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR) is active (Low).

Inputs					Outputs
CLR	PRE	CE	T	C	Q
1	X	X	X	X	0
0	1	X	X	X	1
0	0	0	X	X	No Chg
0	0	1	0	X	No Chg
0	0	1	1	↑	Toggle

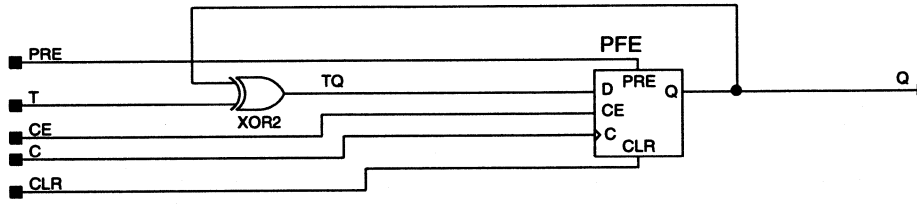
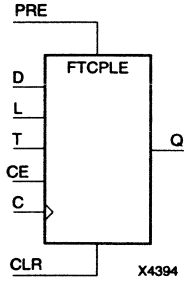


Figure 3-99 FTCPE XC2000 Implementation

FTCPLE

Loadable Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear and Preset



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	Primitive*

* not supported for XC7336 designs

When the asynchronous clear (CLR) input is High, all other inputs are ignored and the output (Q) is reset Low. When the asynchronous preset (PRE) input is High (provided CLR is Low), all other inputs are ignored and Q is set High. The load input (L) loads the data on input D into the flip-flop on the Low-to-High clock transition, regardless of the state of the clock enable (CE). When the toggle enable input (T) and the clock enable input (CE) are High and CLR, PRE, and L are Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition. Clock transitions are ignored when CE is Low.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR) is active (Low).

Inputs							Outputs
CLR	PRE	L	CE	T	C	D	Q
1	X	X	X	X	X	X	0
0	1	X	X	X	X	X	1
0	0	1	X	X	↑	0	0
0	0	1	X	X	↑	1	1
0	0	0	0	X	X	X	No Chg
0	0	0	1	0	X	X	No Chg
0	0	0	1	1	↑	X	Toggle

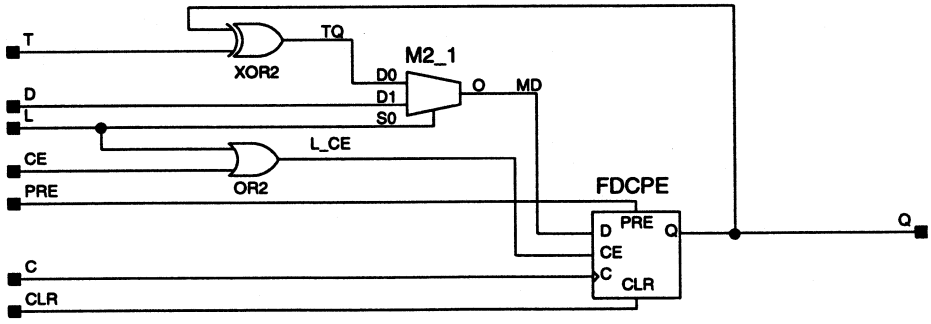
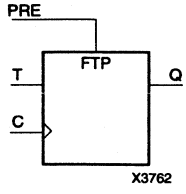


Figure 3-100 FTCPLE XC2000 Implementation

FTP

Toggle Flip-Flop with Toggle Enable and Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	Primitive

When the asynchronous preset (PRE) input is High, all other inputs are ignored and output Q is set High. When toggle enable input (T) is High and PRE is Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs			Outputs
PRE	**T**	**C**	**Q**
1	X	X	1
0	0	X	No Chg
0	1	↑	Toggle

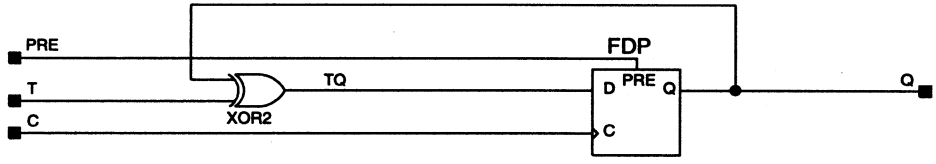
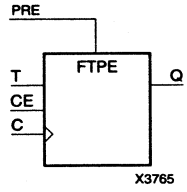


Figure 3-101 FTP XC4000 Implementation

FTPE

Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	Primitive

When the asynchronous preset (PRE) input is High, all other inputs are ignored and output Q is set High during the Low-to-High clock (C) transition. When the toggle enable input (T) is High, clock enable (CE) is High, and PRE is Low, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs				Outputs
PRE	CE	T	C	Q
1	X	X	X	1
0	0	X	X	No Chg
0	1	0	X	No Chg
0	1	1	↑	Toggle

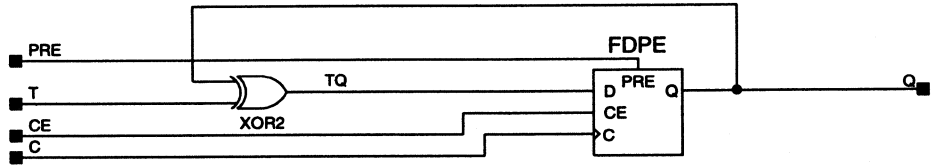
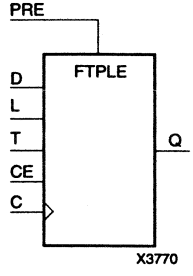


Figure 3-102 FTPE XC4000 Implementation

FTPLE

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Preset



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	Primitive

When the asynchronous preset input (PRE) is High, all other inputs are ignored and output Q is set High during the Low-to-High clock (C) transition. When the load enable input (L) is High and PRE is Low, the clock enable (CE) is overridden and the data on input (D) is loaded into the flip-flop during the Low-to-High clock transition. When L and PRE are Low and toggle enable input (T) and CE are High, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored. The flip-flop is asynchronously set, output High, when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs						Outputs
PRE	L	CE	T	D	C	Q
1	X	X	X	X	X	1
0	1	X	X	1	↑	1
0	1	X	X	0	↑	0
0	0	0	X	X	X	No Chg
0	0	1	0	X	X	No Chg
0	0	1	1	X	↑	Toggle

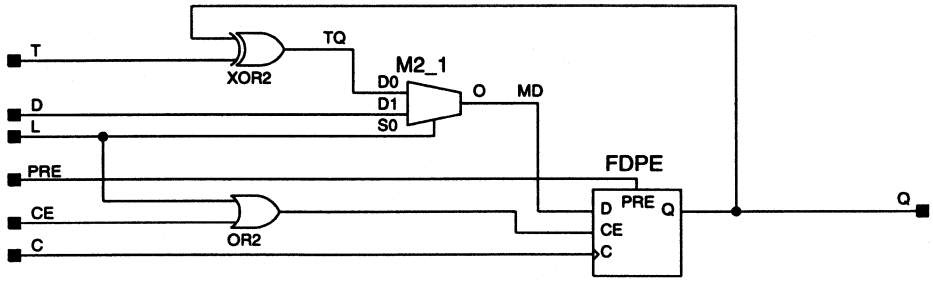
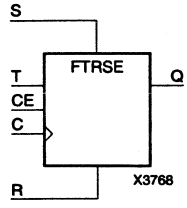


Figure 3-103 FTPLE XC4000 Implementation

FTRSE

Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the synchronous reset input (R) is High, it overrides all other inputs and the data output (Q) is reset Low. When the synchronous set input (S) is High and R is Low, clock enable input (CE) is overridden and output Q is set High. (Reset has precedence over Set.) When toggle enable input (T) and CE are High and R and S are Low, output Q toggles, or changes state, during the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs					Outputs
R	S	CE	T	C	Q
1	X	X	X	↑	0
0	1	X	X	↑	1
0	0	0	X	X	No Chg
0	0	1	0	X	No Chg
0	0	1	1	↑	Toggle

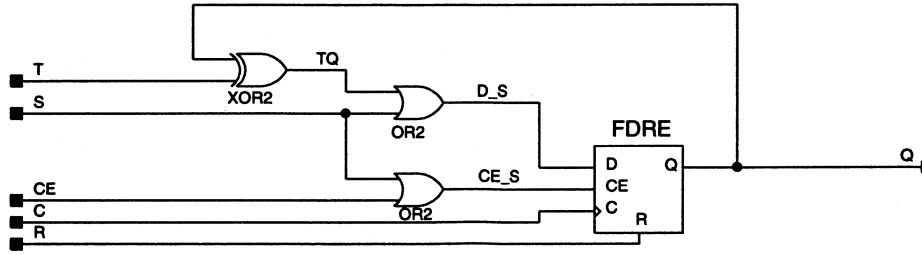
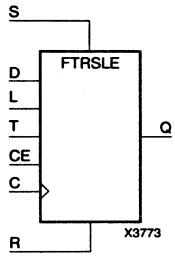


Figure 3-104 FTRSE XC2000/3000/4000 Implementation

FTRSLE

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The synchronous reset input (R), when High, overrides all other inputs and resets the data output (Q) Low. (Reset has precedence over Set.) When R is Low and synchronous set input (S) is High, the clock enable input (CE) is overridden and output Q is set High. When R and S are Low and load enable input (L) is High, CE is overridden and data on data input (D) is loaded into the flip-flop during the Low-to-High clock transition. When R, S, and L are Low and CE is High, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs							Outputs
R	S	L	CE	T	D	C	Q
1	0	X	X	X	X	↑	0
0	1	X	X	X	X	↑	1
0	0	1	X	X	1	↑	1
0	0	1	X	X	0	↑	0
0	0	0	0	X	X	X	No Chg
0	0	0	1	0	X	X	No Chg
0	0	0	1	1	X	↑	Toggle

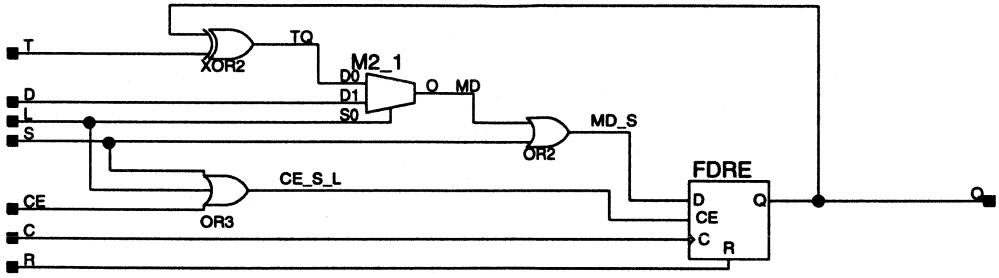
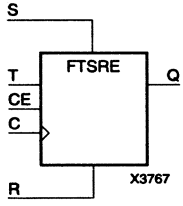


Figure 3-105 FTRSLE XC2000/3000/4000 Implementation

FTSRE

Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Set and Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The synchronous set input, when High, overrides all other inputs and sets data output (Q) High. (Set has precedence over Reset.) When synchronous reset input (R) is High and S is Low, clock enable input (CE) is overridden and output Q is reset Low. When toggle enable input (T) and CE are High and S and R are Low, output Q toggles, or changes state, during the Low-to-High clock transition.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs					Outputs
S	R	CE	T	C	Q
1	X	X	X	↑	1
0	1	X	X	↑	0
0	0	0	X	X	No Chg
0	0	1	0	X	No Chg
0	0	1	1	↑	Toggle

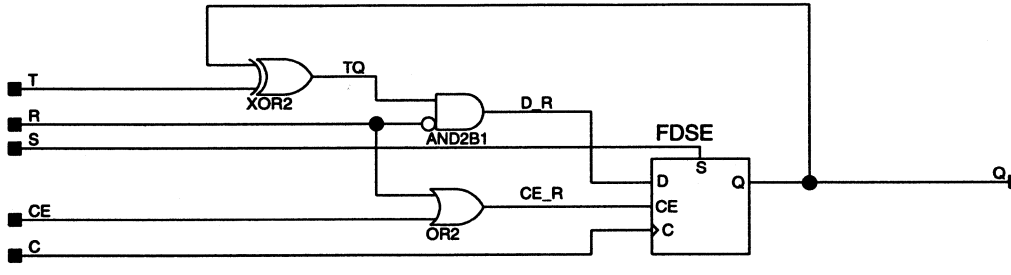
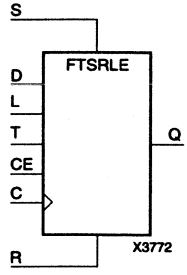


Figure 3-106 FTSRE XC2000/3000/4000 Implementation

FTSRLE

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Set and Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The synchronous set input (S), when High, overrides all other inputs and sets data output (Q) High. (Set has precedence over Reset.) When synchronous reset (R) is High and S is Low, clock enable input (CE) is overridden and output Q is reset Low. When load enable input (L) is High and S and R are Low, CE is overridden and data on data input (D) is loaded into the flip-flop during the Low-to-High clock transition. When the toggle enable input (T) and CE are High and S, R, and L are Low, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs							Outputs
S	R	L	CE	T	D	C	Q
1	0	X	X	X	X	↑	1
0	1	X	X	X	X	↑	0
0	0	1	X	X	1	↑	1
0	0	1	X	X	0	↑	0
0	0	0	0	X	X	X	No Chg
0	0	0	1	0	X	X	No Chg
0	0	0	1	1	X	↑	Toggle

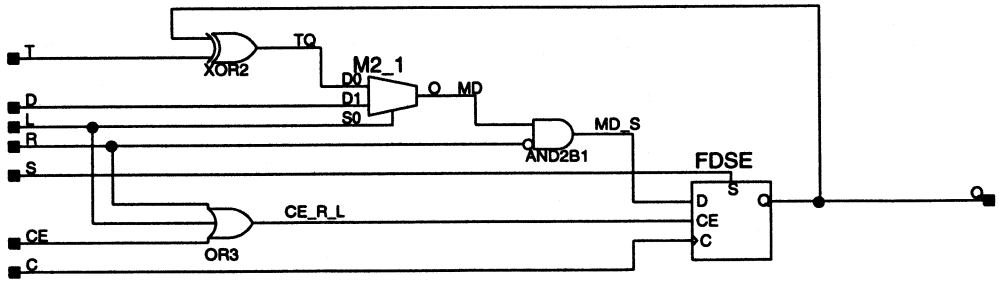


Figure 3-107 FTSRLE XC2000/3000/4000 Implementation

GCLK

Global Clock Buffer

XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	N/A	N/A

GCLK, the global clock buffer, distributes high fan-out clock signals. One GCLK buffer on each device provides direct access to every Configurable Logic Block (CLB) and Input Output Block (IOB) clock pin. If it is not used in a design, its routing resources are not used for any signals. Therefore, the GCLK should always be used for the highest fan–out clock net in the design. The GCLK input (I) can come from one of the following sources.

- From a CMOS-level signal on the dedicated TCLKIN pin (XC3000 only). TCLKIN is a direct CMOS-only input to the GCLK buffer. To use the TCLKIN pin, connect the input of the GCLK element directly to the PAD element (without using an IBUF in between).
- From a CMOS or TTL-level external signal. To connect an external input to the GCLK buffer, connect the input of the GCLK element to the output of the IBUF for that signal. Unless the corresponding PAD element is constrained otherwise, APR or PPR typically places that IOB directly adjacent to the GCLK buffer.
- From an internal signal. To drive the GCLK buffer with an internal signal, connect that signal directly to the input of the GCLK element.

The output of the GCLK buffer can drive all the clock inputs on the chip, but it cannot drive non-clock inputs. For a negative-edge clock, insert an INV (inverter) element between the GCLK output and the clock input. This inversion is performed inside the CLB, or in the case of IOB clock pins, on the IOB clock line (which controls the clock sense for the IOBs on an entire edge of the chip).

GND

Ground-Connection Signal Tag

X3858

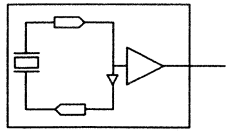
XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	Primitive	Primitive

The GND signal tag, or parameter, forces a net or input function to a Low logic level. A net tied to GND cannot have any other source.

When the logic-trimming software (XNFPrep) or fitter (XEPLD) encounters a net or input function tied to GND, it removes any logic that is disabled by the GND signal. The GND signal is only implemented when the disabled logic cannot be removed.

GXTL

Crystal Oscillator with ACLK Buffer

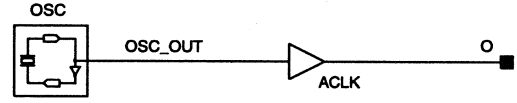


X3886

XC2000	XC3000	XC4000	XC7000
Macro	Macro	N/A	N/A

The GXTL element drives an internal ACLK buffer with a frequency derived from an external crystal-controlled oscillator. The GXTL (or ACLK) output is connected to an internal clock net.

There are two dedicated input pins (XTAL 1 and XTAL 2) on each FPGA device that are internally connected to pads and input/output blocks that are in turn connected to the GXTL amplifier. The external components are connected as shown in the following example. Refer to *The Programmable Gate Array Data Book* for details on component selection and tolerances.

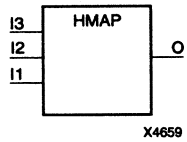


@PULSELO=@PULSEL
@PULSEHI=@PULSEH

Figure 3-108 GXTL XC2000/3000 Implementation

HMAP

H Function Generator Partitioning Control Symbol



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The HMAP symbol is used to control logic partitioning into XC4000 family 3-input H function generators. It is usually used with FMAP, which partitions logic into F and G function generators. You can implement a portion of logic using gates, latches, and flip-flops and specify the logic to be grouped into F, G, and H function generators by naming logic signals and HMAP/FMAP signals correspondingly. These symbols are used for mapping control in addition to the actual gates, latches, and flip-flops and not as a substitute for them. The following figure gives an example of how logic can be placed using HMAP and FMAP symbols.

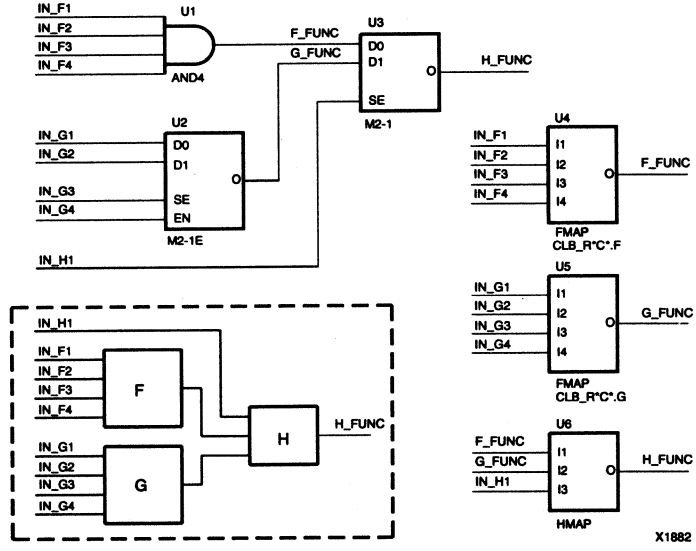


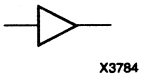
Figure 3-109 Partitioning Logic Using FMAP and HMAP Symbols

The MAP=*type* parameter can only be set to the default value, PUC, for the HMAP symbol. PUC means pins are not locked to the signals but the CLB is closed to addition or removal of logic.

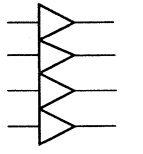
The HMAP symbol can be assigned to specific CLB locations using LOC attributes. Refer to the "Attributes, Constraints, and Carry Logic" chapter for more information on assigning LOC attributes.

IBUF, IBUF4, IBUF8, and IBUF16

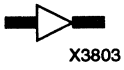
Single- and Multiple-Input Buffers



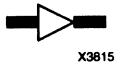
IBUF4



IBUF8



IBUF16



Name	XC2000	XC3000	XC4000	XC7000
IBUF	Primitive	Primitive	Primitive	Primitive
IBUF4, IBUF8, IBUF16	Macro	Macro	Macro	Macro

IBUF, IBUF4, IBUF8, and IBUF16 are single and multiple input buffers. An IBUF isolates the internal circuit from the signals coming into a chip. IBUFs are contained in input/output blocks (IOB). IBUF inputs (I) are connected to an IPAD or an IOPAD. IBUF outputs (O) are connected to the internal circuit.

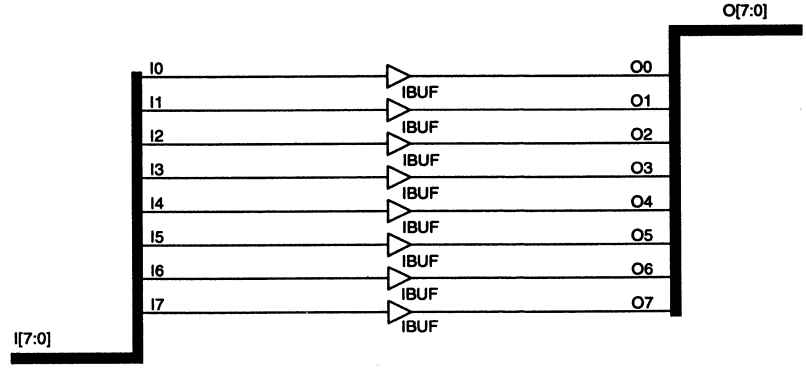
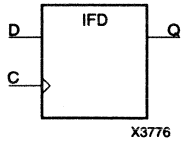
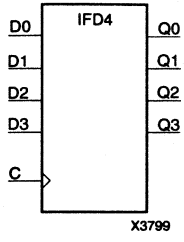


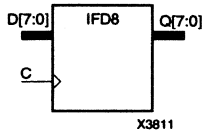
Figure 3-110 IBUF8 XC2000/3000/4000/7000 Implementation

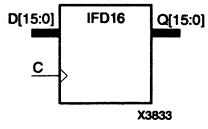
IFD, IFD4, IFD8, and IFD16

Single- and Multiple-Input D Flip-Flops









Name	XC2000	XC3000	XC4000	XC7000
IFD	Primitive	Primitive	Primitive	Primitive*
IFD4, IFD8, IFD16	Macro	Macro	Macro	Macro*

* not supported for XC7336 designs

The IFD D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD (without using an IBUF). The D input provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The clock input is controlled by the internal circuit. For XC7000 EPLDs, the clock (C) can only be driven by a FastCLK represented by the BUFG symbol.

The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. For XC7000 EPLDs (except XC7272), the flip-flops are set High when power is applied.

Refer to the following figures for legal IFD/ILD combinations for XC3000 and XC4000 respectively.

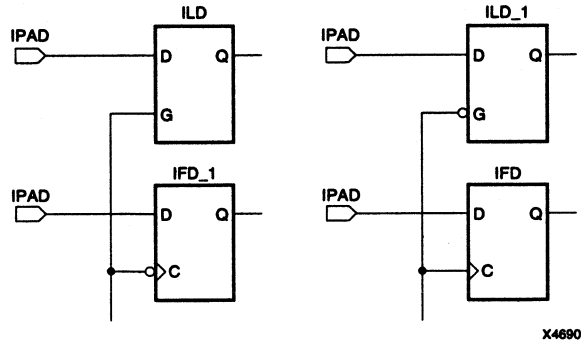


Figure 3-111 Legal Combinations of IFD and ILD for a Single Device Edge of XC3000 IOB

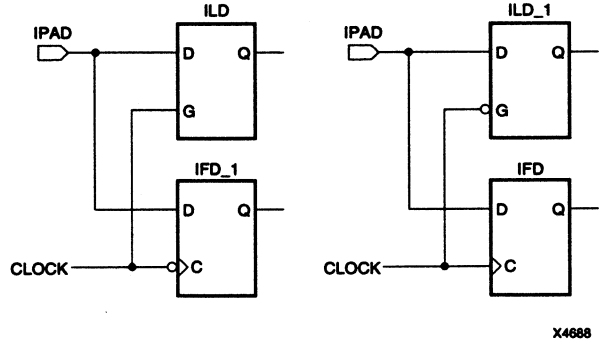


Figure 3-112 Legal Combinations of IFD and ILD for a Single XC4000 IOB

Inputs		Outputs
Dn	**C**	**Qn**
Dn	↑	dn

dn = state of referenced input one set-up time prior to active clock transition

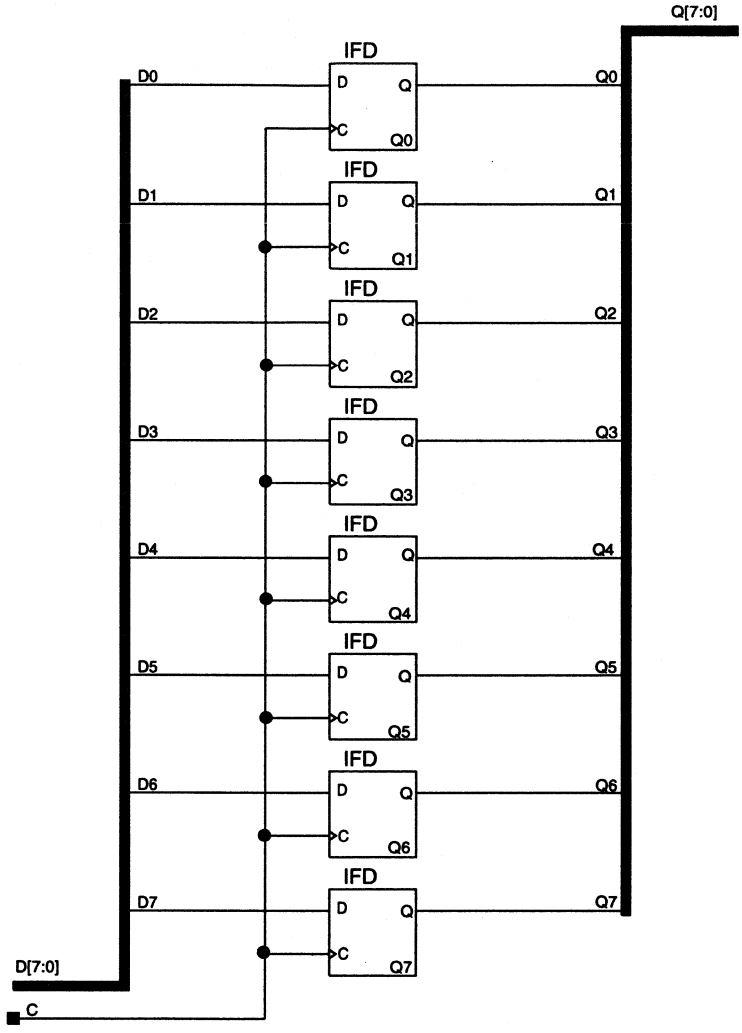
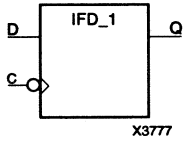


Figure 3-113 IFD8 XC2000/3000/4000/7000 Implementation

IFD_1

Input D Flip-Flop with Inverted Clock



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	N/A

The IFD_1 D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input also provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the High-to-Low clock (C) transition and appears at the output (Q). The clock input is controlled by the internal circuit.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Refer to the following figures for legal IFD/ILD combinations for XC3000 and XC4000 respectively.

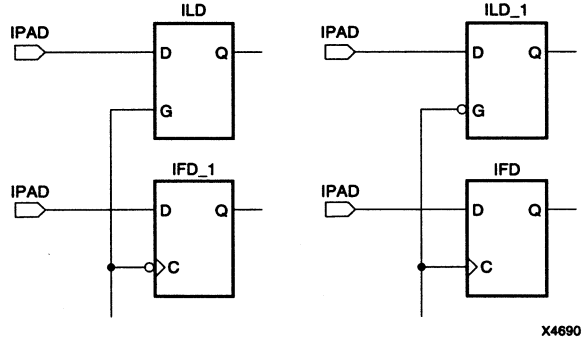


Figure 3-114 Legal Combinations of IFD and ILD for a Single Device Edge of XC3000 IOB

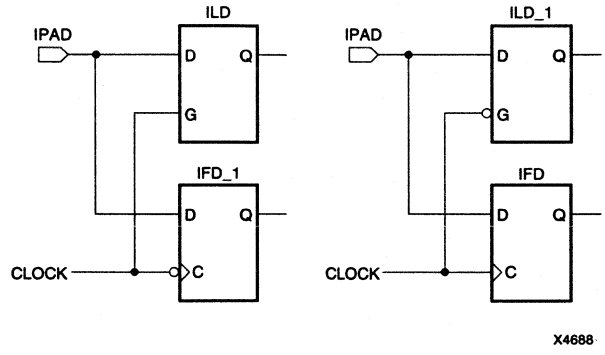


Figure 3-115 Legal Combinations of IFD and ILD for a Single XC4000 IOB

Inputs		Outputs
D	**C**	**Q**
D	↓	d

d = state of referenced input one set-up time prior to active clock transition

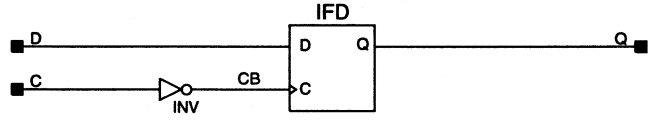
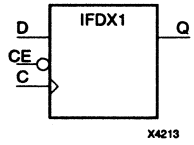
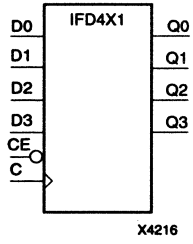


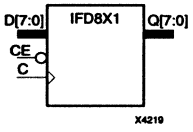
Figure 3-116 IFD_1 XC2000/3000/4000 Implementation

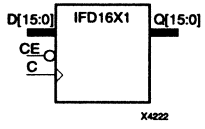
IFDX1, IFD4X1, IFD8X1, and IFD16X1

Input D Flip-Flops for EPLD









Name	XC2000	XC3000	XC4000	XC7000
IFDX1	N/A	N/A	N/A	Primitive*
IFD4X1, IFD8X1, IFD16X1	N/A	N/A	N/A	Macro*

* not supported for XC7236, XC7272, or XC7336 designs

The IFDX1 symbols are D-type flip-flops with synchronous clock enable implemented in the input blocks of an EPLD device. They are commonly used to synchronize and store data entering a chip. The data input (D) of the flip-flop is connected directly to an IPAD or an IOPAD (without using an IBUF). When the clock enable (CE) input is Low, the data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The flip-flop ignores clock transitions when CE is High.

The clock input (C) must be driven by a global FastCLK net of the EPLD device, represented by the BUFG symbol. The clock enable input (CE) must be driven by a global clock enable net of the EPLD device, represented by the BUFCE symbol.

The flip-flops are asynchronously set, outputs High, when power is applied or when the device Master Reset pin is activated.

Inputs			Outputs
D	CE	C	Q
X	1	X	No Chg
0	0	↑	0
1	0	↑	1

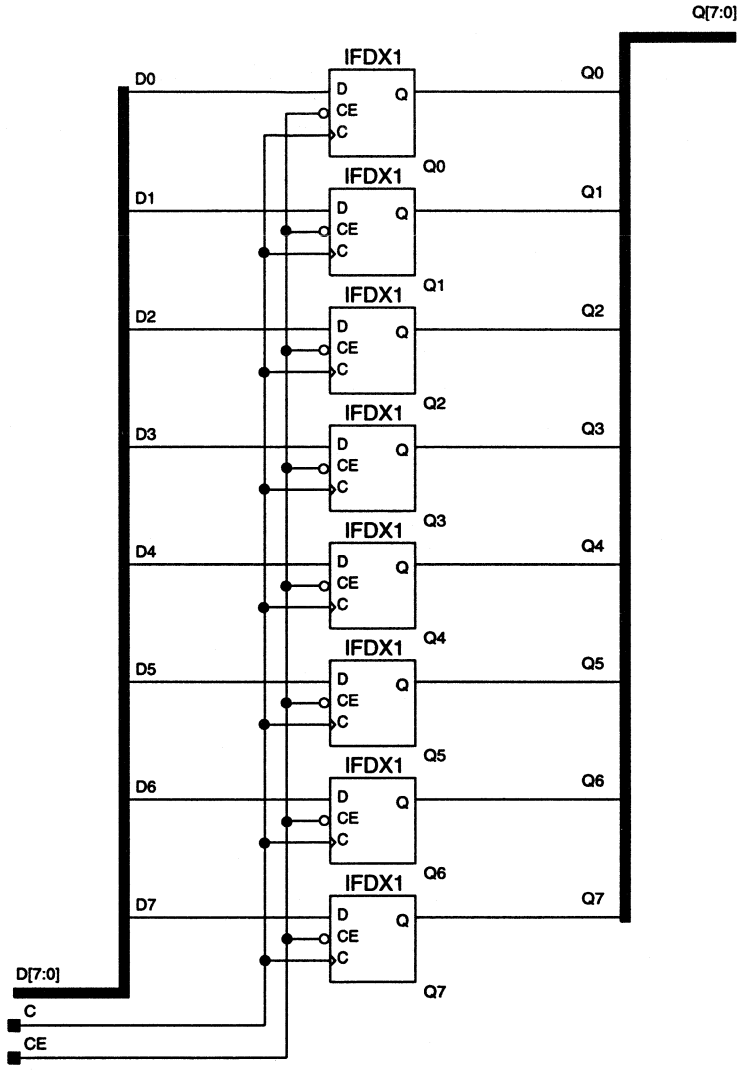
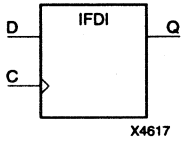


Figure 3-117 IFD8X1 XC7000 Implementation

IFDI

Input D Flip-Flop (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The IFDI D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The clock input is controlled by the internal circuit. The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Refer to the following figures for legal IFDI/ILDI combinations for XC3000 and XC4000 respectively.

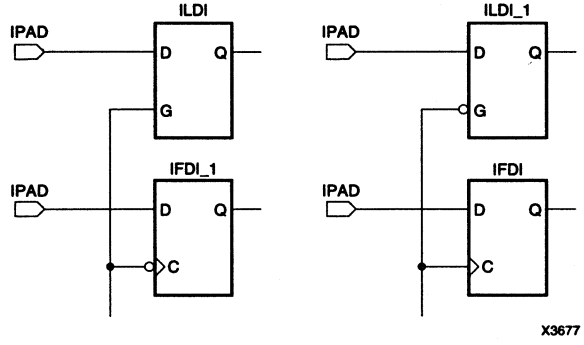


Figure 3-118 Legal Combinations of IFDI and ILDI for a Single Device Edge of XC3000 IOB

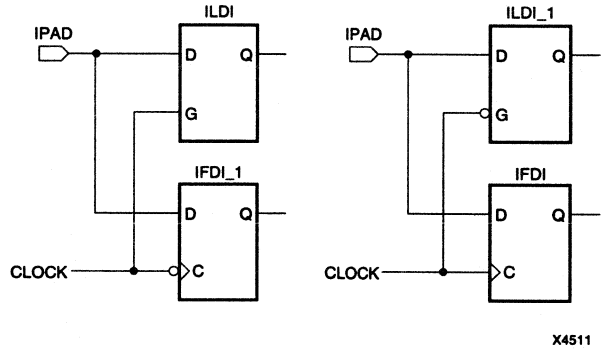


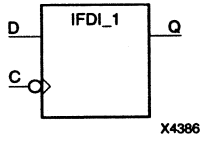
Figure 3-119 Legal Combinations of IFDI and ILDI for a Single XC4000 IOB

Inputs		Outputs
D	**C**	**Q**
D	↑	d

d = state of referenced input one set-up time prior to active clock transition

IFDI_1

D Flip-Flop with Inverted Clock (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

The IFDI_1 D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the High-to-Low clock (C) transition and appears at the output (Q). The clock input is controlled by the internal circuit. The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Refer to the following figures for legal IFDI/ILDI combinations for XC3000 and XC4000 respectively.

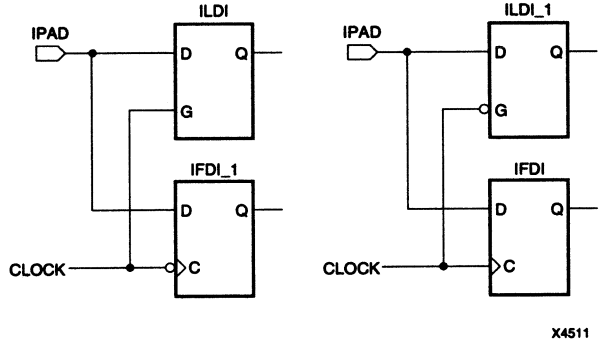


Figure 3-120 Legal Combinations of IFDI and ILDI for a Single XC4000 IOB

Inputs		Outputs
D	C	Q
D	↓	d

d = state of referenced input one set-up time prior to active clock transition

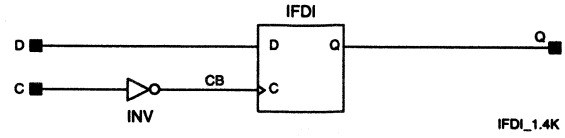
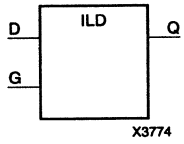


Figure 3-121 IFDI_1 XC4000 Implementation

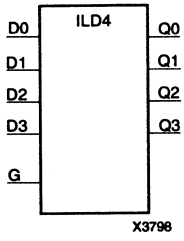
ILD, ILD4, ILD8, and ILD16

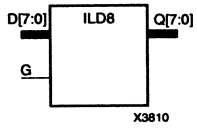
Input Transparent Data Latches

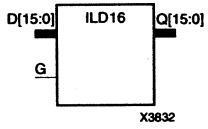


Name	XC2000	XC3000	XC4000	XC7000
ILD	N/A	Primitive	Macro	Primitive*
ILD4, ILD8, ILD16	N/A	Macro	Macro	Macro*

* not supported for XC7336 designs







ILD, ILD4, ILD8, and ILD16 are single or multiple transparent data latches, which can be used to hold transient data entering a chip. The latch input (D) is connected to an IPAD or an IOPAD (without using an IBUF). When the gate input (G) is High, data on the inputs (D) appears on the outputs (Q). Data on the D inputs during the High-to-Low G transition is stored in the latch. For XC7000 EPLDs, the gate input (G) must be driven by a FastCLK, represented by the BUFG symbol.

The latch is reset, output Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable. For XC7000 EPLDs (except XC7272) the latches are set High when power is applied.

XC4000 ILD

The XC4000 ILD is actually the input flip-flop master latch. It is possible to access two different outputs from the input flip-flop: one that responds to the level of the clock signal and another that responds to an edge of the clock signal. When using both outputs from the same input flip-flop, a transparent High latch (ILD) corresponds to a falling edge-triggered flip-flop (IFD_1). Similarly, a transparent Low latch (ILD_1) corresponds to a rising edge-triggered flip-flop (IFD). Refer to the following figure for XC4000 legal IFD/ILD combinations.

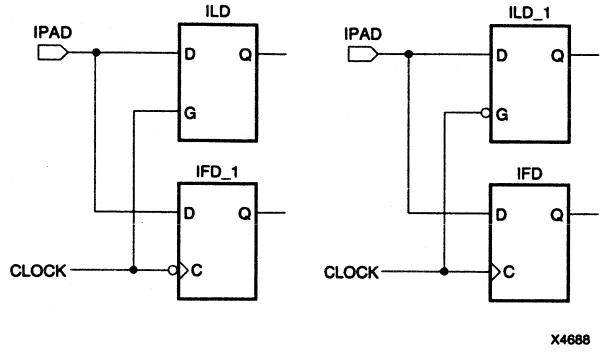


Figure 3-122 Legal Combinations of IFD and ILD for a Single XC4000 IOB

XC3000 ILD

The XC3000 ILD is actually the input flip-flop master latch. If both ILD and IFD elements are controlled by the same clock signal, the relationship between the transparent sense of the latch and the active edge of the flip-flop is fixed as follows: a transparent High latch (ILD) corresponds to a falling edge-triggered flip-flop (IFD_1), and a transparent Low latch (ILD_1) corresponds to a rising edge-triggered flip-flop (IFD). Because the place and route software does not support using both phases of a clock for IOBs on a single edge of the device, certain combinations of ILD and IFD elements are not allowed. Refer to the following figure for XC3000 legal IFD/ILD combinations.

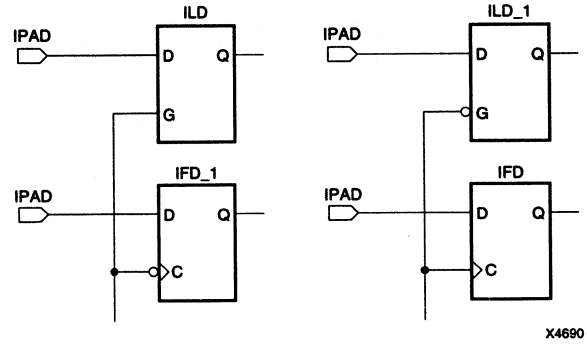


Figure 3-123 Legal Combinations of IFD and ILD for a Single Device Edge of XC3000 IOB

Inputs		Outputs
G	**D**	**Q**
1	1	1
1	0	0
↓	D	d

d = state of referenced input one set-up time prior to High-to-Low gate transition

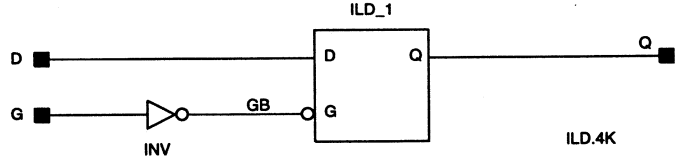


Figure 3-124 ILD XC4000 Implementation

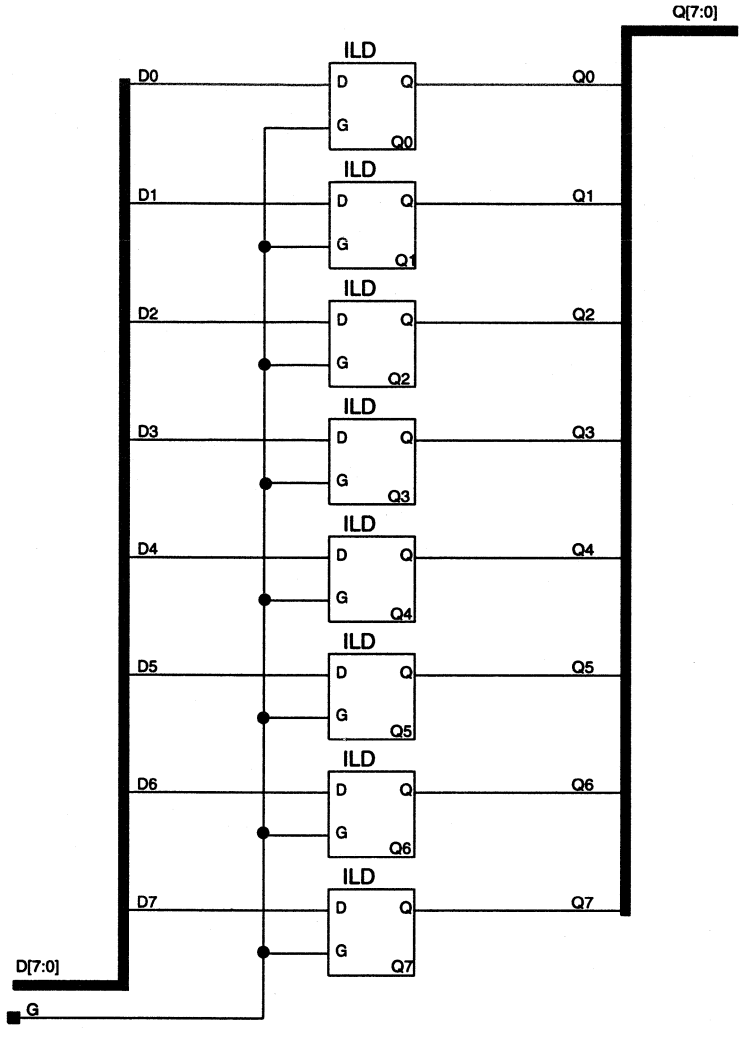
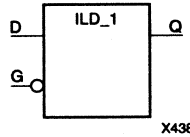


Figure 3-125 ILD8 XC3000/4000/7000 Implementation

ILD_1

Transparent Input Data Latch with Inverted Gate



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Primitive	N/A

ILD_1 is a transparent data latch, which can be used to hold transient data entering a chip. When the gate input (G) is Low, data on the data input (D) appears on the data output (Q). Data on D during the Low-to-High G transition is stored in the latch. For implementation details, refer to the "ILD, ILD4, ILD8, and ILD16" section earlier in this chapter.

The latch is reset, output Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Refer to the following figures for legal IFD/ILD combinations, for XC3000 and XC4000 respectively.

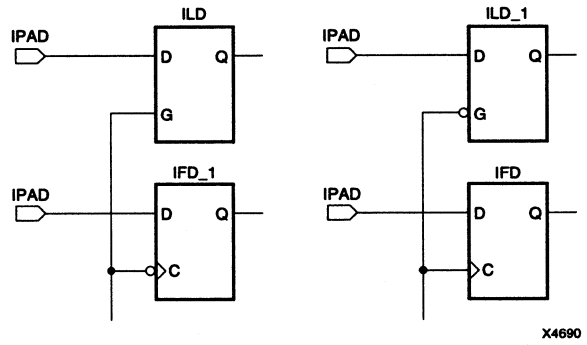


Figure 3-126 Legal Combinations of IFD and ILD for a Single Device Edge of XC3000 IOB

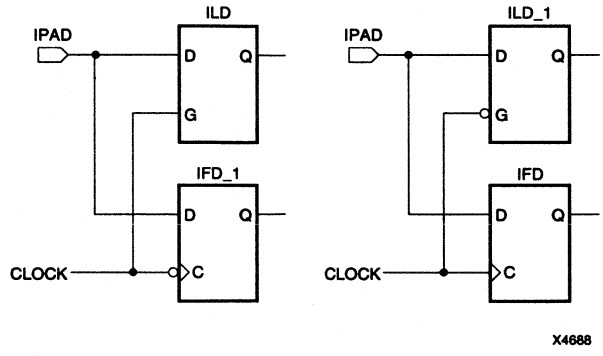


Figure 3-127 Legal Combinations of IFD and ILD for a Single XC4000 IOB

Inputs		Outputs
G	**D**	**Q**
0	1	1
0	0	0
↑	D	d

d = state of referenced input one set-up time prior to Low-to-High gate transition

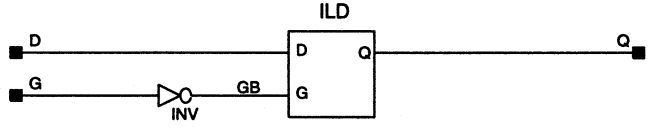
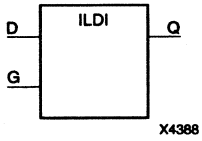


Figure 3-128 ILD_1 XC3000 Implementation

ILDI

Input Transparent Data Latch (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

ILDI is a transparent data latch, which can hold transient data entering a chip. When the gate input (G) is High, data on the input (D) appears on the output (Q). Data on the D input during the High-to-Low G transition is stored in the latch.

The ILDI is actually the input flip-flop master latch. It is possible to access two different outputs from the input flip-flop: one that responds to the level of the clock signal and another that responds to an edge of the clock signal. When using both outputs from the same input flip-flop, a transparent High latch (ILDI) corresponds to a falling edge-triggered flip-flop (IFDI_1). Similarly, a transparent Low latch (ILDI_1) corresponds to a rising edge-triggered flip-flop (IFDI). Refer to the following figures for legal IFDI/ILDI combinations for XC3000 and XC4000 respectively.

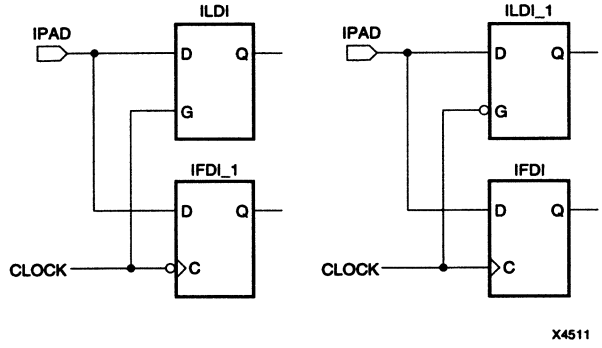


Figure 3-129 Legal Combinations of IFDI and ILDI for a Single XC4000 IOB

The latch is set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs		Outputs
G	**D**	**Q**
1	1	1
1	0	0
↓	D	d

d = state of referenced input one set-up time prior to High-to-Low gate transition

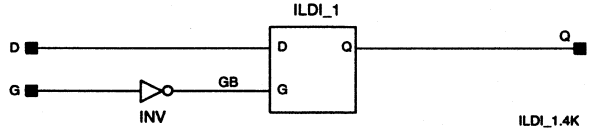
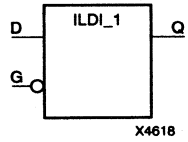


Figure 3-130 ILDI XC4000 Implementation

ILDI_1

Transparent Input Data Latch with Inverted Gate (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

ILDI_1 is a transparent data latch, which can hold transient data entering a chip. When the gate input (G) is Low, data on the data input (D) appears on the data output (Q). Data on D during the Low-to-High G transition is stored in the latch. For implementation details, refer to the "ILD, ILD4, ILD8, and ILD16" section earlier in this chapter.

The latch is set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Refer to the following figures for legal IFDI/ILDI combinations for XC3000 and XC4000 respectively.

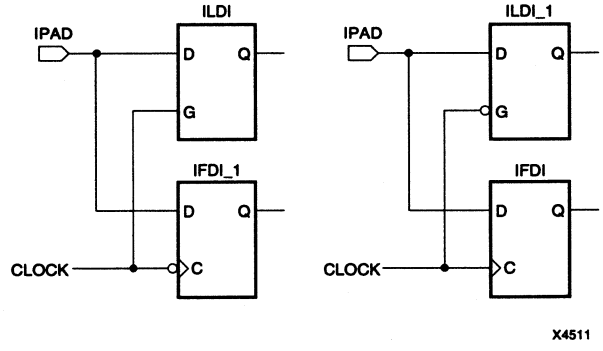


Figure 3-131 Legal Combinations of IFDI and ILDI for a Single XC4000 IOB

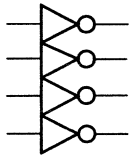
Inputs		Outputs
G	**D**	**Q**
0	1	1
0	0	0
↑	D	d

d = state of referenced input one set-up time prior to Low-to-High gate transition

INV, INV4, INV8, and INV16

Single and Multiple Inverters


X3788


X3795


X3807


X3819

Name	XC2000	XC3000	XC4000	XC7000
INV	Primitive	Primitive	Primitive	Primitive
INV4, INV8, INV16	Macro	Macro	Macro	Primitive

These single and multiple inverters identify signal inversions in a schematic.

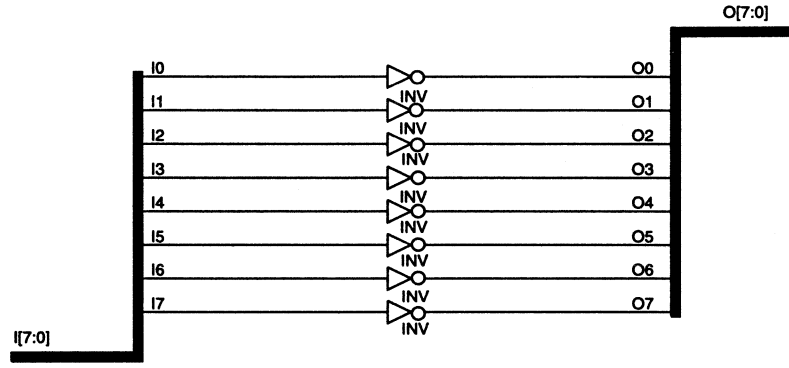


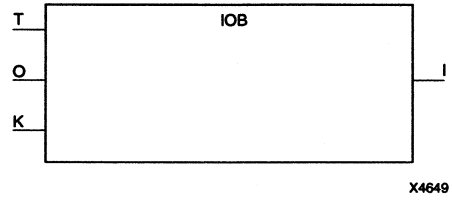
Figure 3-132 INV8 XC2000/3000/4000 Implementation

IOB

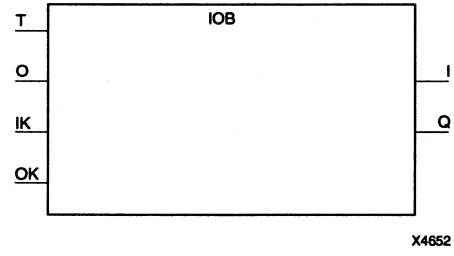
IOB Configuration Symbol

XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	N/A	N/A

The IOB symbol is used to manually specify an IOB configuration. Use it in place of, not in conjunction with, other I/O primitives. The configuration of the IOB is specified using the BASE and CONFIG commands. Enter these commands on the schematic; the translator puts them into the CFG records in the LCA Xilinx netlist file. It is not necessary for the translator program to parse the commands specifying the IOB configuration. The mapping program from the LCA Xilinx netlist to the FPGA design checks these commands for errors.



XC2000



XC3000

Refer to the appropriate CAE tool interface user guide for more information on specifying the IOB configuration commands in a schematic.

The XC2000 blank IOB primitive symbol and its corresponding configured IOB primitive and circuit are shown in the following figure.

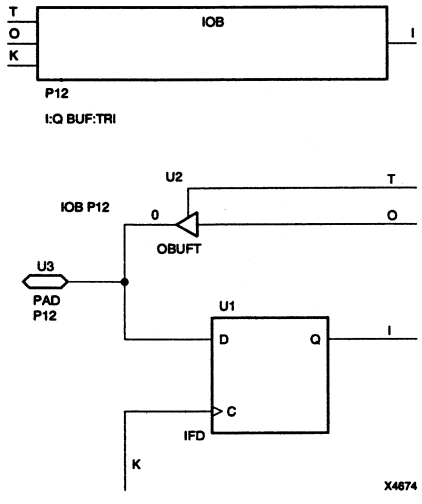


Figure 3-133 XC2000 IOB Primitive Example and Equivalent Circuit

The XC3000 blank IOB primitive symbol and its corresponding configured IOB primitive and circuit are shown in the following figure.

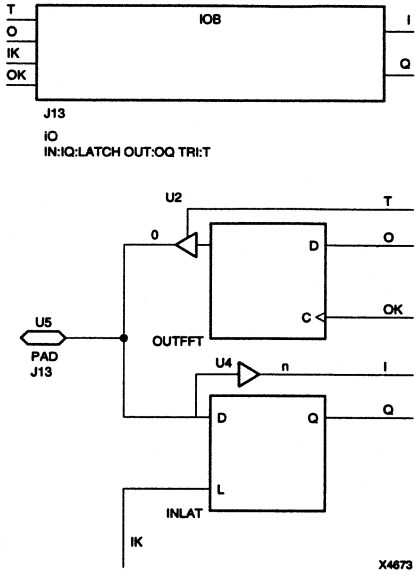


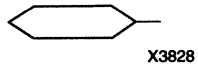
Figure 3-134 XC3000 IOB Primitive Example and Equivalent Circuit

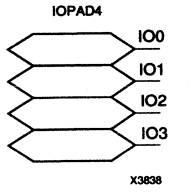
The configuration commands must be consistent with the connections to the pins on the symbol. For example, if the configuration commands specify the IOB as a 3-state buffer, the T and O pins must be connected to signals.

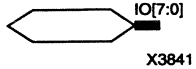
You can specify the location of the IOB on the device. When specifying the LOC attribute, a valid IOB location name must be used. Refer to the "Attributes, Constraints, and Carry Logic" chapter for more information the LOC attribute.

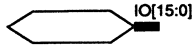
IOPAD, IOPAD4, IOPAD8, and IOPAD16

Input/Output Pads









Name	XC2000	XC3000	XC4000	XC7000
IOPAD	Primitive	Primitive	Primitive	Primitive
IOPD4, IOPAD8, IOPAD16	Macro	Macro	Macro	Macro

IOPAD, IOPAD4, IOPAD8, and IOPAD16 are single and multiple input/output pads. The IOPAD is a connection point from a device pin, used as a bidirectional signal, to a PLD device. The IOPAD is connected internally to an input/output block (IOB), which is configured by the XACT software as a bidirectional block. Bidirectional blocks can consist of any combinations of a 3-state output buffer (such as OBUFT or OFDE) and any available input buffer (such as IBUF or IFD). Refer to the appropriate CAE tool interface user guide for details on assigning pin location and identification.

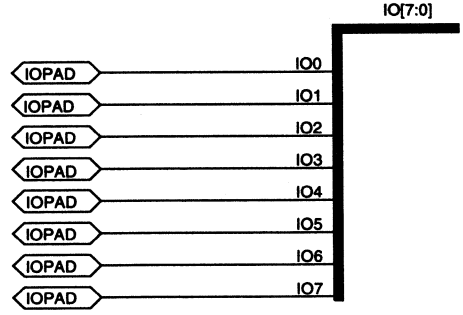
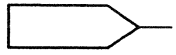
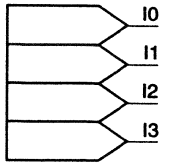


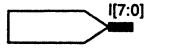
Figure 3-135 IOPAD8 XC2000/3000/4000/7000 Implementation

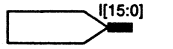
IPAD

Single- and Multiple-Input Pads









Name	XC2000	XC3000	XC4000	XC7000
IPAD	Primitive	Primitive	Primitive	Primitive
IPAD4, IPAD8, IPAD16	Macro	Macro	Macro	Macro

IPAD, IPAD4, IPAD8, and IPAD16 are single and multiple input pads (IPADs). The IPAD is a connection point from a device pin used for an input signal to the PLD device. It is connected internally to an input/output block (IOB), which is configured by the XACT software as an IBUF, IFD or ILD. Refer to the appropriate CAE tool interface user guide for details on assigning pin location and identification.

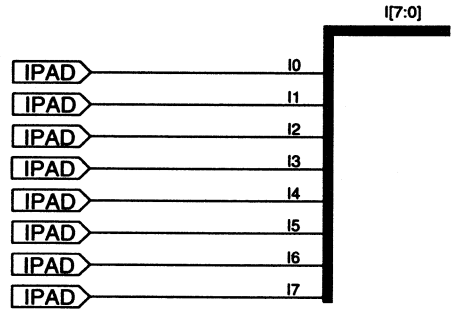
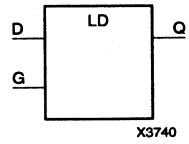
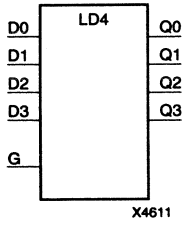


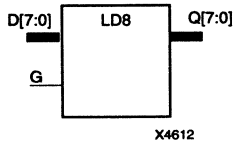
Figure 3-136 IPAD8 XC2000/3000/4000/7000 Implementation

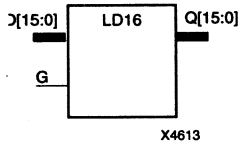
LD, LD4, LD8, and LD16

Single and Multiple Transparent Data Latches









Element	XC2000	XC3000	XC4000	XC7000
LD	Macro	N/A	N/A	Primitive*
LD4, LD8, LD16	N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

The data output (Q) of the latch reflects the data (D) input while the gate enable (G) input is High. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low. LD4, LD8, and LD16 have 4, 8, and 16 transparent latches, respectively, with a common Gate enable (G).

The latch is reset, output Low, when power is applied or when global reset (GR) is active. For EPLD designs, the G input may not be driven by a FastCLK signal (BUFG).

Inputs		Outputs
G	D	Q
1	0	0
1	1	1
0	X	No Chg
↓	D	d

d = state of input one set-up time prior to High-to-Low gate transition

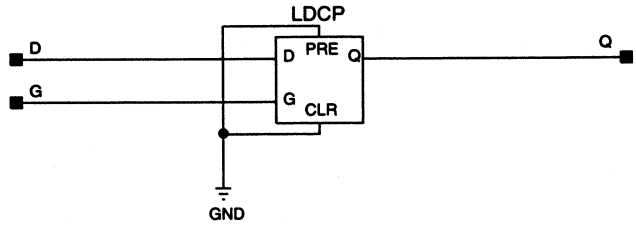
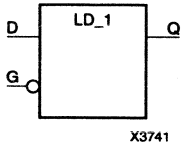


Figure 3-137 LD XC2000 Implementation

LD_1

Transparent Data Latch with Inverted Gate



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	N/A

The data output (Q) of the latch reflects the data (D) input while the gate enable (G) input is Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs		Outputs
G	D	Q
0	0	0
0	1	1
1	X	No Chg
↑	D	d

d = state of input one set-up time prior to Low-to-High gate transition

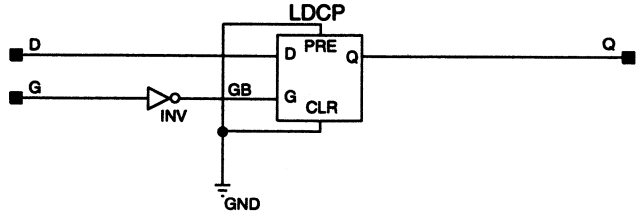
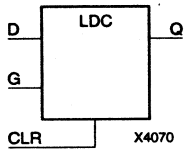


Figure 3-138 LD_1 XC2000 Implementation

LDC

Transparent Data Latch with Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	N/A

When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate enable (G) input is High and CLR is Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs			Outputs
CLR	G	D	Q
1	X	X	0
0	1	1	1
0	1	0	0
0	0	X	No Chg
0	↓	D	d

d = state of input one set-up time prior to High-to-Low gate transition

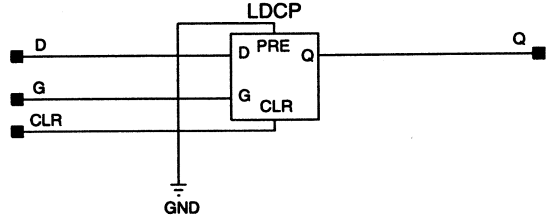
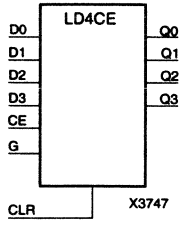
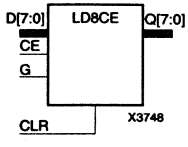


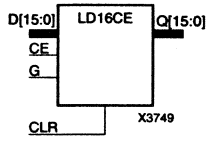
Figure 3-139 LDC XC2000 Implementation

LD4CE, LD8CE, and LD16CE

Transparent Data Latches with Asynchronous Clear and Clock Enable







XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	N/A

LD4CE, LD8CE, and LD16CE have 4, 8, and 16 transparent data latches, respectively. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) outputs Low. Q reflects the data (D) inputs while the gate enable (G) input is High, clock enable (CE) is High, and CLR is Low. If CE is Low, data on D cannot be latched. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or CE remains Low.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs				Outputs
CLR	**CE**	**G**	**Dn**	**Qn**
1	X	X	X	0
0	0	X	X	No Chg
0	1	1	1	1
0	1	1	0	0
0	1	0	X	No Chg
0	1	↓	Dn	dn

Dn = referenced input, for example, D0, D1, D2

Qn = referenced output, for example, Q0, Q1, Q2

dn = referenced input state, one set-up time prior to High-to-Low gate transition

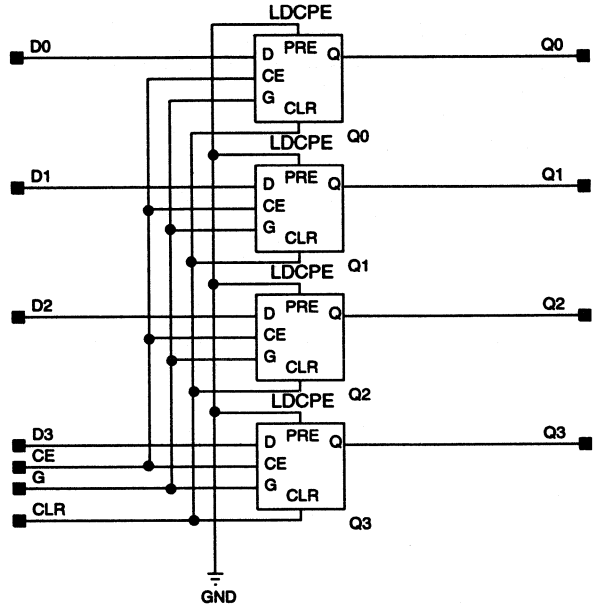


Figure 3-140 LD4CE XC2000 Implementation

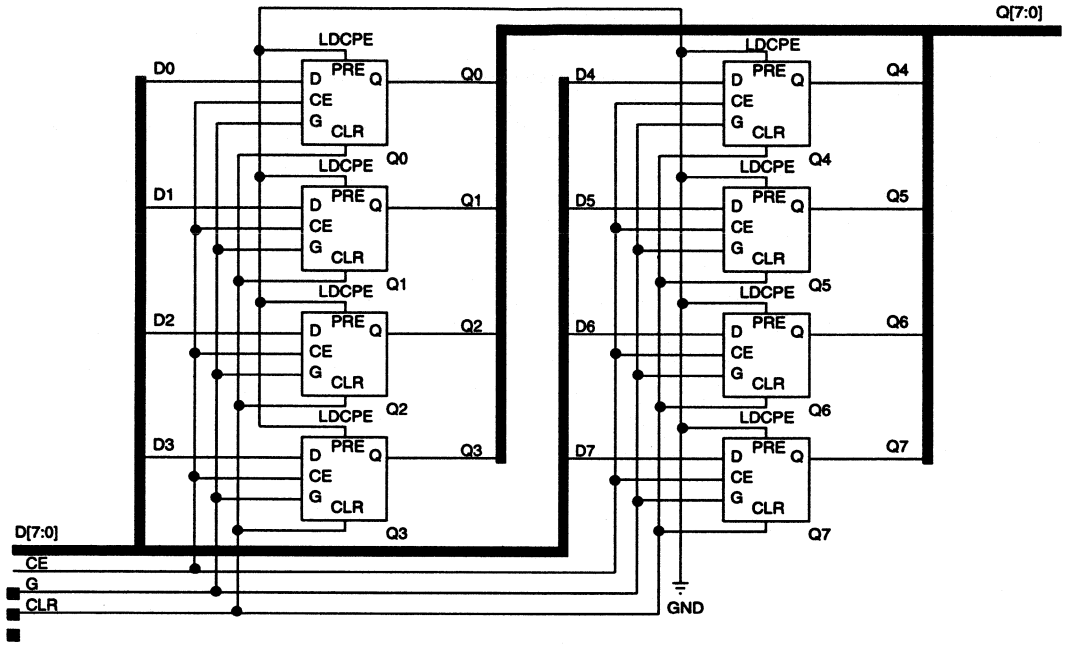
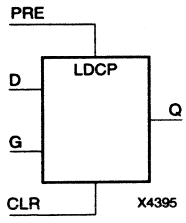


Figure 3-141 LD8CE XC2000 Implementation

LDCP

Transparent Data Latch with Asynchronous Clear and Preset



XC2000	XC3000	XC4000	XC7000
Primitive	N/A	N/A	N/A

When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. When the asynchronous preset (PRE) input is High (and CLR is Low), it sets Q High. Q reflects the data (D) input while the gate enable (G) input is High and CLR and PRE are Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low.

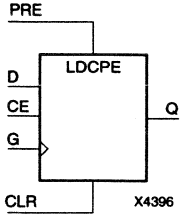
The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs				Outputs
CLR	PRE	G	D	Q
1	X	X	X	0
0	1	X	X	1
0	0	1	0	0
0	0	1	1	1
0	0	0	X	No Chg
0	0	↓	D	d

d = state of input one set-up time prior to High-to-Low gate transition

LDCPE

Transparent Data Latch with Asynchronous Clear and Preset and Clock Enable



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	N/A

When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. When the asynchronous preset (PRE) input is High (and CLR is Low), it sets Q High. Q reflects the data (D) input while the gate enable (G) input and clock enable (CE) are High and CLR and PRE are Low. If CE is Low, data on D cannot be latched. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or CE remains Low.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs					Outputs
CLR	**PRE**	**CE**	**G**	**D**	**Q**
1	X	X	X	X	0
0	1	X	X	X	1
0	0	0	X	X	No Chg
0	0	1	1	1	1
0	0	1	1	0	0
0	0	1	0	X	No Chg
0	0	1	↓	D	d

d = state of input one set-up time prior to High-to-Low gate transition

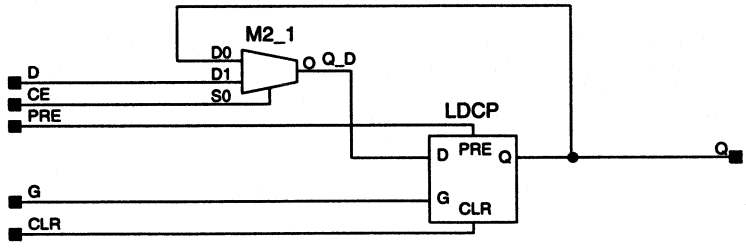
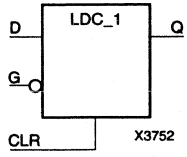


Figure 3-142 LDCPE XC2000 Implementation

LDC_1

Transparent Data Latch with Asynchronous Clear and Inverted Gate Input



XC2000	XC3000	XC4000	XC7000
Macro	N/A	N/A	N/A

When the asynchronous clear input (CLR) is High, it overrides the other inputs (D and G) and resets the data (Q) output Low. Q reflects the data (D) input while the gate enable (G) input and CLR are Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is reset, output Low, when power is applied or when Global Reset (GR) is active.

Inputs			Outputs
CLR	**G**	**D**	**Q**
1	X	X	0
0	0	1	1
0	0	0	0
0	1	X	No Chg
0	↑	D	d

d = state of input one set-up time prior to Low-to-High gate transition

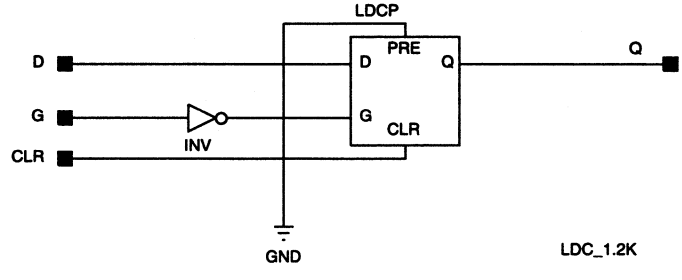
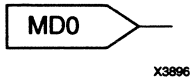


Figure 3-143 LDC_1 XC2000 Implementation

MD0

Mode 0/Input Pad Used for Readback Trigger Input



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The MD0 input pad is connected to the Mode 0 (M0) input pin, which is used to determine the configuration mode on an XC4000 device. Following configuration, MD0 can be used as an input pad, but it must be connected through an IBUF to the user circuit. However, the user input signal must not interfere with the device configuration. The MD0 pad cannot be used as an output pad and the IOB associated with it has no flip-flop or latch. For compatibility with XC2000 and XC3000 devices, this pad is usually connected to the RTRIG input of the READBACK function.

MD1

Mode 1/Output Pad Used for Readback Data Output

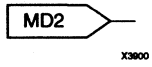
X3896

XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The MD1 input pad is connected to the Mode 1 (M1) input pin, which is used to determine the configuration mode on an XC4000 device. Following configuration, MD1 can be used as a 3-state or simple output pad, but it must be connected through an OBUF or an OBUFT to the user circuit. However, the user output signal must not interfere with the device configuration. An MD1 pad cannot be used as an input pad and the IOB associated with it has no flip-flop or latch. This pad is usually connected to the DATA output of the READBACK function, and the output-enable input of the 3-state OBUFT is connected to the RIP output of the READBACK function.

MD2

Mode 2/Input Pad

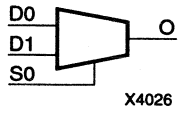


XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The MD2 input pad is connected to the Mode 2 (M2) input pin, which is used to determine the configuration mode on an XC4000 device. Following configuration, MD2 can be used as an input pad, but it must be connected through an IBUF to the user circuit. However, the user input signal must not interfere with the device configuration. An MD2 pad cannot be used as an output pad and the IOB associated with it has no flip-flop or latch.

M2_1

2-to-1 Multiplexer



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The M2_1 multiplexer chooses one data bit from two sources (D1 or D0) under the control of the select input (S0). The output (O) reflects the state of the selected data input. When Low, S0 selects D0 and when High, S0 selects D1.

Inputs			Outputs
S0	D1	D0	O
1	1	X	1
1	0	X	0
0	X	1	1
0	X	0	0

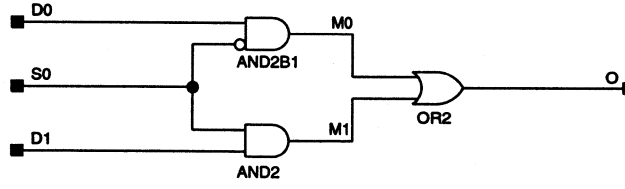
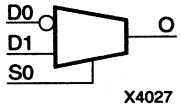


Figure 3-144 M2_1 XC2000/3000/4000 Implementation

M2_1B1

2-to-1 Multiplexer with D0 Inverted



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The M2_1B1 multiplexer chooses one data bit from two sources (D1 or D0) under the control of select input (S0). When S0 is Low, the output (O) reflects the state of $\overline{D0}$. When S0 is High, O reflects the state of D1.

Inputs			Outputs
S0	D1	D0	O
1	1	X	1
1	0	X	0
0	X	1	0
0	X	0	1

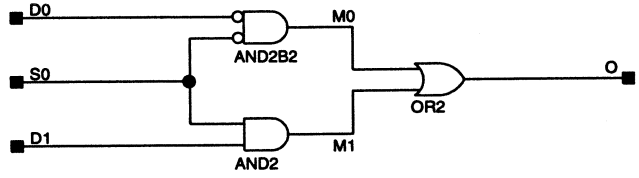
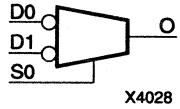


Figure 3-145 M2_1B1 XC2000/3000/4000 Implementation

M2_1B2

2-to-1 Multiplexer with D0 and D1 Inverted



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The M2_1B2 multiplexer chooses one data bit from two sources (D1 or D0) under the control of select input (S0). When S0 is Low, the output (O) reflects the state of $\overline{D0}$. When S0 is High, O reflects the state of $\overline{D1}$.

Inputs			Outputs
S0	D1	D0	O
1	1	X	0
1	0	X	1
0	X	1	0
0	X	0	1

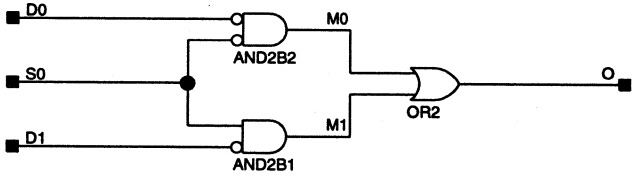
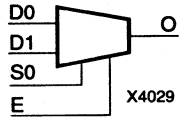


Figure 3-146 M2_1B2 XC2000/3000/4000 Implementation

M2_1E

2-to-1 Multiplexer with Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the enable input (E) is High, the M2_1E chooses one data bit from two sources (D1 or D0) under the control of select input (S0). When E is High, the output (O) reflects the state of the selected input. When Low, S0 selects D0 and when High, S0 selects D1. When E is Low, the output is Low.

Inputs				Outputs
E	S0	D1	D0	O
0	X	X	X	0
1	0	X	1	1
1	0	X	0	0
1	1	1	X	1
1	1	0	X	0

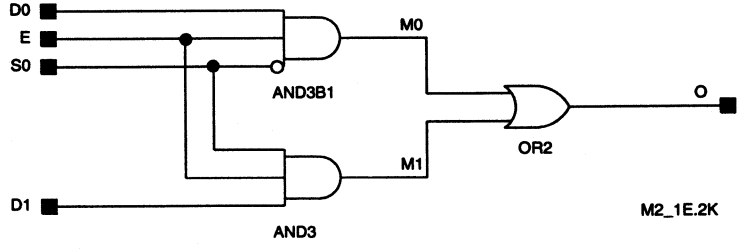
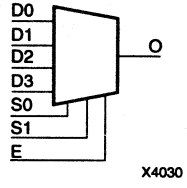


Figure 3-147 M2_1E XC2000/3000/4000 Implementation

M4_1E

4-to-1 Multiplexer with Enable



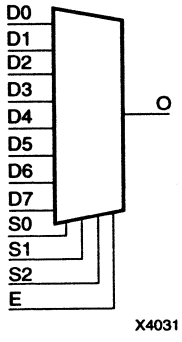
XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the enable input (E) is High, the M4_1E multiplexer chooses one data bit from four sources (D3, D2, D1, or D0) under the control of the select inputs (S1 – S0). The output (O) reflects the state of the selected input as shown in the truth table. When E is Low, the output is Low.

Inputs							Outputs
E	S1	S0	D0	D1	D2	D3	O
0	X	X	X	X	X	X	0
1	0	0	D0	X	X	X	D0
1	0	1	X	D1	X	X	D1
1	1	0	X	X	D2	X	D2
1	1	1	X	X	X	D3	D3

M8_1E

8-to-1 Multiplexer with Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the enable input (E) is High, the M8_1E multiplexer chooses one data bit from eight sources (D7 – D0) under the control of the select inputs (S2 – S0). The output (O) reflects the state of the selected input as shown in the truth table. When E is Low, the output is Low.

Inputs					Outputs
E	S2	S1	S0	D7 – D0	O
0	X	X	X	X	0
1	0	0	0	D0	D0
1	0	0	1	D1	D1
1	0	1	0	D2	D2
1	0	1	1	D3	D3
1	1	0	0	D4	D4
1	1	0	1	D5	D5
1	1	1	0	D6	D6
1	1	1	1	D7	D7

Dn represents signal on the Dn input; all other data inputs are don't-cares (X).

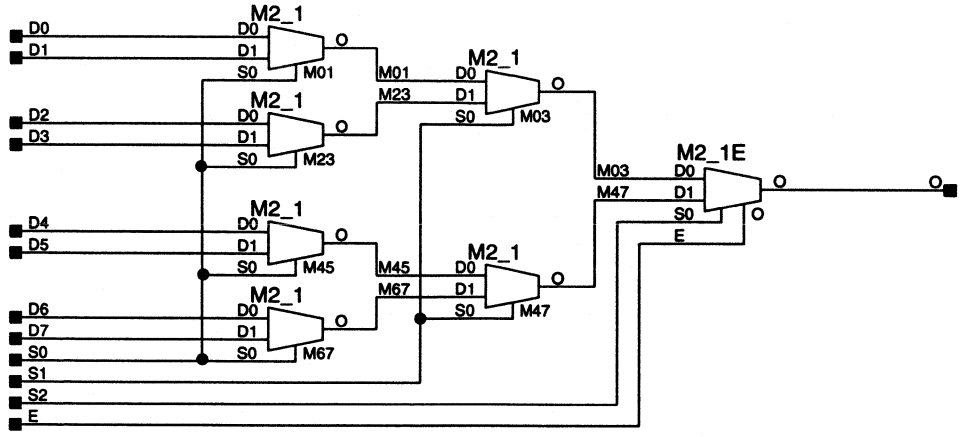
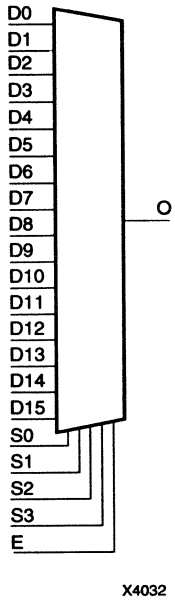


Figure 3-148 M8_1E XC2000/3000/4000 Implementation

M16_1E

16-to-1 Multiplexer with Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the enable input (E) is High, the M16_1E multiplexer chooses one data bit from 16 sources (D15 – D0) under the control of the select inputs (S3 – S0). The output (O) reflects the state of the selected input as shown in the truth table. When E is Low, the output is Low.

Inputs						Outputs
E	S3	S2	S1	S0	D15 – D0	O
0	X	X	X	X	X	0
1	0	0	0	0	D0	D0
1	0	0	0	1	D1	D1
1	0	0	1	0	D2	D2
1	0	0	1	1	D3	D3
. . .	. . .	. . .	. . .	. . .	. . .	. . .
1	1	1	0	0	D12	D12
1	1	1	0	1	D13	D13
1	1	1	1	0	D14	D14
1	1	1	1	1	D15	D15

Dn represents signal on the Dn input; all other data inputs are don't-cares (X).

NAND

2- to 9-Input NAND Gates with Inverted and Non-Inverted Inputs

Name	XC2000	XC3000	XC4000	XC7000
NAND2 – NAND4B4	Primitive	Primitive	Primitive	Primitive
NAND5 – NAND5B5	Macro	Primitive	Primitive	Primitive
NAND6 – NAND9	Macro	Macro	Macro	Primitive

The NAND function is performed in the Configurable Logic Block (CLB) function generators for XC2000, XC3000, and XC4000. NAND functions of up to five inputs are available in any combination of inverting and non-inverting inputs. NAND functions of six to nine inputs are available with only non-inverting inputs. To invert some or all inputs, use external inverters. Since each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.

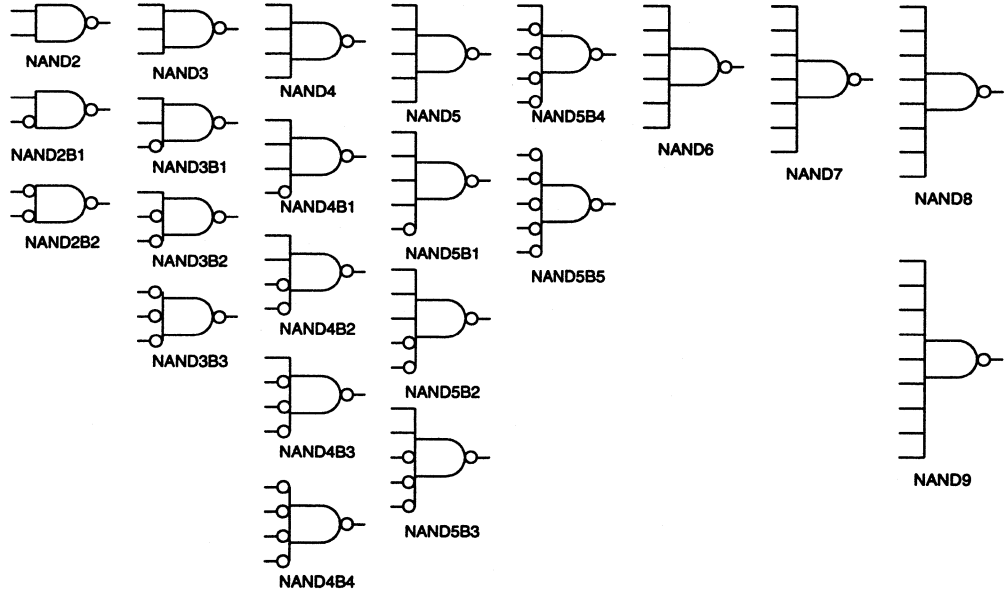


Figure 3-149 NAND Gate Representations

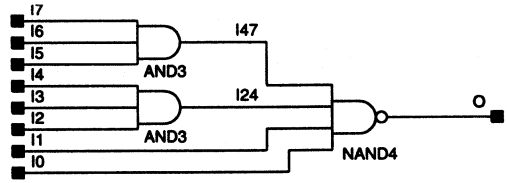


Figure 3-150 NAND8 XC2000 Implementation

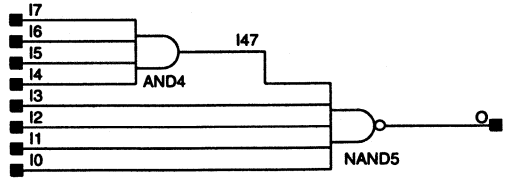


Figure 3-151 NAND8 XC3000 Implementation

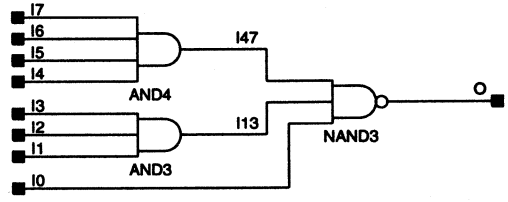


Figure 3-152 NAND8 XC4000 Implementation

NOR

2- to 9-Input NOR Gates with Inverted and Non-Inverted Inputs

Name	XC2000	XC3000	XC4000	XC7000
NOR2 – NOR4B4	Primitive	Primitive	Primitive	Primitive
NOR5 – NOR5B5	Macro	Primitive	Primitive	Primitive
NOR6 – NOR9	Macro	Macro	Macro	Primitive

The NOR function is performed in the Configurable Logic Block (CLB) function generators for XC2000, XC3000, and XC4000. NOR functions of up to five inputs are available in any combination of inverting and non-inverting inputs. NOR functions of six to nine inputs are available with only non-inverting inputs. To invert some or all inputs, use external inverters. Since each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.

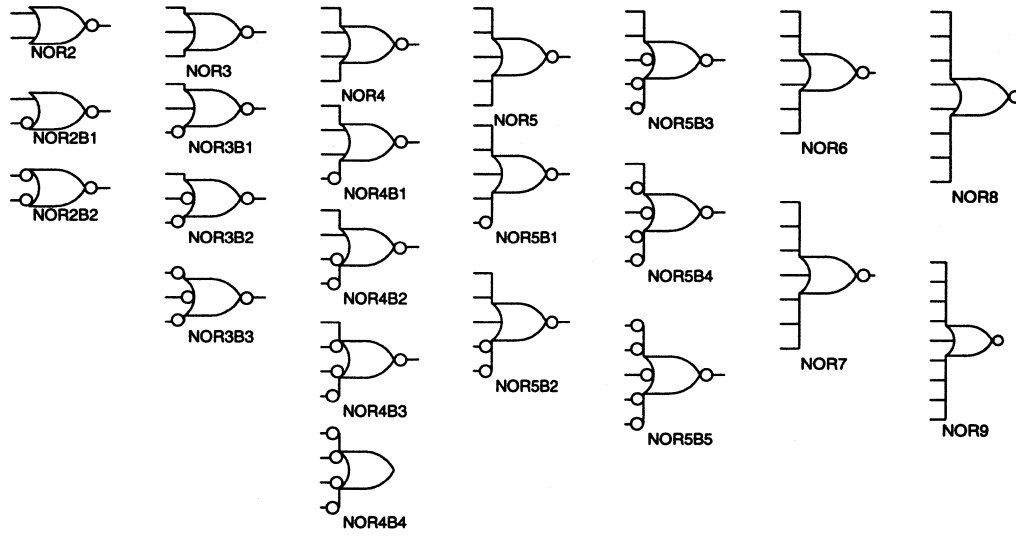


Figure 3-153 NOR Gate Representations

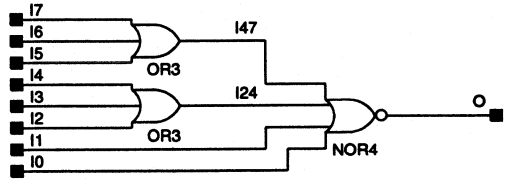


Figure 3-154 NOR8 XC2000 Implementation

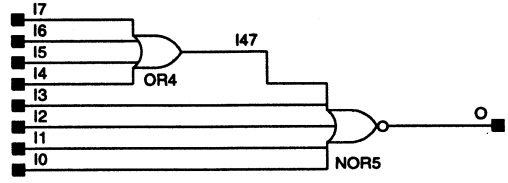


Figure 3-155 NOR8 XC3000 Implementation

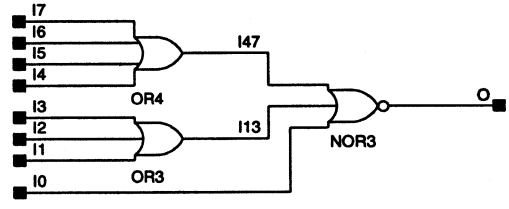
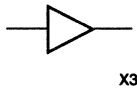


Figure 3-156 NOR8 XC4000 Implementation

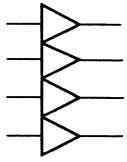
OBUF, OBUF4, OBUF8, and OBUF16

Single- and Multiple-Output Buffers



X3785

OBUF4



X3792

OBUF8

X3804

OBUF16

X3816

Name	XC2000	XC3000	XC4000	XC7000
OBUF	Primitive	Primitive	Primitive	Primitive
OBUF4, OBUF8, OBUF16	Macro	Macro	Macro	Macro

OBUF, OBUF4, OBUF8, and OBUF16 are single and multiple output buffers. An OBUF isolates the internal circuit and provides drive current for signals leaving a chip. OBUFs exist in input/output blocks (IOB). The output (O) of an OBUF is connected to an OPAD or an IOPAD. For XC7000, if a high impedance (Z) signal from an on-chip 3-state buffer (like BUFE) is applied to the input of an OBUF, it is propagated to the EPLD device output pin.

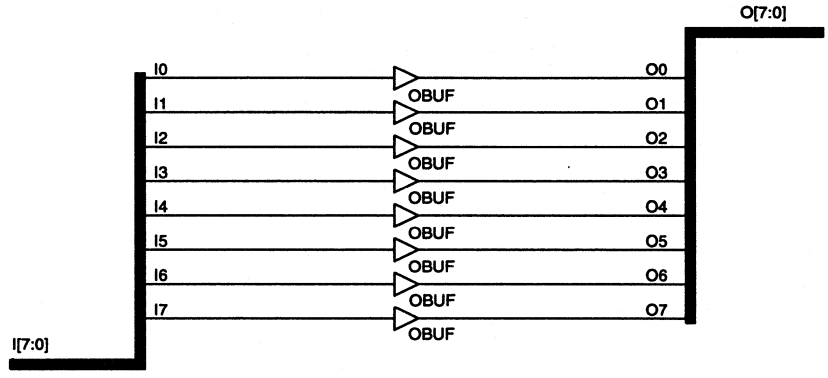
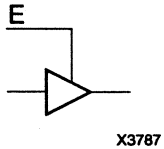


Figure 3-157 OBUF8 XC2000/3000/4000/7000 Implementation

OBUFE, OBUFE4, OBUFE8, and OBUFE16

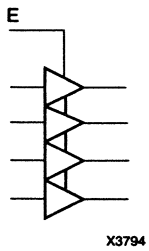
3-State Output Buffers with Active-High Output Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Macro

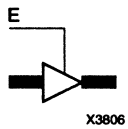
OBUFE, OBUFE4, OBUFE8, and OBUFE16 are single or multiple 3-state buffers with inputs I, I3 – I0, I7 – I0, and so forth, outputs O, O3 – O0, O7 – O0, and so forth, and active-High output enable (E). When E is High, data on the inputs of the buffers is transferred to the corresponding outputs. When E is Low, the output is High impedance (off or Z state). An OBUFE isolates the internal circuit and provides drive current for signals leaving a chip. An OBUFE output is connected to an OPAD or an IOPAD. An OBUFE input is connected to the internal circuit.

OBUFE4

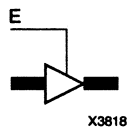


Inputs		Outputs
E	I	O
0	X	Z
1	1	1
1	0	0

OBUFE8



OBUFE16



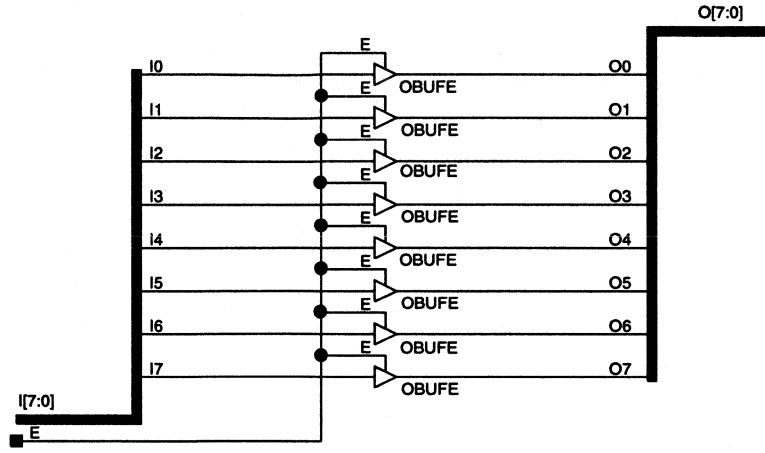


Figure 3-158 OBUFE8 XC2000/3000/4000/7000 Implementation

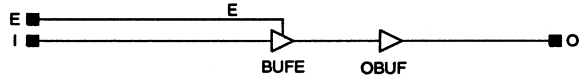
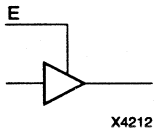


Figure 3-159 OBUFE XC7000 Implementation

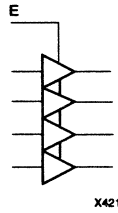
OBUFEX1, OBUFE4X1, OBUFE8X1, and OBUFEX2

EPLD 3-State Output Buffers with Active-High Output Enable

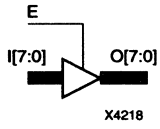
OBUFEX1



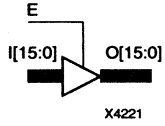
OBUFE4X1



OBUFE8X1



OBUFEX2



Name	XC2000	XC3000	XC4000	XC7000
OBUFEX1	N/A	N/A	N/A	Primitive*
OBUFE4X1, OBUFE8X1, OBUFEX2	N/A	N/A	N/A	Macro*

* not supported for XC7272 designs

OBUFEX1, OBUFE4X1, OBUFE8X1, and OBUFEX2 provide one, four, eight, and sixteen 3-state output buffers, respectively, with active-High output enable (E) and EPLD-style 3-state properties. OBUFEX1 symbols identify signals that are driven onto EPLD device pins. When E is High, data on the inputs of the buffers is transferred to the corresponding device outputs. When E is Low, the output is high impedance (off or Z state). The E input can only be driven by an EPLD global Fast Output Enable (FOE) net represented by the BUFFOE symbol.

If the input (I) of an OBUFEX1 is driven by an on-chip 3-state buffer (such as BUFT), the 3-state signal is propagated through the OBUFEX1 onto the EPLD device pin, which means the output driver of the device pin is controlled by both the on-chip 3-state buffer and the E input of the OBUFEX1. An output of an OBUFEX1 is connected to an OPAD or an IOPAD.

Inputs		Outputs
E	I	O
0	X	Z
1	1	1
1	0	0
1	Z	Z

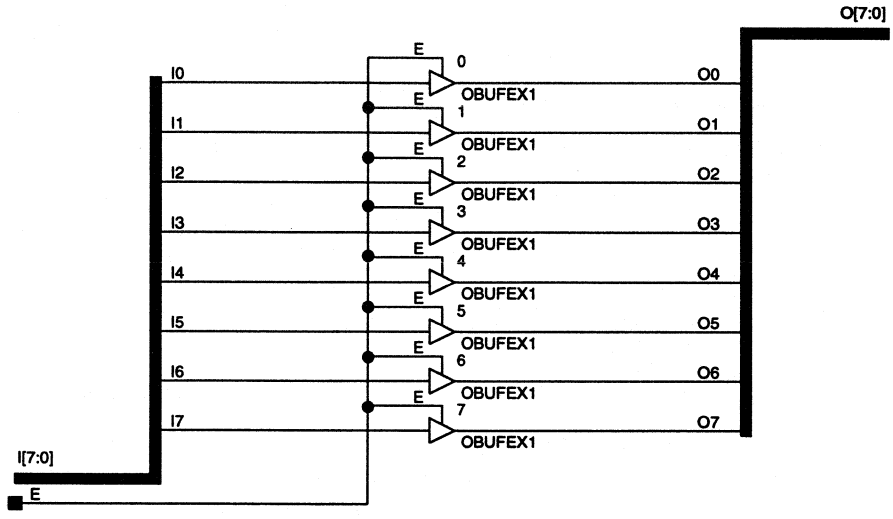
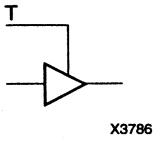


Figure 3-160 OBUFE8X1 XC7000 Implementation

OBUFT, OBUFT4, OBUFT8, and OBUFT16

Single and Multiple 3-State Output Buffers with Active-Low Output Enable

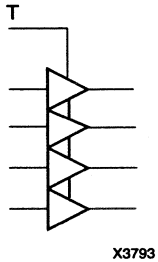
OBUFT



Name	XC2000	XC3000	XC4000	XC7000
OBUFT	Primitive	Primitive	Primitive	Macro*
OBUFT4, OBUFT8, OBUFT16	Macro	Macro	Macro	Macro*

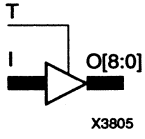
* not supported for XC7336 designs

OBUFT4



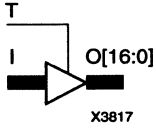
OBUFT, OBUFT4, OBUFT8, and OBUFT16 are single and multiple 3-state output buffers with inputs I, I3 – I0, I7 – I0, I15 – I0, outputs O, O3 – O0, O7 – O0, O15 – O0, and active-Low output enables (T). When T is Low, data on the inputs of the buffers is transferred to the corresponding outputs. When T is High, the output is high impedance (off or Z state). OBUFTs isolate the internal circuit and provide extra drive current for signals leaving a chip. An OBUFT output is connected to an OPAD or an IOPAD.

OBUFT8



Inputs		Outputs
T	I	O
1	X	Z
0	1	1
0	0	0

OBUFT16



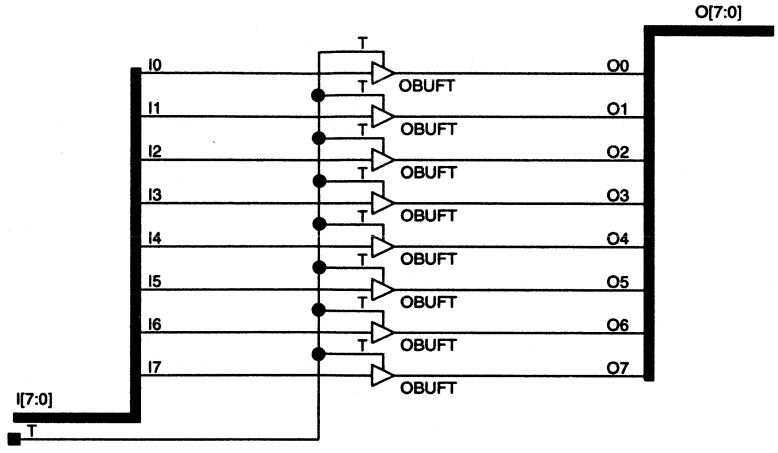


Figure 3-161 OBUFT8 XC2000/3000/4000/7000 Implementation

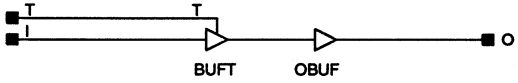
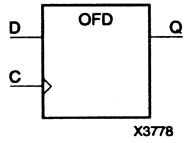
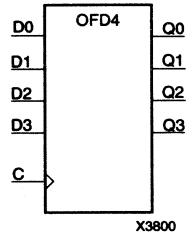


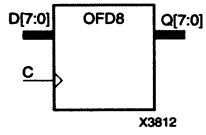
Figure 3-162 OBUFT XC7000 Implementation

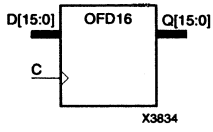
OFD, OFD4, OFD8, and OFD16

Single- and Multiple-Output D Flip-Flops









Name	XC2000	XC3000	XC4000	XC7000
OFD	N/A	Primitive	Primitive	Macro
OFD4, OFD8, OFD16	N/A	Macro	Macro	Macro

OFD, OFD4, OFD8, and OFD16 are single and multiple output D flip-flops. The flip-flops exist in an input/output block (IOB) for XC3000 and XC4000. The outputs (for example, Q3 – Q0) are connected to OPADs or IOPADs. The data on the D inputs is loaded into the flip-flops during the Low-to-High clock (C) transition and appears on the Q outputs.

The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs		Outputs
D	C	Q
D	↑	dn

dn = state of referenced input one set-up time prior to active clock transition

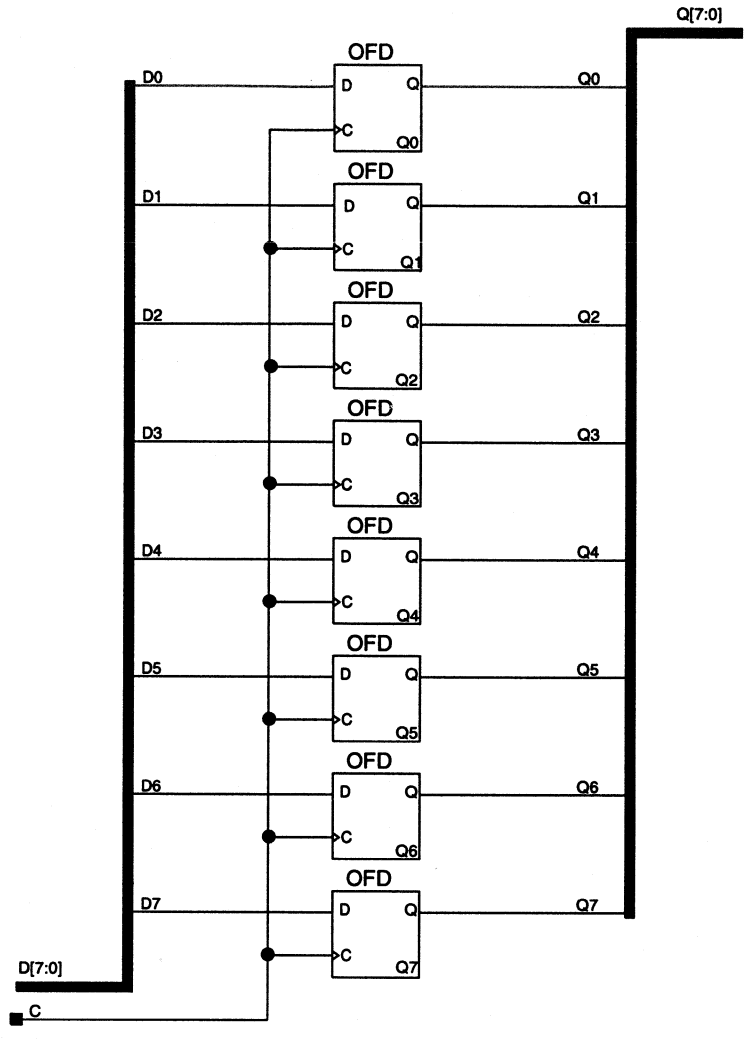


Figure 3-163 OFD8 XC3000/4000 Implementation

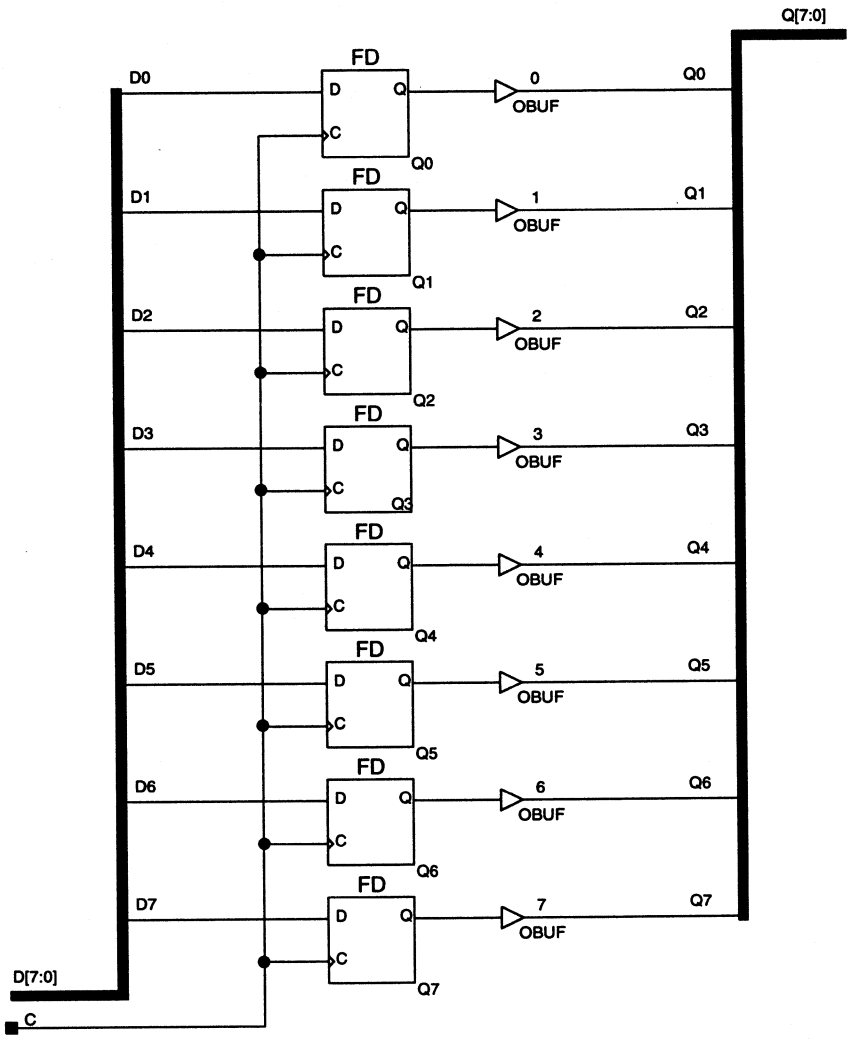


Figure 3-164 OFD8 XC7000 Implementation

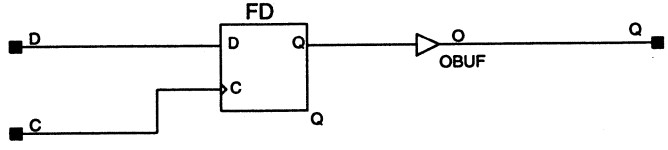
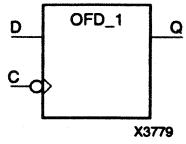


Figure 3-165 OFD XC7000 Implementation

OFD_1

Output D Flip-Flop with Inverted Clock



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	N/A

OFD_1 exists in an input/output block (IOB). The output (Q) of the D flip-flop is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition and appears on the Q output.

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC2000, XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs		Outputs
D	C	Q
D	↓	d

d = state of referenced input one set-up time prior to active clock transition

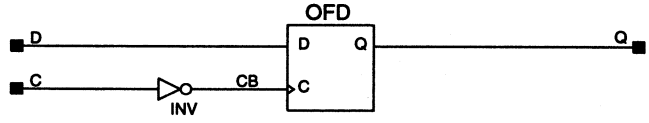
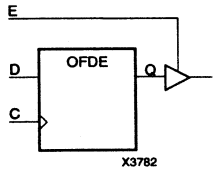


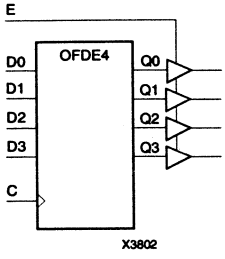
Figure 3-166 OFD_1 XC3000/4000 Implementation

OFDE, OFDE4, OFDE8, and OFDE16

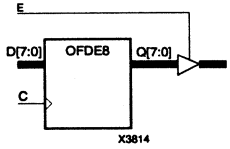
D Flip-Flops with Active-High Enable Output Buffers



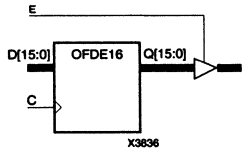
Name	XC2000	XC3000	XC4000	XC7000
OFDE	N/A	Macro	Macro	Macro
OFDE4, OFDE8, OFDE16	N/A	Macro	Macro	Macro



OFDE, OFDE4, OFDE8, and OFDE16 are single or multiple D flip-flops whose outputs are enabled by 3-state buffers. The flip-flop data outputs (Q) are connected to the inputs of output buffers (OBUFE). The OBUFE outputs (O) are connected to OPADs or IOPADs. These flip-flops and buffers are contained in input/output blocks (IOB) for XC3000 and XC4000. The data on the data inputs (D) is loaded into the flip-flops during the Low-to-High clock (C) transition. When the active-High enable inputs (E) are High, the data on the flip-flop outputs (Q) appears on the O outputs. When E is Low, outputs are high impedance (Z state or off).



The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.



Inputs			Outputs
E	D	C	O
0	X	X	Z, not off
1	1	↑	1
1	0	↑	0

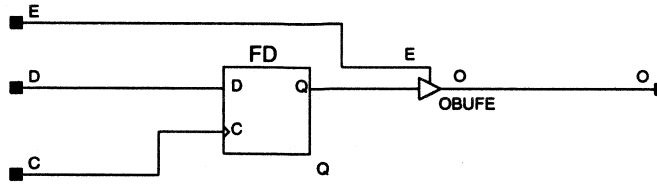


Figure 3-167 OFDE XC7000 Implementation

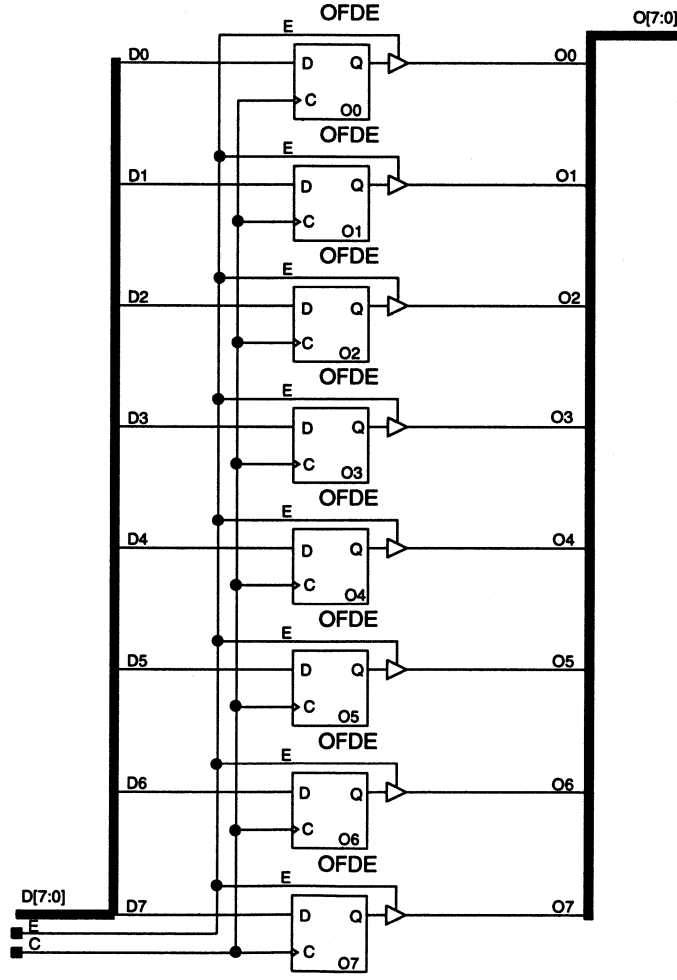


Figure 3-168 OFDE8 XC3000/4000 Implementation

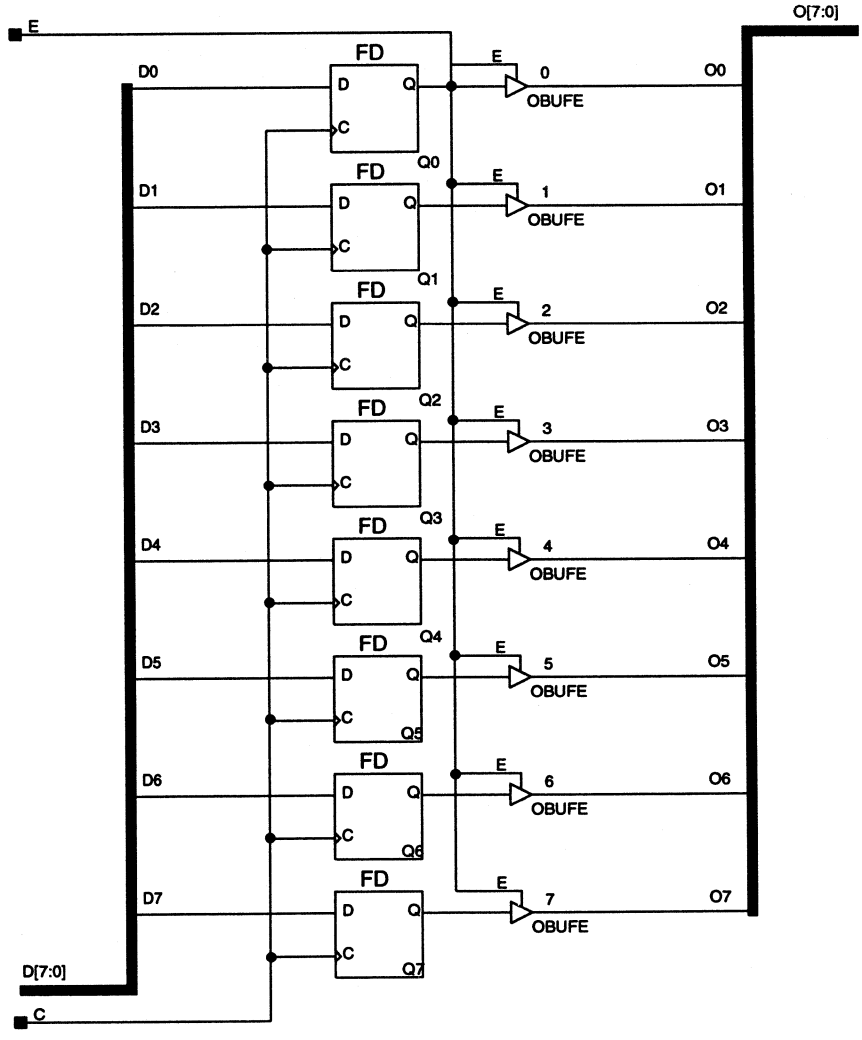
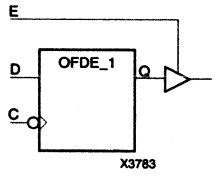


Figure 3-169 OFDE8 XC7000 Implementation

OFDE_1

D Flip-Flop with Active-High Enable Output Buffer and Inverted Clock



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	N/A

OFDE_1 and its output buffer exist in an input/output block (IOB). The data output of the flip-flop (Q) is connected to the input of an output buffer or OBUF. The output of the OBUF is connected to an OPAD or an IOPAD. The data on the data input (D) is loaded into the flip-flop on the High-to-Low clock (C) transition. When the active-High enable input (E) is High, the data on the flip-flop output (Q) appears on the O output. When E is Low, the output is high impedance (Z state or off).

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
E	D	C	O
0	X	X	Z
1	1	↓	1
1	0	↓	0

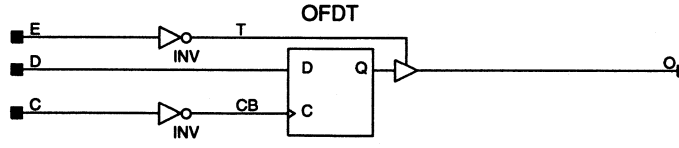
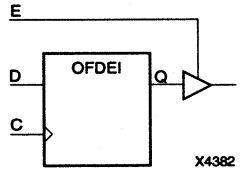


Figure 3-170 OFDE_1 XC3000/4000 Implementation

OFDEI

D Flip-Flop with Active-High Enable Output Buffer (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

OFDEI is a D flip-flop whose output is enabled by a 3-state buffer. The data output (Q) of the flip-flop is connected to the input of an output buffer or OBUF. The output of the OBUF (O) is connected to an OPAD or an IOPAD. These flip-flops and buffers are contained in input/output blocks (IOB). The data on the data input (D) is loaded into the flip-flop during the Low-to-High clock (C) transition. When the active-High enable input (E) is High, the data on the flip-flop output (Q) appears on the O output. When E is Low, the output is high impedance (Z state or off). The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs			Outputs
E	D	C	O
0	X	X	Z
1	1	↑	1
1	0	↑	0

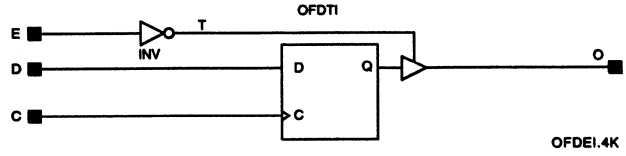
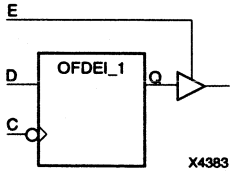


Figure 3-171 OFDEI XC4000 Implementation

OFDEI_1

D Flip-Flop with Active-High Enable Output Buffer and Inverted Clock (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

OFDEI_1 and its output buffer exist in an input/output block (IOB). The data output of the flip-flop (Q) is connected to the input of an output buffer or OBUF. The output of the OBUF is connected to an OPAD or an IOPAD. The data on the data input (D) is loaded into the flip-flop on the High-to-Low clock (C) transition. When the active-High enable input (E) is High, the data on the flip-flop output (Q) appears on the O output. When E is Low, the output is high impedance (Z state or off). The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs			Outputs
E	**D**	**C**	**O**
0	X	X	Z
1	1	↓	1
1	0	↓	0

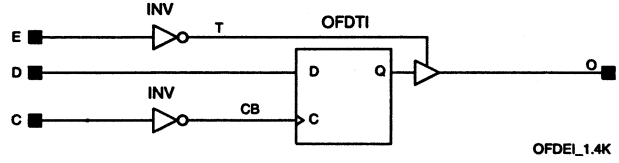
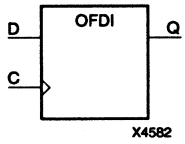


Figure 3-172 OFDEI_1 XC4000 Implementation

OFDI

Output D Flip-Flop (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

OFDI is contained in an input/output block (IOB). The output (Q) of the D flip-flop is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q).

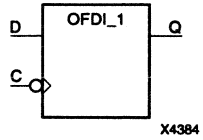
The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs		Outputs
D	C	Q
D	↑	d

d = state of referenced input one set-up time prior to active clock transition

OFDI_1

Output D Flip-Flop with Inverted Clock (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

OFDI_1 exists in an input/output block (IOB). The D flip-flop output (Q) is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition and appears on the Q output. The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs		Outputs
D	C	Q
D	↓	d

d = state of referenced input one set-up time prior to the active clock transition

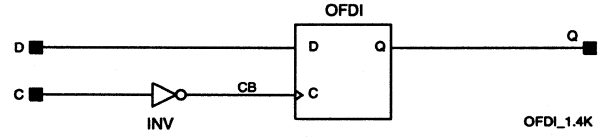
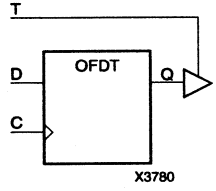


Figure 3-173 OFDI_1 XC4000 Implementation

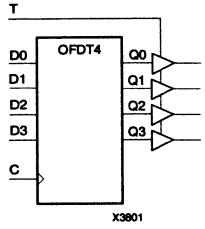
OFDT, OFDT4, OFDT8, and OFDT16

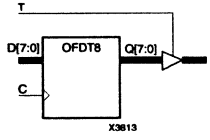
Single and Multiple D Flip-Flops with Active-High 3-State Active-Low Output Enable Buffers

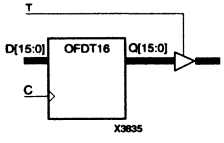


Name	XC2000	XC3000	XC4000	XC7000
OFDT	N/A	Primitive	Primitive	Macro*
OFDT4, OFDT8, OFDT16	N/A	Macro	Macro	Macro*

* not supported for XC7336 designs







OFDT, OFDT4, OFDT8, and OFDT16 are single or multiple D flip-flops whose outputs are enabled by a 3-state buffers. The data outputs (Q) of the flip-flops are connected to the inputs of output buffers (OBUFT). The outputs of the OBUFTs (O) are connected to OPADs or IOPADs. These flip-flops and buffers exist in input/output blocks (IOB) for XC3000 and XC4000. The data on the data inputs (D) is loaded into the flip-flops during the Low-to-High clock (C) transition. When the active-Low enable inputs (T) are Low, the data on the flip-flop outputs (Q) appears on the O outputs. When T is High, outputs are high impedance (off).

The flip-flops are asynchronously reset, outputs Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
T	D	C	O
1	X	X	Z
0	D	↑	d

d = state of referenced input one set-up time prior to active clock transition

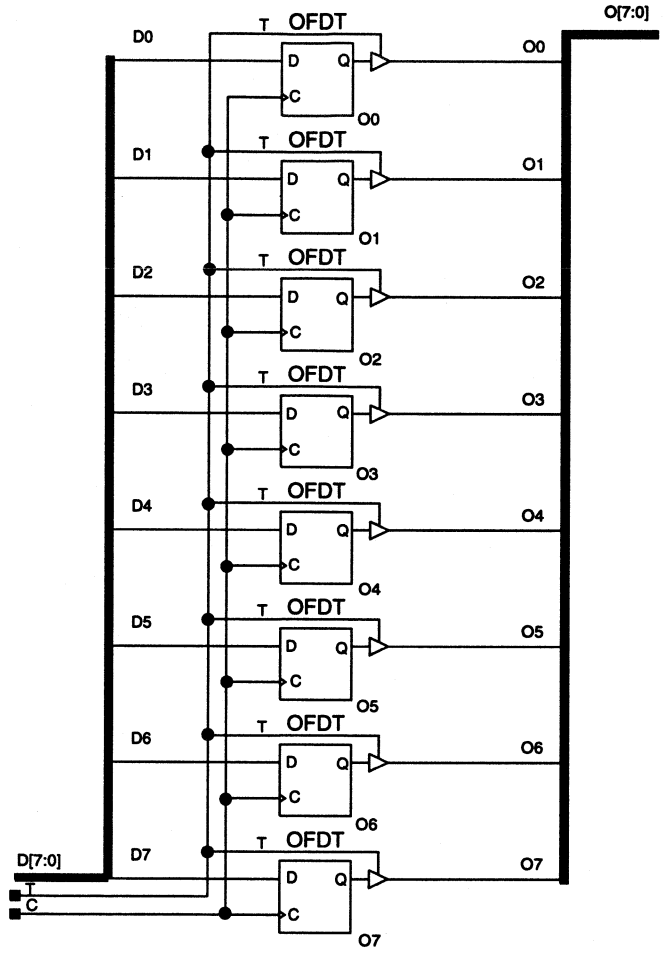


Figure 3-174 OFDT8 XC3000/4000 Implementation

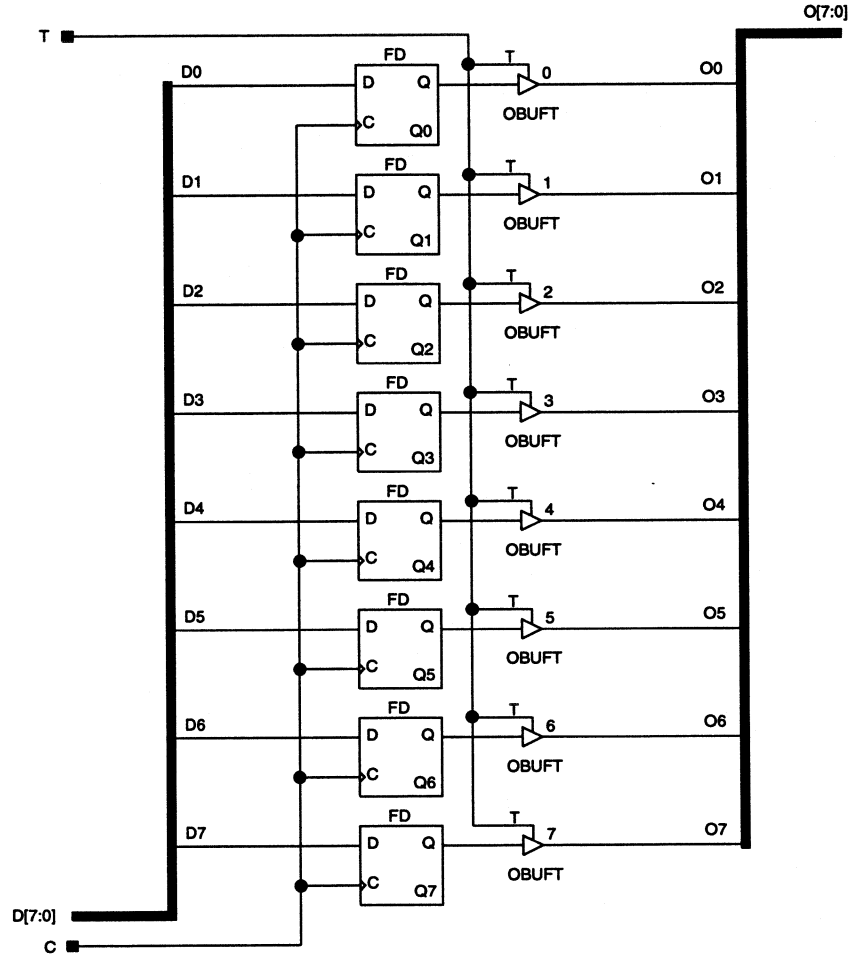


Figure 3-175 OFDT8 XC7000 Implementation

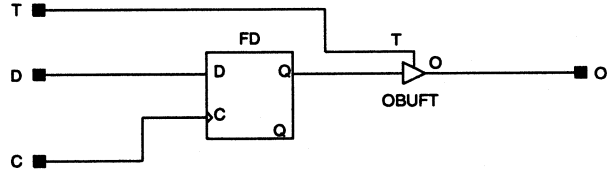
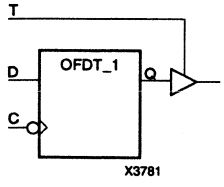


Figure 3-176 OFDT XC7000 Implementation

OFDT_1

D Flip-Flop with Active-High 3-State and Active-Low Output Buffer and Inverted Clock



XC2000	XC3000	XC4000	XC7000
N/A	Macro	Macro	N/A

OFDT_1 and its output buffer exist in an input/output block (IOB). The flip-flop data output (Q) is connected to the input of an output buffer (OBUF). The OBUF output is connected to an OPAD or an IOPAD. The data on the data input (D) is loaded into the flip-flop on the High-to-Low clock (C) transition. When the active-Low enable input (T) is Low, the data on the flip-flop output (Q) appears on the O output. When T is High, the output is high impedance (off).

The flip-flop is asynchronously reset, output Low, when power is applied or when global reset (GR for XC3000) or global set/reset (GSR for XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs			Outputs
T	**D**	**C**	**O**
1	X	X	Z
0	1	↓	1
0	0	↓	0

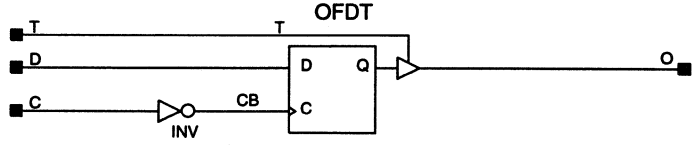
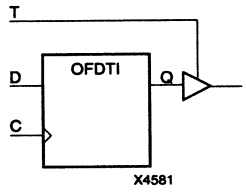


Figure 3-177 OFDT_1 XC3000/4000 Implementation

OFDTI

D Flip-Flop with Active-High 3-State and Active-Low Output Buffer (Asynchronous Set)



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

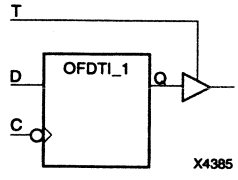
OFDTI and its output buffer are contained in an input/output block (IOB). The data output of the flip-flop (Q) is connected to the input of an output buffer (OBUF). The output of the OBUF is connected to an OPAD or an IOPAD. The data on the data input (D) is loaded into the flip-flop on the Low-to-High clock (C) transition. When the active-Low enable input (T) is Low, the data on the flip-flop output (Q) appears on the output (O). When T is High, the output is high impedance (off).

The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs			Outputs
T	D	C	O
1	X	X	Z
0	1	↑	1
0	0	↑	0

OFDTI_1

D Flip-Flop with Active-High 3-State, Active-Low Output Buffer and Inverted Clock



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

OFDTI_1 and its output buffer are contained in an input/output block (IOB). The data output of the flip-flop (Q) is connected to the input of an output buffer (OBUF). The OBUF output is connected to an OPAD or an IOPAD. The data on the data input (D) is loaded into the flip-flop on the High-to-Low clock (C) transition. When the active-Low enable input (T) is Low, the data on the flip-flop output (Q) appears on the O output. When T is High, the output is high impedance (off).

The flip-flop is asynchronously set, output High, when power is applied or when global set/reset (GSR) is active. The GSR active level is programmable.

Inputs			Outputs
T	**D**	**C**	**O**
1	X	X	Z
0	1	↓	1
0	0	↓	0

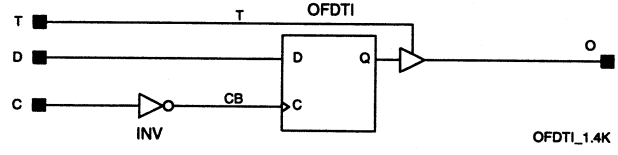
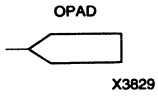


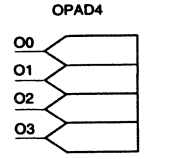
Figure 3-178 OFDTI_1 XC4000 Implementation

OPAD, OPAD4, OPAD8, and OPAD16

Single- and Multiple-Output Pads

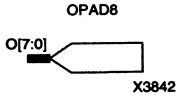


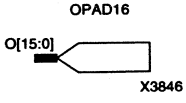
Name	XC2000	XC3000	XC4000	XC7000
OPAD	Primitive	Primitive	Primitive	Primitive
OPAD4, OPAD8, OPAD16	Macro	Macro	Macro	Macro



OPAD, OPAD4, OPAD8, and OPAD16 are single and multiple output pads. An OPAD connects a device pin to an output signal of a PLD. It is internally connected to an input/output block (IOB), which is configured by the XACT software as an OBUF, an OBUFT, an OBUFE, an OFD, or an OFDT.

Refer to the appropriate CAE tool interface user guide for details on assigning pin location and identification.





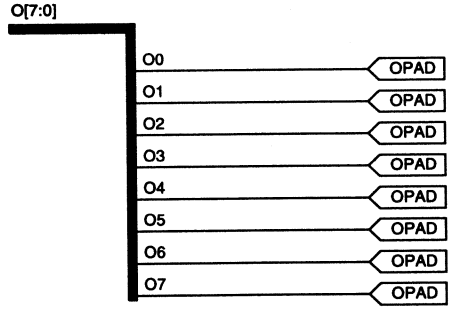


Figure 3-179 OPAD8 XC2000/3000/4000 Implementation

OR

2- to 9-Input OR Gates with Inverted and Non-Inverted Inputs

Name	XC2000	XC3000	XC4000	XC7000
OR2 – OR4B4	Primitive	Primitive	Primitive	Primitive
OR5 – OR5B5	Macro	Primitive	Primitive	Primitive
OR6 – OR9	Macro	Macro	Macro	Primitive

The OR function is performed in the Configurable Logic Block (CLB) function generators for XC2000, XC3000, and XC4000. OR functions of up to five inputs are available in any combination of inverting and non-inverting inputs. OR functions of six to nine inputs are available with only non-inverting inputs. To invert some or all inputs, use external inverters. Since each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.

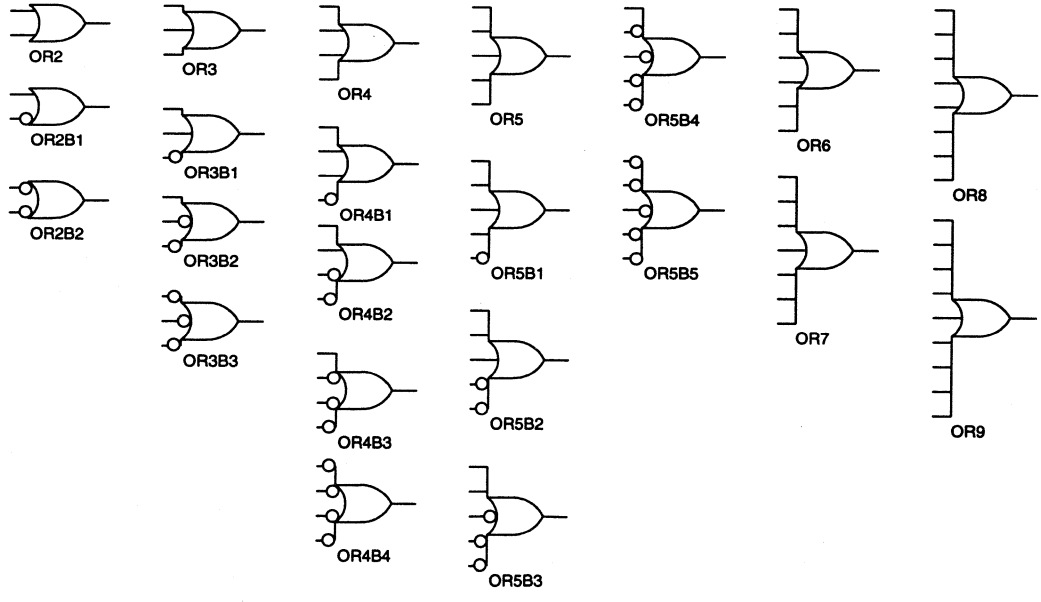


Figure 3-180 OR Gate Representations

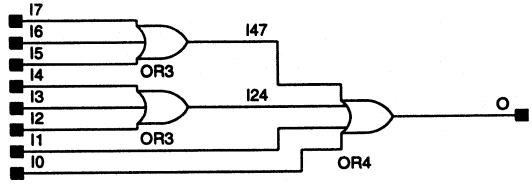


Figure 3-181 OR8 XC2000 Implementation

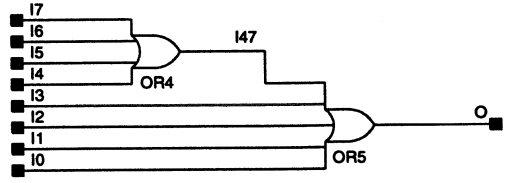


Figure 3-182 OR8 XC3000 Implementation

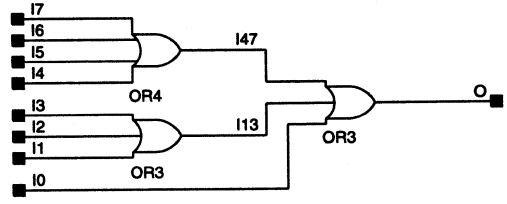
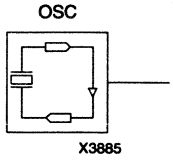


Figure 3-183 OR8 XC4000 Implementation

OSC

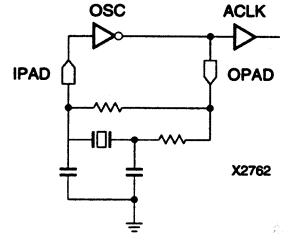
Crystal Oscillator Amplifier



XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	N/A	N/A

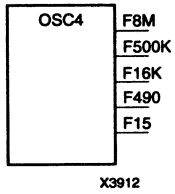
The OSC element's clock signal frequency is derived from an external crystal-controlled oscillator. The OSC output can be connected to an ACLK buffer, which is connected to an internal clock net.

Two dedicated input pins (XTAL 1 and XTAL 2) on each FPGA device are internally connected to pads and input/output blocks that are connected to the OSC amplifier. The external components are connected as shown in the following example. Refer to *The Programmable Gate Array Data Book* for details on component selection and tolerances.



OSC4

Internal 5-Frequency Clock-Signal Generator

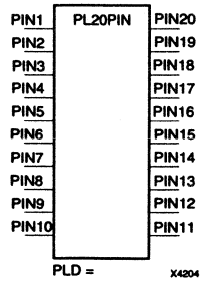


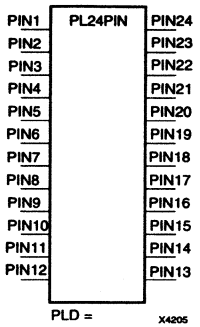
XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

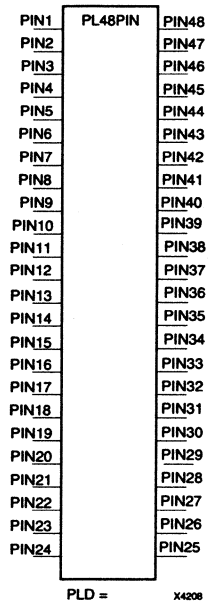
OSC4 provides internal clock signals in applications where timing is not critical. The available frequencies are determined by FPGA device components, which are process dependent. Therefore, the available frequencies vary from device to device. Nominal frequencies are 8 MHz, 500 KHz, 16 KHz, 490 Hz, and 15 Hz. Although there are five outputs, only three can be used at a time, with 8 MHz on one output and one frequency each on any two of the remaining four outputs. An error occurs if more than three outputs are used simultaneously. The internal circuit must be connected through buffers to OSC4 outputs.

PL20PIN, PL24PIN, and PL48PIN

Generic PLD Symbols for EPLD







XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

PL20PIN, PL24PIN, and PL48PIN symbols represent various discrete PLD devices that can be integrated into an EPLD design. Functionality is defined using PALASM-compatible Boolean equation files, similar to files used to pattern actual PLD devices. Pins on the generic PLD symbol correspond to the ordered list of signal names appearing in the equation file. The equation file defines the functionality; it is specified by applying the attribute PLD=*filename*. EPLD implementation software reads the specified equation file when it encounters the generic PLD symbol.

By default, the PIN1 input of the generic PLD symbol is the clock for all registered outputs, unless otherwise specified by CLKF equations.

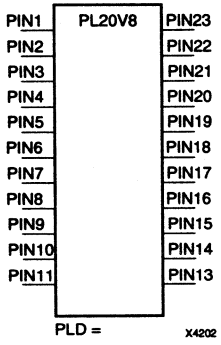
If a PLD symbol output is connected to an output buffer (OBUF or OBUFEX1), any 3-state (TRST) control function specified for the output controls the corresponding I/O pin of the chip. By default, 3-state control equations only control the 3-state drivers of connected EPLD device pins; the signals received by any other on-chip logic and feedback always remain enabled, unless you specify NODETRST or you use an XC7272 device.

EPLD Device Limitations

In XC7272 designs, both SETF and TRST equations cannot be used for the same output. Also, you can only specify one input variable (true or complement) as a 3-state function; AND functions are not supported in TRST equations. For XC7272, the only behavior available is NODETRST. Therefore, any 3-state output signals are completely disabled by the 3-state (TRST) control equation, including feedback within the PLD equation file.

PL20V8

20V8-Compatible PLD Symbol for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

The PL20V8 symbol represents a GAL20V8 PLD covering all 24-pin medium PAL devices. Functionality is defined using the same PALASM-compatible Boolean equation file or GAL20V8 JEDEC programming file used to pattern an actual GAL or PAL device. Pins on the PL20V8 symbol correspond to the ordered list of signal names appearing in the equation file (or JEDEC file). You can specify the equation file used to define the functionality by applying the attribute PLD=*filename*. EPLD implementation software reads the specified equation file when it encounters the PL20V8 symbol.

GAL20V8-Compatible Functionality

Generally, the PL20V8 produces up to eight output functions. You can synchronize each output with a D-type flip-flop. Flip-flops are triggered on the rising edge of the clock signal. By default, the PIN1 input is the clock for all registers. You can invert each output.

Each output has 3-state drive capability. If the output is connected to an output buffer (OBUF or OBUFEX1), the corresponding 3-state I/O pin is controlled by the PL20V8 output-enable control input, PIN 13, or a 3-state (TRST) control equation. By default, each output specified by a non-registered equation is always enabled, and each registered output is enabled when the PIN13 input is Low, which makes it compatible with 24-pin medium PAL devices. Unused output pins are always disabled to allow these pins to be reused as inputs. A single product term can be used to control the output enable of each output (pins 15 – 22) through the 3-state (TRST) control equation.

The defaults mentioned in the previous paragraphs override the normal defaults for the Xilinx PLUSASM equation language when PL20V8 is the specified PLD component type.

GAL20V8 Exceptions

By default, the output enable controls of a PL20V8 control only the 3-state drivers of the EPLD device pins, not the signals received by any other on-chip logic. For example, if a 3-state control equation is specified and the corresponding output only connects to other on-chip logic symbols, the 3-state equation has no effect. You can override this default by using the PLUSASM NODETRST declaration in your equation file. When NODETRST is specified, the output signal is disabled by the 3-state control equation everywhere it occurs in the design, including feedback within the PL20V8 equation file.

Extended Functionality

As an alternative to using the default clock, PIN1, you can take a single product term from each registered output function and use it as the clock source (logic-controlled clocking) by specifying a CLKF control equation in your equation file.

You can synchronously set and/or reset each output register using single product-term functions. You can take a single product term from each output to control the register set or reset by specifying the SETF or the RSTF control equation. The register is forced High or Low while the set or reset product term is satisfied.

When the EPLD device is powered up or its Master Reset pin is activated, all registers in the device are initialized. You can select the initial state of each PL20V8 output flip-flop using the PLUSASM PRLD control equation. Polarity inversion is performed before the D-input to the flip-flop, making the results of reset, set, and preloading (power on or Master Reset) on the outputs independent of selected polarity.

You can override the default 3-state control of registered outputs through PIN 13 by specifying the control equation *output*•TRST=VCC in the equation file for each output. Overriding the default forces the output to always be enabled, allows PIN13 to be used as an ordinary logic input, and allows the associated product term to be used as part of the output logic-defining function.

Pin Description

PIN1 is the default clock input. It triggers all registers on the Low-to-High transition and can be driven by a FastCLK global net (BUFG symbol). Only the PIN1 input can be driven by a FastCLK global net. If it is driven by a FastCLK net, the signal cannot be used in any equations in the PLD. PIN1 can be used as a general-purpose logic input if it is not driven by a FastCLK.

PIN2 – PIN11, PIN14, and PIN23 are general-purpose logic inputs.

PIN13 is the default active-Low output-enable control input for registered outputs. Outputs are enabled while pin 13 is Low. Pin 13 can also be used as an ordinary logic input.

PIN15 – PIN22 are logic function outputs with optional 3-state control. These pins can be used as ordinary logic inputs when no corresponding output function is specified.

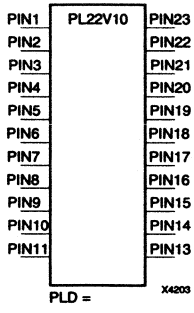
PIN12 and PIN24 are not shown on the PL20V8 symbol, because they are the ground and VCC pins.

EPLD Device Limitations

In XC7272 designs, both SETF and TRST equations cannot be used for the same output. Also, only one input variable (true or complement) can be specified as a 3-state function; AND functions are not supported in TRST equations. For XC7272, only the NODETRST behavior is available. Therefore, any 3-state output signals are completely disabled by the 3-state control equation (or pin 13 for registered outputs), including feedback within the PL20V8 equation file.

PL22V10

22V10-Compatible PLD Symbol for EPLD



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive

The PL22V10 symbol represents a PAL22V10 PLD. Functionality is defined using the same PALASM-compatible Boolean equation file or PAL22V10 JEDEC programming file used to pattern an actual PAL22V10 device. The pins on the PL22V10 symbol correspond to the ordered list of signal names appearing in the equation file (or JEDEC file). Specify the equation file by applying the attribute PLD=*filename*. EPLD implementation software reads the specified equation file when it encounters the PL22V10 symbol.

PAL22V10-Compatible Functionality

Generally, the PL22V10 produces up to 10 output functions. You can synchronize each output with a D-type flip-flop. Flip-flops are triggered on the rising edge of the clock signal. By default, the PIN1 input is the clock for all registers.

Each output has 3-state drive capability. If the output is connected to an output buffer (OBUF or OBUFEX1), the corresponding 3-state I/O pin is controlled by the PL22V10 output-enable (TRST) control equation. By default, each output equation is always enabled. Unused PL22V10 output pins are always disabled, allowing the symbol pin to be re-used as an input. You can use a single product term to control the output enable of each output through the 3-state control equation.

You can implement global asynchronous reset by specifying a signal name in the 25th pin position in the equation file and by applying a RSTF control equation that is compatible with PALASM syntax for the PAL22V10 architecture. Implementing global asynchronous reset causes the EPLD implementation software to automatically replicate the reset function on each of the outputs' reset product terms. All registers are forced Low while the global asynchronous reset function is satisfied.

Similarly, you can implement global synchronous preset by specifying a signal name in the 25th pin position in the equation file and by applying a SETF control equation. Implementing global synchro-

nous preset causes the EPLD implementation software to automatically replicate it into each of the registered output logic functions. All registers are forced High synchronously with the clock, if the global synchronous preset function is satisfied.

Individual output SETF and RSTF control equations are not supported for PL22V10 equation files.

Each output can be inverted. EPLD software automatically emulates PAL22V10 architecture by applying logic inversions to the Q-output of the flip-flop. The results of asynchronous reset, synchronous preset, and pre-loading (power-on) at the outputs are reversed as a result of selecting active-Low output polarity.

The defaults mentioned in the previous paragraphs override the normal defaults for the Xilinx PLUSASM equation language when PL22V10 is the specified PLD component type.

PAL22V10 Exceptions

Output-enable controls, by default, only control EPLD device pin 3-state drivers, not signals received by any other on-chip logic. For example, if you specify a 3-state control equation and the corresponding output only connects to other on-chip logic symbols, the 3-state equation has no effect. You can override this default using the PLUSASM NODETRST declaration in your equation file. When NODETRST is specified, the output signal is disabled by the 3-state control equation everywhere it occurs, including feedback within the PL22V10 equation file.

Extended Functionality

As an alternative to using the default clock, PIN1, you can take a single product term from each registered output function and use it as the clock source (logic-controlled clocking) by specifying a CLKF control equation in your equation file.

When the EPLD device is powered up or its Master Reset pin is activated, all registers in the device are initialized. You can select the initial state of each PL22V10 output flip-flop using the PLUSASM PRLD control equation.

Pin Description

PIN1, the default clock input, triggers all registers on the Low-to-High transition. It can be driven by a FastCLK global net (BUFG symbol). Only the PIN1 input can be driven by a FastCLK global net. If it is driven by a FastCLK net, the signal on PIN1 cannot be used in any equations in the PLD. PIN1 can be used as a general-purpose logic input if it is not driven by a FastCLK.

PIN2 – PIN11 and PIN13 are general-purpose logic inputs.

PIN14 – PIN23 are logic function outputs with optional 3-state control. These pins can be used as ordinary logic inputs when no corresponding output function is specified.

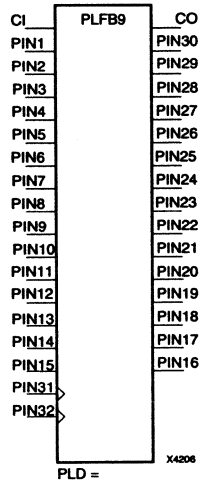
PIN12 and PIN24 are not shown on the PL22V10 symbol, because they are the ground and VCC pins.

EPLD Device Limitations

In XC7272 designs, 3-state control equations cannot be used if an asynchronous reset function is also specified. You can only specify one input variable (true or complement) as a 3-state function; AND functions are not supported in TRST equations. Only the NODETRST behavior is available for XC7272. Therefore, any 3-state output signals are completely disabled by the 3-state control equation, including feedback within the PL22V10 equation file.

PLFB9

EPLD High-Density Function Block PLD Symbol



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7336 designs

PLFB9 is a Xilinx-proprietary PLD configuration with all the logic and macrocell resources available in a Xilinx EPLD High-Density Function Block. Its features are a superset of various discrete PLD products, including medium PAL devices and registered-asynchronous devices, such as PAL20RA10.

Specify custom logic functions using the Xilinx PLUSASM Boolean equation language in a syntax upwardly compatible with the popular PALASM Boolean equation language. PLFB9 pins correspond to the ordered list of signal names appearing in the equation file. The equation file used to define the functionality is specified by applying the attribute PLD=*filename*. EPLD implementation software reads this equation file when it encounters the PLFB9 symbol in the schematic. (For equivalent logic of XC7272 macrocells, consult the device data sheet.)

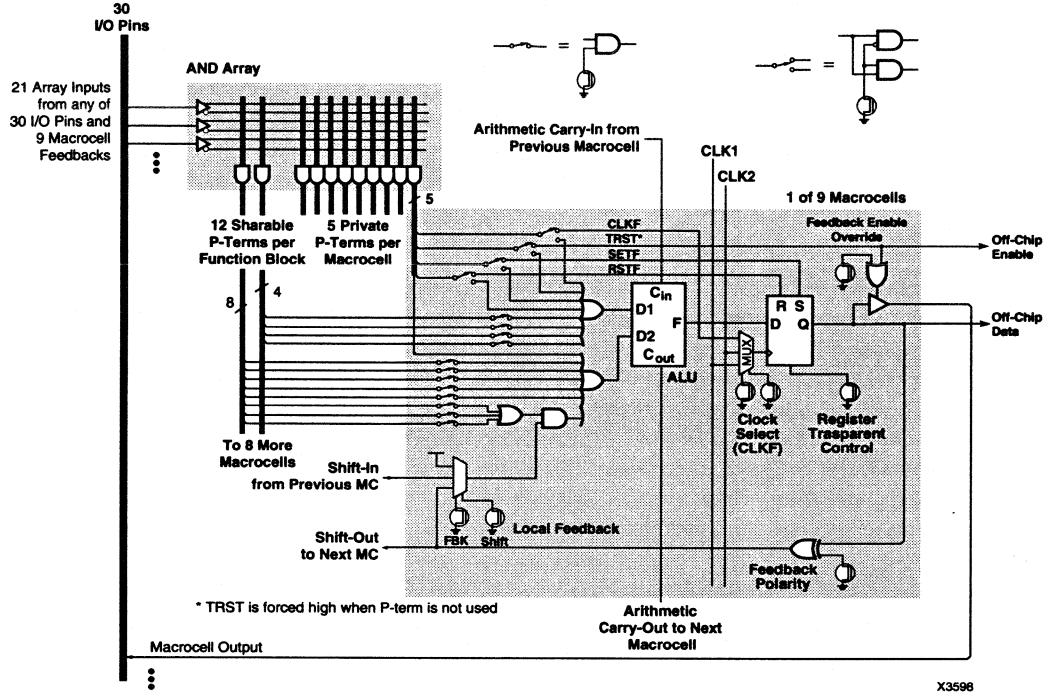


Figure 3-184 PLFB9 Macrocell Equivalent Logic

You can generate up to nine output functions. Each output can be synchronized (registered) by a D-type flip-flop. Flip-flops are triggered on the Low-to-High transition of a clock signal. The PLFB9 has two dedicated clock inputs, PIN31 and PIN32; each can only be driven by a global FastCLK signal, represented by the BUFG symbol. By default, PIN31 is the clock for all registers, unless otherwise specified by a CLKF control equation. You must specify signal names in pin positions 31 and 32 in the equation file if the corresponding clock input is used.

When the EPLD device is powered up or its Master Reset pin is activated, all registers in the device are initialized. You can select the initial state of each output flip-flop using the PLUSASM PRLD control equation.

PLFB9 Inputs and Outputs

Each PLFB9 output has an available feedback path that feeds back into the logic array. Therefore, output variables defined by PLUSASM equations can be reused as inputs to the same or other equations for state sequencing or cascaded logic. A total of 21 input channels feed into the logic array in a high-density function block. The product terms can use both true and complement of each input. These 21 input channels include any output feedback paths used by any of the equations, as well as inputs applied to PLFB9. The software automatically allocates one of the available logic array input channels whenever an output is reused by any of the equations. The total number of input variables used by all logic equations must therefore not exceed 21, even though the PLFB9 symbol provides more external connections.

PLFB9 has 30 generic I/O pins (PIN1 – PIN30) so that all logic array input channels and output functions are accessible when no output feedbacks are used. You can attach the incoming or outgoing signals to any of the symbol pins. Pin direction is determined according to usage. Unused symbol pins must be left unconnected and their corresponding positions in the equation file pin list must indicate NC (not connected).

If a PLFB9 symbol output is connected to an output buffer (OBUF or OBUFEX1), any 3-state (TRST) control function specified for the output controls the corresponding I/O pin of the chip. By default, 3-state control equations in the PLD only control 3-state drivers of connected EPLD device pins; the signals received by any other on-chip logic and feedback always remain enabled unless you specify NODETRST or you use an XC7272 device.

EPLD Device Limitations

In XC7272 designs, you cannot use both SETF and TRST equations for the same output. You can specify only one input variable (true or complement) as a 3-state function; AND functions are not supported in TRST equations. Only the NODETRST behavior is available in XC7272. Therefore, any 3-state output signals are completely disabled by the 3-state control equation, including feedback within the PLD equation file.

Arithmetic Carry

You can specify arithmetic functions across a set of adjacent outputs. The ADD (or ADDMODE) extension in PLUSASM enables the arithmetic carry path between macrocells to build efficient, high-speed, ripple-carry adders, subtracters, magnitude comparators, and accumulators. The output order is defined by the equation file pin list; output variables are listed from least- to most-significant bits.

CI and CO pins represent arithmetic carry paths into and out of the function block. CI represents the carry-in of the first macrocell. CO represents the carry-out from the ninth macrocell, and therefore is only valid if an arithmetic output function is specified in the equation file for the ninth macrocell. The CI input is from the EPLD carry chain and must only be connected to the CO output of another PLFB9 or EPLD-specific arithmetic component. The CO output passes into the EPLD carry chain and can only be connected to the CI input of another PLFB9 or EPLD-specific arithmetic component.

Pin Descriptions

PIN1 – PIN30 are generic I/O pins. Each pin can be used as one of the 21 high-density function block (HDFB) logic array inputs or nine function outputs.

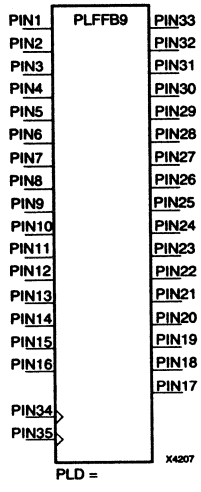
PIN31 and PIN32 are clock inputs that trigger registers on Low-to-High transitions. They must be driven by a FastCLK buffer (BUFG) and cannot be used in any logic equations.

CI is the arithmetic carry-in to the first macrocell of the function block; it can only be driven by the CO from another arithmetic component or PLFB9.

CO is the arithmetic carry-out from the ninth macrocell of the function block; it can only drive a CI input of another arithmetic component or PLFB9.

PLFFB9

EPLD Fast Function Block PLD Symbol



XC2000	XC3000	XC4000	XC7000
N/A	N/A	N/A	Primitive*

* not supported for XC7236 or XC7272 designs

PLFFB9 is a Xilinx-proprietary PLD configuration with the all logic and macrocell resources available in a Xilinx EPLD Fast Function Block. Specify custom logic functions using the Xilinx PLUSASM Boolean equation language in a syntax upwardly compatible with the PALASM Boolean equation language. Pins on a PLFFB9 symbol correspond to the ordered list of signal names appearing in the equation file. Specify the equation file used to define the functionality by applying the attribute PLD=*filename* to the PLFFB9 instance in the schematic. EPLD implementation software reads this equation file when it encounters a PLFFB9 symbol. (For extended logic capabilities of XC7336 macrocells, consult the device data sheet.)

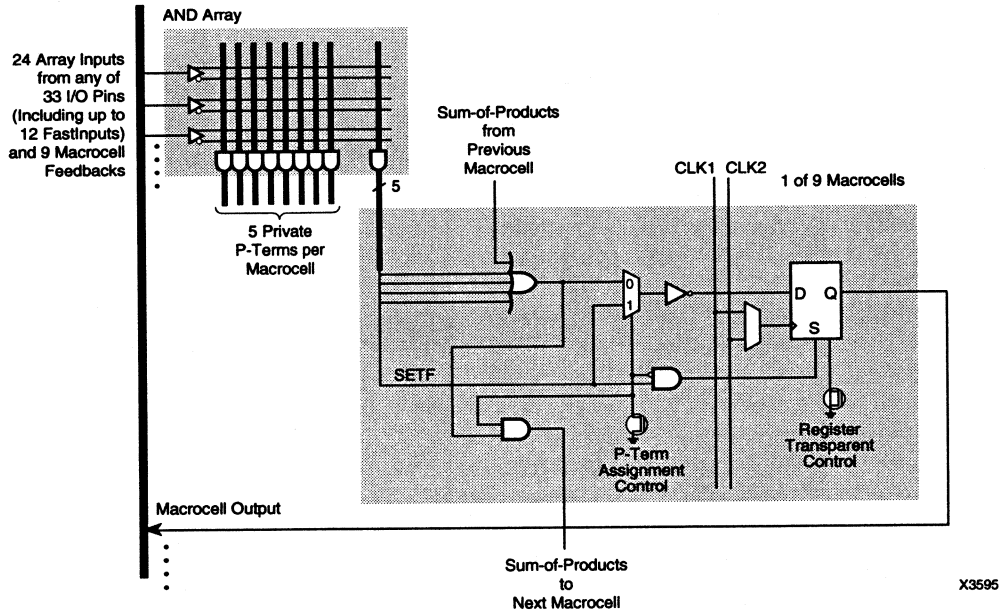


Figure 3-185 PLFFB9 Macrocell Equivalent Logic

You can generate up to nine output functions. Each output can be synchronized (registered) by a D-type flip-flop. Flip-flops are triggered on the Low-to-High transition of a clock signal. PLFFB9 has two dedicated clock inputs, PIN34 and PIN35; each can only be driven by a global FastCLK signal (represented by the BUFG symbol). By default, PIN34 is the clock for all registers, unless otherwise specified by a CLKF control equation. Signal names must be specified in pin positions 34 and 35 in the equation file if the corresponding clock input is used.

PLFFB9 Inputs and Outputs

Each output has an available feedback path that feeds back into the logic array. Output variables defined by PLUSASM equations can be reused as inputs to the same or other equations for state sequencing or cascaded logic. A total of 24 input channels are available to feed into the logic array in a Fast Function Block. Both true and complement of each input can be used by the product terms. These 24 input channels include any output feedback paths used by any of the equations, as well as inputs applied to a PLFFB9 symbol. The software automatically allocates one of the available logic array input channels whenever an output re-uses any of the equations. The total number of input variables used by all logic equations must not exceed 24, even though more external connections are provided.

There are 33 generic I/O pins (PIN1 – PIN33) so that all logic array input channels and output functions are accessible when no output feedbacks are used. Any of the symbol pins can be used to attach the incoming or outgoing signals. Pin direction is determined according to usage. Unused symbol pins must be left unconnected and their corresponding positions in the equation file pin list must indicate NC (not connected).

Pin Descriptions

PIN1 – PIN33 are generic I/O pins. Each pin can be used as one of the 24 Fast Function Block logic array inputs or nine function outputs.

PIN34 and PIN35 are clock inputs that trigger registers on Low-to-High transitions. They must be driven by a FastCLK buffer (BUFG) and cannot be used in any logic equations.

PULLDOWN

Resistor to GND for Input Pads

X3860

XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

PULLDOWN resistor elements are available in each XC4000 Input/Output Block (IOB). They are connected to input, output, or bidirectional pads to guarantee a logic Low level for nodes that might float.

PULLUP

Resistor to V_{CC} for Input PADs, Open-Drain, and 3-State Outputs

X3861

XC2000	XC3000	XC4000	XC7000
N/A	Primitive	Primitive	Primitive

PULLUP resistor elements are available in each XC3000 and XC4000 Input/Output Block (IOB). XC3000 IOBs only use PULLUP resistors on input pads. XC4000 IOBs connect PULLUP resistors to input, output, or bidirectional pads to guarantee a logic High level for nodes that might float.

The pull-up elements also establish a High logic level for open-drain elements and macros (DECODE, WAND, WORAND) or 3-state nodes (TBUF) when all the drivers are off.

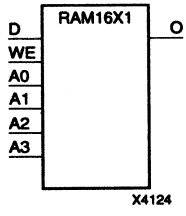
The buffer outputs are connected together as a wired-AND to form the output (O). When all the inputs are High, the output is off. To establish an output High level, a PULLUP resistor(s) is tied to output (O). One PULLUP resistor uses the least power, two pull-up resistors achieve the fastest Low-to-High speed.

To indicate two PULLUP resistors, append a DOUBLE parameter to the pull-up symbol attached to the output (O) node. Refer to the appropriate CAE tool interface user guide for details.

The PULLUP element is ignored in XC7000 designs. PULLUP is only supported for compatibility with FPGA designs. Internal 3-state nodes (from BUFE or BUFT) in EPLD designs are always pulled up when not driven.

RAM16X1

16-Deep by 1-Wide Static RAM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive*	N/A

* Not supported for XC4010D designs

RAM16X1 is a 16-word by 1-bit static read-write random-access memory. When the write enable (WE) is High, the data on the data input (D) is loaded into the word selected by the 4-bit address (A3 – A0). The data output (O) reflects the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or input data transitions. Address inputs must be stable before the High-to-Low WE transition for predictable performance.

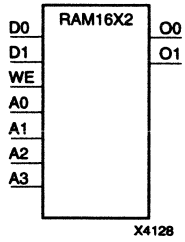
RAM16X1 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	**D**	**O**
0(read)	X	data
1(write)	D	data

data = word addressed by bits A3 – A0

RAM16X2

16-Deep by 2-Wide Static RAM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro*	N/A

* Not supported for XC4010D designs

RAM16X2 is a 16-word by 2-bit static read-write random-access memory. When the write enable (WE) is High, the data on data inputs (D1 – D0) is loaded into the word selected by the 4-bit address (A3 – A0). The data outputs (O1 – O0) reflect the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or data input transitions. Address inputs must be stable before the High-to-Low WE transition for predictable performance.

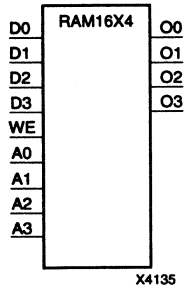
RAM16X2 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	**D1 – D0**	**O1 – O0**
0(read)	X	data
1(write)	D1 – D0	data

data = word addressed by bits A3 – A0

RAM16X4

16-Deep by 4-Wide Static Ram



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro*	N/A

* Not supported for XC4010D designs

RAM16X4 is a 16-word by 4-bit static read-write random-access memory. When the write enable (WE) is High, the data on data inputs (D3 – D0) is loaded into the word selected by the 4-bit address (A3 – A0). The data outputs (O3 – O0) reflect the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or data input transitions. Address inputs must be stable before the High-to-Low WE transition for predictable performance.

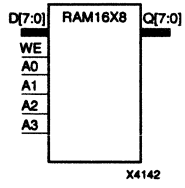
RAM16X4 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	**D3 – D0**	**O3 – O0**
0(read)	X	data
1(write)	D3 – D0	data

data = word addressed by bits A3 – A0

RAM16X8

16-Deep by 8-Wide Static RAM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro*	N/A

* Not supported for XC4010D designs

RAM16X8 is a 16-word by 8-bit static read-write random-access memory. When the write enable (WE) is High, the data on data inputs (D7 – D0) is loaded into the word selected by the 4-bit address (A3 – A0). The data outputs (O7 – O0) reflect the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or data input transitions. Address inputs must be stable before the High-to-Low WE transition for predictable performance.

RAM16X8 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	**D7 – D0**	**O7 – O0**
0(read)	X	data
1(write)	D7 – D0	data

data = word addressed by bits A3 – A0

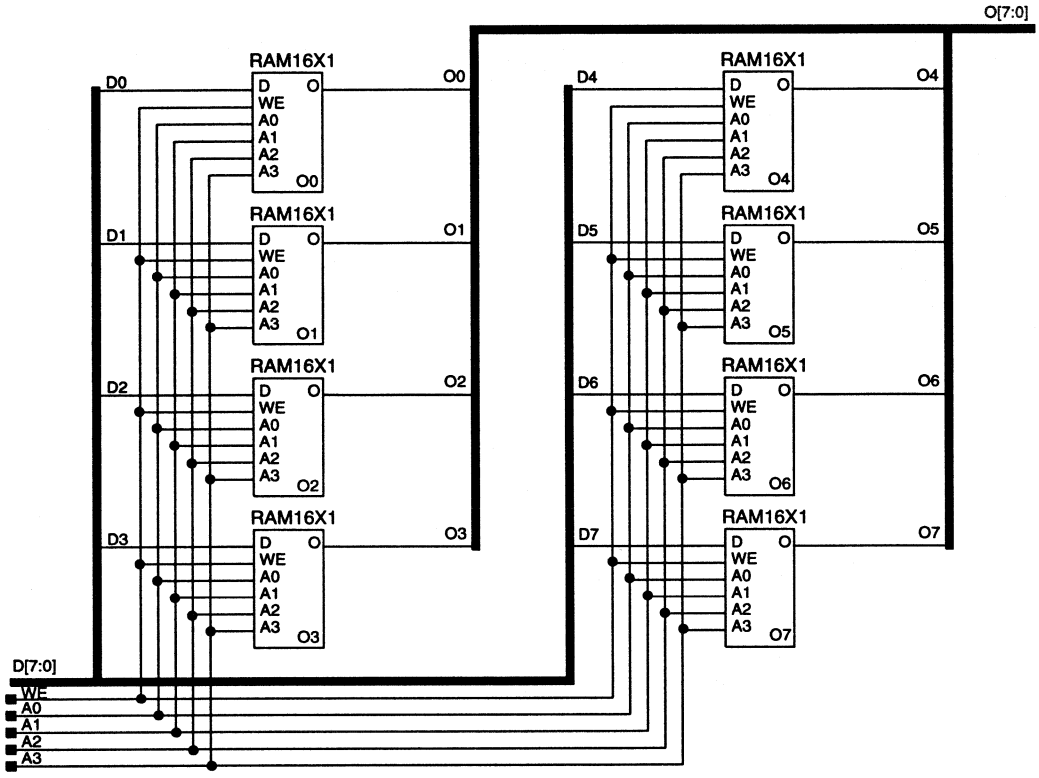
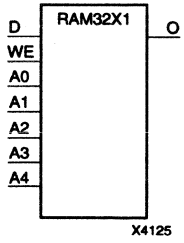


Figure 3-186 RAM16X8 XC4000 Implementation

RAM32X1

32-Deep by 1-Wide Static RAM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive*	N/A

* Not supported for XC4010D designs

RAM32X1 is a 32-word by 1-bit static read-write random-access memory. When the write enable (WE) is High, the data on the data input (D) is loaded into the word selected by the 5-bit address (A4 – A0). The data output (O) reflects the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or input data transitions. Address inputs must be stable before the High-to-Low WE transition for predictable performance.

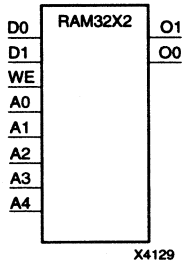
RAM32X1 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	**D**	**O**
0(read)	X	data
1(write)	D	data

data = word addressed by bits A4 — A0

RAM32X2

32-Deep by 2-Wide Static RAM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro*	N/A

* Not supported for XC4010D designs

RAM32X2 is a 32-word by 2-bit static read-write random-access memory. When the write enable (WE) is High, the data on the data inputs (D1 – D0) is loaded into the word selected by the address bits (A4 – A0). The data outputs (O1 – O0) reflect the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or input data transitions. Address inputs must be stable before the High-to- Low WE transition for predictable performance.

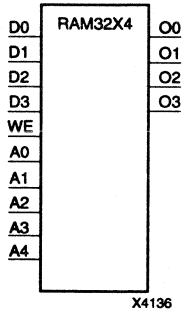
RAM32X2 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	**D1 – D0**	**O1 – O0**
0(read)	X	data
1(write)	D1 – D0	data

data = word addressed by bits A4 — A0

RAM32X4

32-Deep by 4-Wide Static RAM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro*	N/A

* Not supported for XC4010D designs

RAM32X4 is a 32-word by 4-bit static read-write random-access memory. When the write enable (WE) is High, the data on the data inputs (D3 – D0) is loaded into the word selected by the address bits (A4 – A0). The data outputs (O3 – O0) reflect the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or input data transitions. Address inputs must be stable before the High-to- Low WE transition for predictable performance.

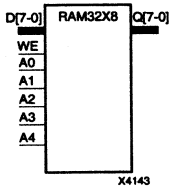
RAM32X4 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	D3 – D0	O3 – O0
0(read)	X	data
1(write)	D3 – D0	data

data = word addressed by bits A4 — A0

RAM32X8

32-Deep by 8-Wide Static RAM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro*	N/A

* Not supported for XC4010D designs

RAM32X8 is a 32-word by 8-bit static read-write random-access memory. When the write enable (WE) is High, the data on the data inputs (D7 – D0) is loaded into the word selected by the address bits (A4 – A0). The data outputs (O7 – O0) reflect the selected (addressed) word, whether WE is High or Low. When WE is Low, the RAM content is unaffected by address or input data transitions. The address inputs must be stable before the High-to- Low WE transition for predictable performance.

RAM32X8 cannot be initialized during configuration, only after configuration. Mode selection is shown in the following truth table.

Inputs		Outputs
WE(mode)	**D7 – D0**	**O7 – O0**
0(read)	X	data
1(write)	D7 – D0	data

data = word addressed by bits A4 — A0

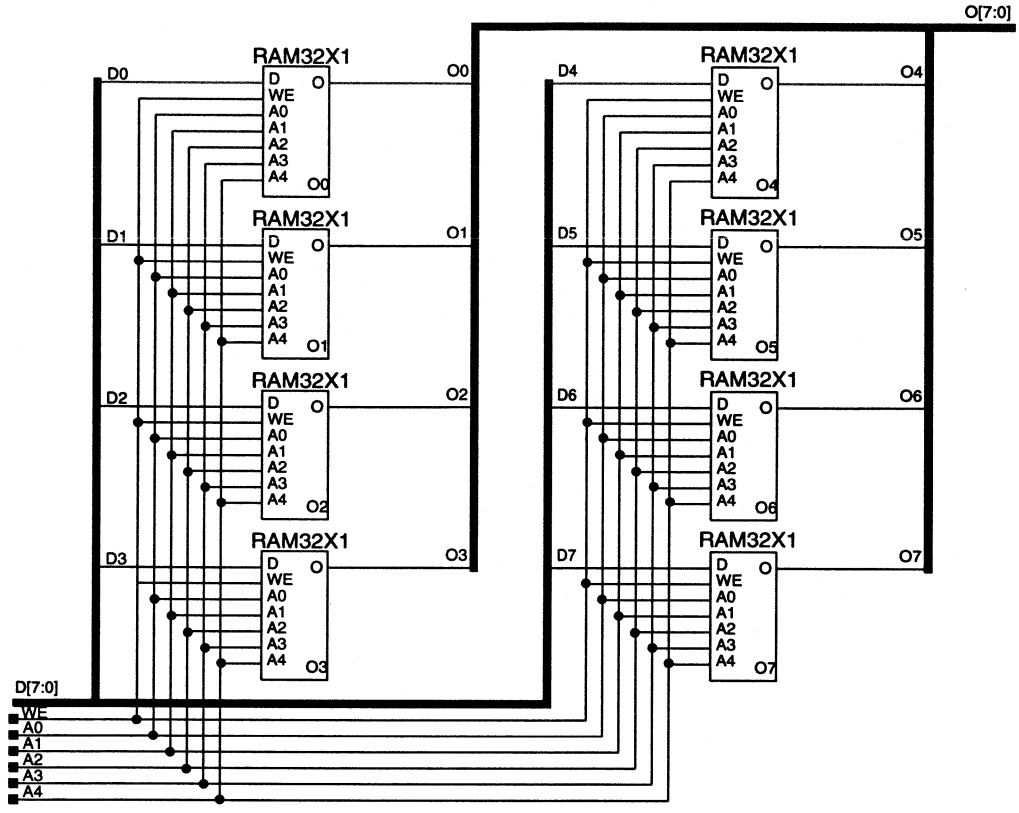
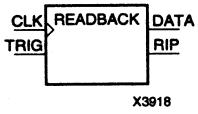


Figure 3-187 RAM32X8 XC4000 Implementation

READBACK

FPGA Bitstream Readback Controller



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Macro	N/A

The READBACK macro accesses the bitstream readback function. A Low-to-High transition on the TRIG input initiates the readback process. The readback data appears on the DATA output. The RIP (readback-in-progress) output remains High during the readback process. If you use the ReadAbort:Enable option in MakeBits, a High-to-Low transition on the TRIG input aborts the process. The signal on the CLK input clocks out the readback data; if no signal is connected to the CLK input, the internal CCLK is used. Set the ReadClk option in MakeBits to indicate the readback clock source.

Typically, READBACK inputs are sourced by device-external input pins and outputs drive device-external output pins. If you want external input and output pins, connect READBACK pins through IBUFs or OBUFs to pads, as with any I/O device. However, you can connect READBACK pins to device-internal logic instead. For details on the readback process, refer to the application note "Using the XC4000 Readback Capability" in *The Programmable Logic Data Book.*

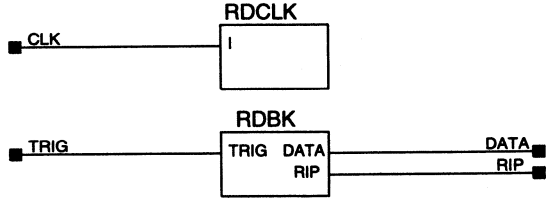
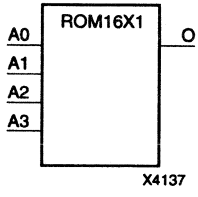


Figure 3-188 READBACK XC4000 Implementation

ROM16X1

16-Deep by 1-Wide ROM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive*	N/A

* Not supported for XC4010D designs

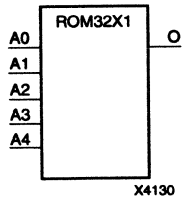
ROM16X1 is a 16-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 4-bit address (A3 – A0). The ROM is initialized to a known value during configuration with the INIT=*value* parameter. The *value* consists of four hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H. For example, the INIT=10A7 parameter produces the data stream

0001 0000 1010 0111

An error occurs if the INIT=*value* is not specified. Refer to the appropriate CAE tool interface user guide for details.

ROM32X1

32-Deep by 1-Wide ROM



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive*	N/A

* Not supported for XC4010D designs

ROM32X1 is a 32-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 5-bit address (A4 – A0). The ROM is initialized to a known value during configuration with the INIT=*value* parameter. The *value* consists of eight hexadecimal digits that are written into the ROM from the most-significant digit A=1FH to the least-significant digit A=00H. For example, the INIT=10A78F39 parameter produces the data stream

0001 0000 1010 0111 1000 1111 0011 1001

An error occurs if the INIT=*value* is not specified. Refer to the appropriate CAE tool interface user guide for details.

SOP

Sum Of Products

Name	XC2000	XC3000	XC4000	XC7000
SOP3 – SOP3B3	Macro	Macro	Macro	Primitive
SOP4 – SOP4B4	Macro	Macro	Macro	Primitive

Sum Of Products macros and primitives provide common logic functions by OR gating the outputs of two AND functions or the output of one AND function with one direct input. Variations of inverting and non-inverting inputs are available.

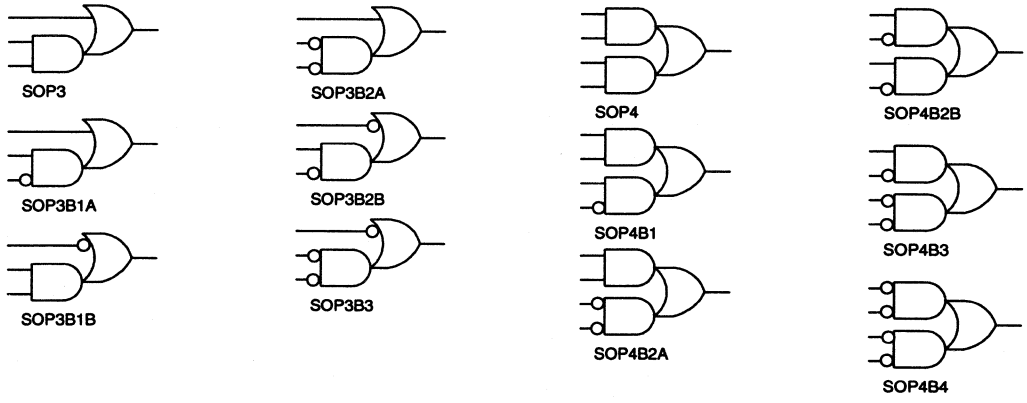
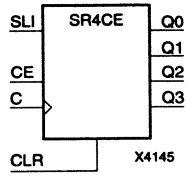


Figure 3-189 SOP Gate Representations

SR4CE

4-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR4CE is a 4-bit shift register with a shift-left serial input (SLI), parallel outputs (Q3 – Q0), and clock enable (CE) and asynchronous clear (CLR) inputs. The CLR input, when High, overrides all other inputs and resets the data outputs (Q3 – Q0) Low. When CE is High and CLR is Low, the data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and CLR is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the Q3 output of one stage to the SLI input of the next stage and connecting clock, CE, and CLR in parallel.

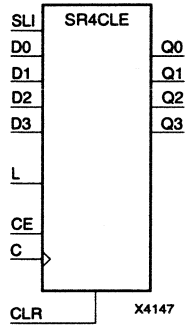
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs	
CLR	CE	SLI	C	Q0	Q3 – Q1
1	X	X	X	0	0
0	0	X	X	---No Change---	
0	1	1	↑	1	qn-1
0	1	0	↑	0	qn-1

qn-1 = state of referenced output one set-up time prior to active clock transition

SR4CLE

4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR4CLE is a 4-bit shift register with a shift-left serial input (SLI), parallel inputs (D3 – D0), parallel outputs (Q3 – Q0), and three control inputs – clock enable (CE), load enable (L), and asynchronous clear (CLR).The register ignores clock transitions when L and CE are Low. The asynchronous CLR, when High, overrides all other inputs and resets the data outputs (Q3 – Q0) Low. When L is High and CLR is Low, data on the D3 – D0 inputs is loaded into the corresponding Q3 – Q0 bits of the register. When CE is High and L and CLR are Low, data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and CLR are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth).

Registers can be cascaded by connecting the Q3 output of one stage to the SLI input of the next stage and connecting clock, CE, L, and CLR inputs in parallel.

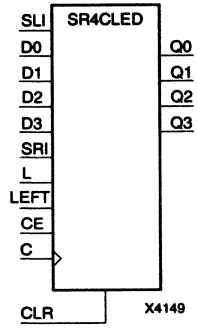
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs	
CLR	L	CE	SLI	D3 – D0	C	Q0	Q3 – Q1
1	X	X	X	X	X	0	0
0	1	X	X	D3 – D0	↑	d0	dn
0	0	1	SLI	X	↑	SLI	qn-1
0	0	0	X	X	X	--No Change--	

dn or qn-1 = state of referenced input or output one set-up time prior to active clock transition

SR4CLED

4-Bit Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR4CLED is a 4-bit shift register with shift-left (SLI) and shift-right (SRI) serial inputs, four parallel inputs (D3 – D0), and four control inputs – clock enable (CE), load enable (L), shift left/right (LEFT), and asynchronous clear (CLR). The register ignores clock transitions when CE and L are Low. The asynchronous clear, when High, overrides all other inputs and resets the data outputs (Q3 – Q0) Low. When L is High and CLR is Low, the data on the D3 – D0 inputs is loaded into the corresponding Q3 – Q0 bits of the register. When CE is High and L and CLR are Low, data is shifted right or left, depending on the state of the LEFT input. If LEFT is High, data on the SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on the SRI is loaded into Q3 during the Low-to-High clock transition and shifted right (to Q2, Q1, and so forth) during subsequent clock transitions. The truth table indicates the state of the Q3 – Q0 outputs under all input conditions.

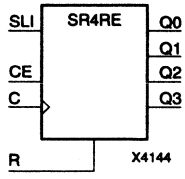
The register is asynchronously reset, outputs Low, when power is applied, or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs								Outputs		
CLR	L	CE	LEFT	SLI	SRI	D3 – D0	C	Q0	Q3	Q2 – Q1
1	X	X	X	X	X	X	X	0	0	0
0	1	X	X	X	X	D3 – D0	↑	d0	d3	dn
0	0	0	X	X	X	X	X	----No Change-----		
0	0	1	1	SLI	X	X	↑	SLI	q2	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1

dn, qn-1 or qn+1 = state of referenced input one set-up time prior to active clock transition

SR4RE

4-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR4RE is a 4-bit shift register with shift-left serial input (SLI), parallel outputs (Q3 – Q0), clock enable (CE), and synchronous reset (R) inputs. The R input, when High, overrides all other inputs and resets the data outputs (Q3 – Q0) Low. When CE is High and R is Low, the data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and R is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the Q3 output of one stage to the SLI input of the next stage and connecting clock, CE, and R in parallel.

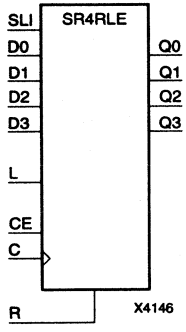
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs	
R	CE	SLI	C	Q0	Q3 – Q1
1	X	X	↑	0	0
0	0	X	X	---No Change---	
0	1	1	↑	1	qn-1
0	1	0	↑	0	qn-1

qn-1 = state of referenced output one set-up time prior to active clock transition

SR4RLE

4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR4RLE is a 4-bit shift register with shift-left serial input (SLI), four parallel inputs (D3 – D0), four parallel outputs (Q3 – Q0), and three control inputs – clock enable (CE), load enable (L), and synchronous reset (R). The register ignores clock transitions when L and CE are Low. The synchronous R, when High, overrides all other inputs and resets the data outputs (Q3 – Q0) Low. When L is High and R is Low, data on the D3 – D0 inputs is loaded into the corresponding Q3 – Q0 bits of the register. When CE is High and L and R are Low, data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and R are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth).

Registers can be cascaded by connecting the Q3 output of one stage to the SLI input of the next stage and connecting clock, CE, L, and R inputs in parallel.

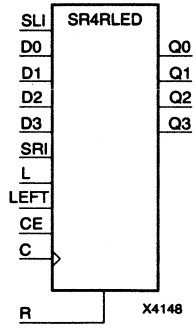
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs	
R	L	CE	SLI	D3 – D0	C	Q0	Q3 – Q1
1	X	X	X	X	↑	0	0
0	1	X	X	D3 – D0	↑	d0	dn
0	0	1	SLI	X	↑	SLI	qn-1
0	0	0	X	X	X	--No Change--	

dn or qn-1 = state of referenced input one set-up time prior to active clock transition

SR4RLED

4-Bit Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR4RLED is a 4-bit shift register with shift-left (SLI) and shift-right (SRI) serial inputs, four parallel inputs (D3 – D0), and four control inputs – clock enable (CE), load enable (L), shift left/right (LEFT), and synchronous reset (R). The register ignores clock transitions when CE and L are Low. The synchronous R, when High, overrides all other inputs and resets the data outputs (Q3 – Q0) Low. When L is High and R is Low, the data on the D3 – D0 inputs is loaded into the corresponding Q3 – Q0 bits of the register. When CE is High and L and R are Low, data is shifted right or left, depending on the state of the LEFT input. If LEFT is High, data on SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on the SRI is loaded into Q3 during the Low-to-High clock transition and shifted right (to Q2, Q1, and so forth) during subsequent clock transitions. The truth table indicates the state of the Q3 – Q0 outputs under all input conditions.

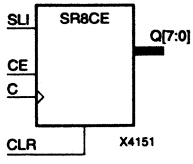
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs								Outputs		
R	L	CE	LEFT	SLI	SRI	D3 – D0	C	Q0	Q3	Q2 – Q1
1	X	X	X	X	X	X	↑	0	0	0
0	1	X	X	X	X	D3 – D0	↑	d0	d3	dn
0	0	0	X	X	X	X	X	----No Change-----		
0	0	1	1	SLI	X	X	↑	SLI	q2	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1

dn, qn-1 or qn+1 = state of referenced input one set-up time prior to active clock transition

SR8CE

8-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR8CE is an 8-bit shift-left serial input (SLI), parallel output (Q7 – Q0) shift register with clock enable (CE) and asynchronous clear (CLR) inputs. The CLR input, when High, overrides all other inputs and resets the data outputs (Q7 – Q0) Low. When CE is High and CLR is Low, the data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and CLR is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the Q7 output of one stage to the SLI input of the next stage and connecting clock, CE, and CLR in parallel.

The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs	
CLR	CE	SLI	C	Q0	Q7 – Q1
1	X	X	X	0	0
0	0	X	X	--No Change---	
0	1	1	↑	1	qn-1
0	1	0	↑	0	qn-1

qn-1 = state of referenced output one set-up time prior to active clock transition

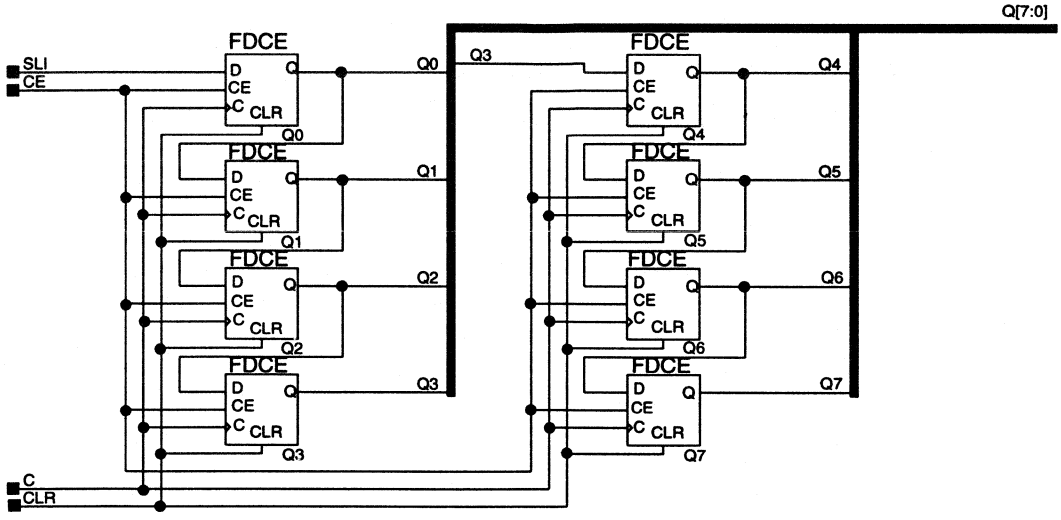
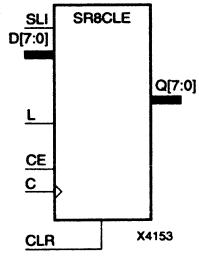


Figure 3-190 SR8CE XC2000/3000/4000 Implementation

SR8CLE

8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR8CLE is an 8-bit shift register with a shift-left serial input (SLI), parallel inputs (D7 – D0), parallel outputs (Q7 – Q0), and three control inputs – clock enable (CE), load enable (L) and asynchronous clear (CLR). The register ignores clock transitions when L and CE are Low. The asynchronous CLR, when High, overrides all other inputs and resets the data outputs (Q7 – Q0) Low. When L is High and CLR is Low, data on the D7 – D0 inputs is loaded into the corresponding Q7 – Q0 bits of the register. When CE is High and L and CLR are Low, data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and CLR are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth).

Registers can be cascaded by connecting the Q7 output of one stage to the SLI input of the next stage and connecting clock, CE, L, and CLR inputs in parallel.

The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs	
CLR	L	CE	SLI	D7 – D0	C	Q0	Q7 – Q1
1	X	X	X	X	X	0	0
0	1	X	X	D7 – D0	↑	d0	dn
0	0	1	SLI	X	↑	SLI	qn-1
0	0	0	X	X	X	--No Change--	

dn or qn-1 = state of referenced input one set-up time prior to active clock transition

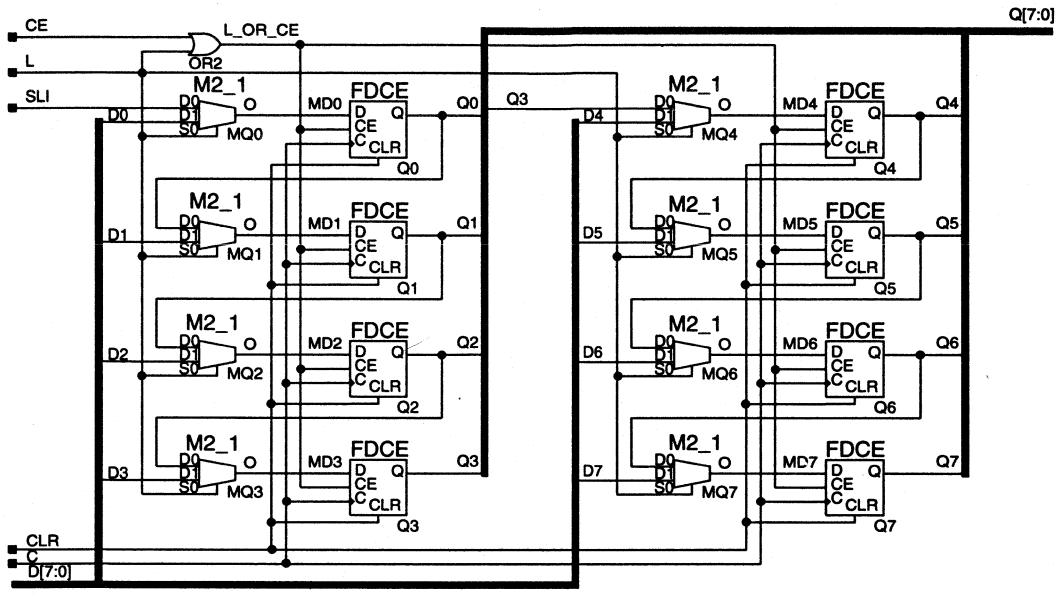
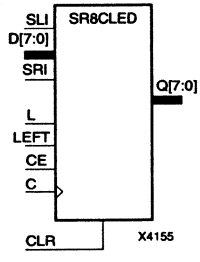


Figure 3-191 SR8CLE XC2000/3000/4000 Implementation

SR8CLED

8-Bit Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR8CLED is an 8-bit shift register with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D7 – D0), and four control inputs – clock enable (CE), load enable (L), shift left/right (LEFT), and asynchronous clear (CLR). The register ignores clock transitions when CE and L are Low. The asynchronous CLR, when High, overrides all other inputs and resets the data outputs (Q7 – Q0) Low. When L is High and CLR is Low, data on the D7 – D0 inputs is loaded into the corresponding Q7 – Q0 bits of the register. When CE is High and L and CLR are Low, data is shifted right or left depending on the state of the LEFT input. If LEFT is High, data on the SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on the SRI is loaded into Q7 during the Low-to-High clock transition and shifted right (to Q6, Q5, and so forth) during subsequent clock transitions. The truth table indicates the state of the Q7 – Q0 outputs under all input conditions.

The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs								Outputs		
CLR	L	CE	LEFT	SLI	SRI	D7 – D0	C	Q0	Q7	Q6 – Q1
1	X	X	X	X	X	X	X	0	0	0
0	1	X	X	X	X	D7 – D0	↑	d0	d7	dn
0	0	0	X	X	X	X	X	-----No Change-----		
0	0	1	1	SLI	X	X	↑	SLI	q6	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1

dn, qn-1 or qn+1 = state of referenced input one set-up time prior to active clock transition

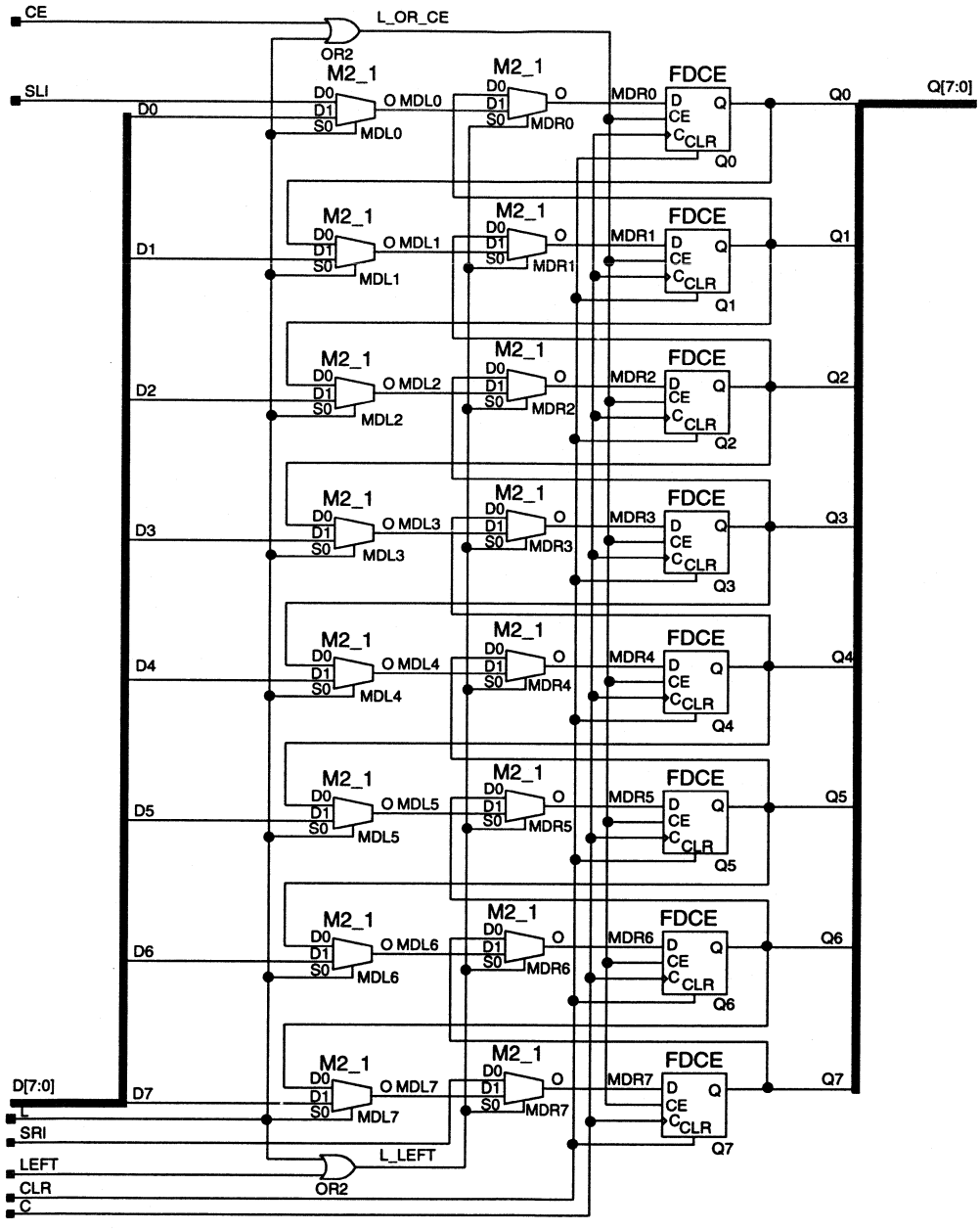
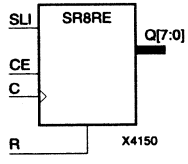


Figure 3-192 SR8CLED XC2000/3000/4000 Implementation

SR8RE

8-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR8RE is an 8-bit shift-left serial input (SLI), parallel output (Q7 – Q0) shift register with clock enable (CE) and synchronous reset (R) inputs. The R input, when High, overrides all other inputs and resets the data outputs (Q7 – Q0) Low. When CE is High and R is Low, the data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and R is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the Q7 output of one stage to the SLI input of the next stage and by connecting clock, CE, and R in parallel.

The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs	
R	CE	SLI	C	Q0	Q7 – Q1
1	X	X	↑	0	0
0	0	X	X	--No Change---	
0	1	1	↑	1	qn-1
0	1	0	↑	0	qn-1

qn-1 = state of referenced output one set-up time prior to active clock transition

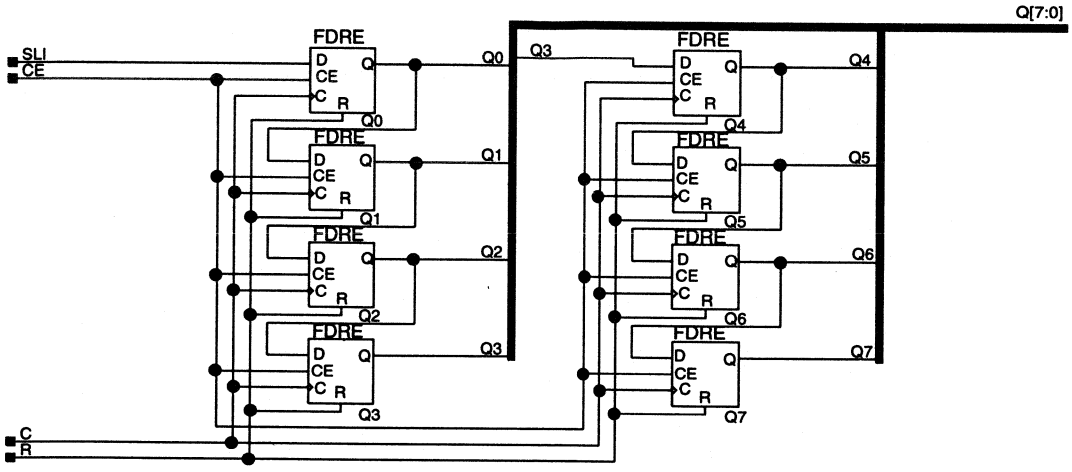
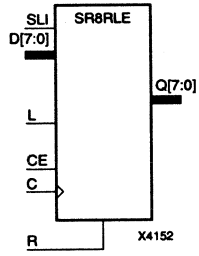


Figure 3-193 SR8RE XC2000/3000/4000 Implementation

SR8RLE

8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR8RLE is an 8-bit shift register with shift-left serial input (SLI), parallel inputs (D7 – D0), parallel outputs (Q7 – Q0), and three control inputs – clock enable (CE), load enable (L), and synchronous reset (R). The register ignores clock transitions when L and CE are Low. The synchronous R, when High, overrides all other inputs and resets the data outputs (Q7 – Q0) Low. When L is High and R is Low, data on the D7 – D0 inputs is loaded into the corresponding Q7 – Q0 bits of the register. When CE is High and L and R are Low, data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and R are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). Registers can be cascaded by connecting the Q7 output of one stage to the SLI input of the next stage and connecting clock, CE, L, and R inputs in parallel.

The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs	
R	L	CE	SLI	D7 – D0	C	Q0	Q7 – Q1
1	X	X	X	X	↑	0	0
0	1	X	X	D7 – D0	↑	d0	dn
0	0	1	SLI	X	↑	SLI	qn-1
0	0	0	X	X	X	--No Change--	

dn or qn-1 = state of referenced input one set-up time prior to active clock transition

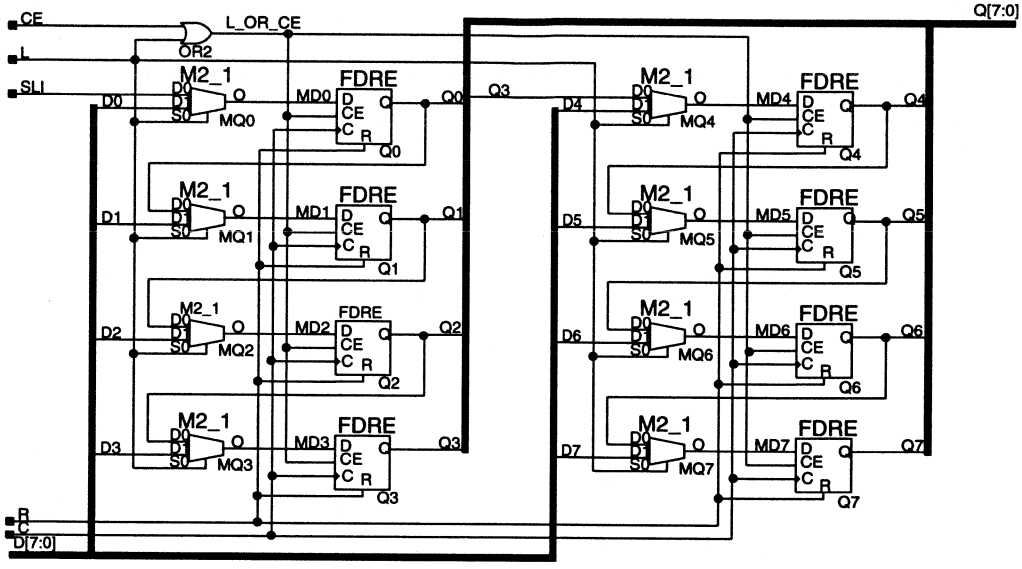
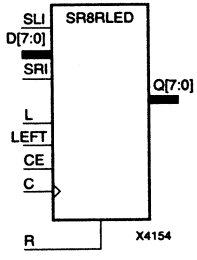


Figure 3-194 SR8RLE XC2000/3000/4000 Implementation

SR8RLED

8-Bit Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR8RLED is an 8-bit shift register with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D7 – D0), and four control inputs – clock enable (CE), load enable (L), shift left/right (LEFT), and synchronous reset (R). The register ignores clock transitions when CE and L are Low. The synchronous R, when High, overrides all other inputs and resets the data outputs (Q7 – Q0) Low. When L is High and R is Low, the data on the D7 – D0 inputs is loaded into the corresponding Q7 – Q0 bits of the register. When CE is High and L and R are Low, data is shifted right or left depending on the state of the LEFT input. If LEFT is High, data on SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on SRI is loaded into Q7 bit during the Low-to-High clock transition and shifted right (to Q6, Q5, and so forth) during subsequent clock transitions. The truth table indicates the state of the Q7 – Q0 outputs under all input conditions.

The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs								Outputs		
R	L	CE	LEFT	SLI	SRI	D7– D0	C	Q0	Q7	Q6 – Q1
1	X	X	X	X	X	X	↑	0	0	0
0	1	X	X	X	X	D7 – D0	↑	d0	d7	dn
0	0	0	X	X	X	X	X	----No Change-----		
0	0	1	1	SLI	X	X	↑	SLI	q6	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1

dn, qn-1 or qn+1 = state of referenced input one set-up time prior to active clock transition

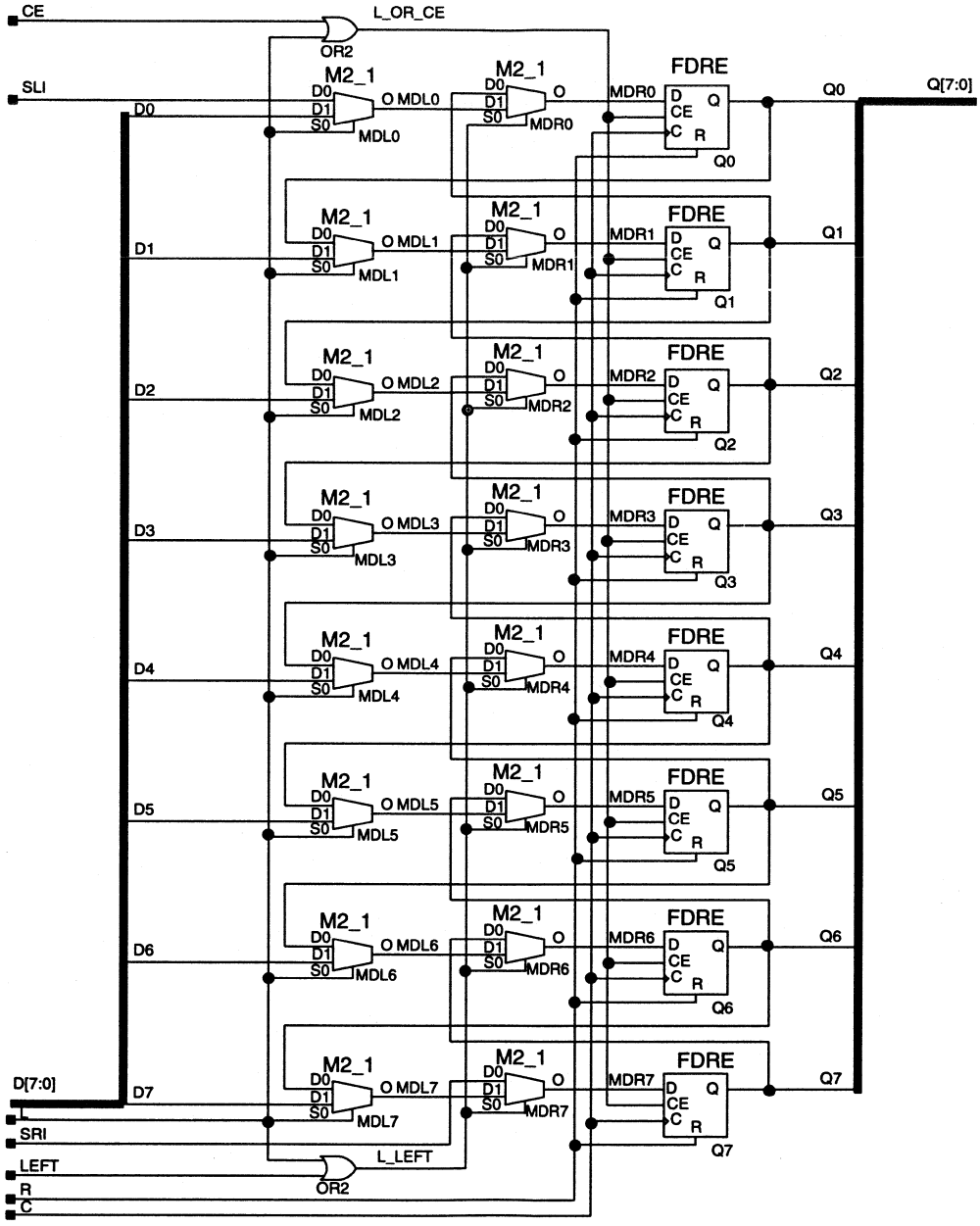
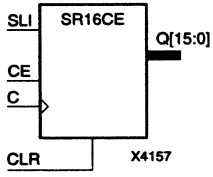


Figure 3-195 SR8RLED XC2000/3000/4000 Implementation

SR16CE

16-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR16CE is a 16-bit a shift-left serial input (SLI), parallel outputs (Q15 – Q0) shift register with clock enable (CE) and asynchronous clear (CLR) inputs. The CLR input, when High, overrides all other inputs and resets the data outputs (Q15 – Q0) Low. When CE is High and CLR is Low, the data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and CLR is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). The register ignores clock transitions when CE is Low. Registers can be cascaded by connecting the Q15 output of one stage to the Shift Left Input (SLI) of the next stage and connecting clock, CE, and CLR in parallel.

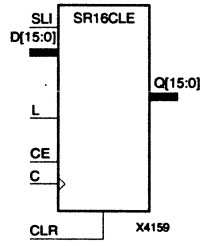
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs	
CLR	CE	SLI	C	Q0	Q15 – Q1
1	X	X	X	0	0
0	0	X	X	---No Change---	
0	1	1	↑	1	qn-1
0	1	0	↑	0	qn-1

qn-1 = state of referenced output one set-up time prior to active clock transition

SR16CLE

16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR16CLE is a 16-bit shift register with shift-left serial input (SLI), parallel inputs (D15 – D0), parallel outputs (Q15 – Q0), and three control inputs – clock enable (CE), load enable (L), and asynchronous clear (CLR). The register ignores clock transitions when L and CE are Low. The asynchronous CLR, when High, overrides all other inputs and resets the data outputs (Q15 – Q0) Low. When L is High and CLR is Low, data on the D15 – D0 inputs is loaded into the corresponding Q15 – Q0 bits of the register. When CE is High and L and CLR are Low, data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and CLR are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). Registers can be cascaded by connecting the Q15 output of one stage to the Shift Left Input (SLI) of the next stage and connecting clock, CE, L, and CLR inputs in parallel.

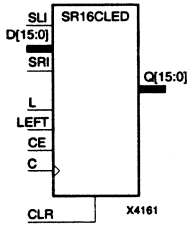
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs	
CLR	L	CE	SLI	D15 – D0	C	Q0	Q15 – Q1
1	X	X	X	X	X	0	0
0	1	X	X	D15 – D0	↑	d0	dn
0	0	1	SLI	X	↑	SLI	qn-1
0	0	0	X	X	X	--No Change--	

dn or qn-1 = state of referenced input one set-up time prior to active clock transition

SR16CLED

16-Bit Shift Register with Clock Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR16CLED is a 16-bit shift register with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D15 – D0), and four control inputs – clock enable (CE), load enable (L), shift left/right (LEFT), and asynchronous clear (CLR). The register ignores clock transitions when CE and L are Low. The asynchronous CLR, when High, overrides all other inputs and resets the data outputs (Q15 – Q0) Low. When L is High and CLR is Low the data on the D15 – D0 inputs is loaded into the corresponding Q15 – Q0 bits of the register. When CE is High and L and CLR are Low, data is shifted right or left depending on the state of the LEFT input. If LEFT is High, data on SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on SRI is loaded into Q15 during the Low-to-High clock transition and shifted right (to Q14, Q13, and so forth) during subsequent clock transitions. The truth table indicates the state of the Q15 – Q0 outputs under all input conditions.

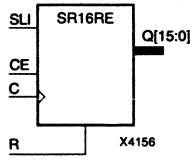
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs								Outputs		
CLR	L	CE	LEFT	SLI	SRI	D15 – D0	C	Q0	Q15	Q14 – Q1
1	X	X	X	X	X	X	X	0	0	0
0	1	X	X	X	X	D15 – D0	↑	d0	d15	dn
0	0	0	X	X	X	X	X	----No Change----		
0	0	1	1	SLI	X	X	↑	SLI	q14	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1

dn, qn-1 or qn+1 = state of referenced input one set-up time prior to active clock transition

SR16RE

16-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR16RE is a 16-bit shift-left serial input (SLI), parallel output (Q15 – Q0) shift register with clock enable (CE) and synchronous reset (R) inputs. The R input, when High, overrides all other inputs and resets the data outputs (Q15 – Q0) Low. When CE is High and R is Low, the data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and R is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). The register ignores clock transitions when CE is Low. Registers can be cascaded by connecting the Q15 output of one stage to the SLI input of the next stage and connecting clock, C, and R in parallel.

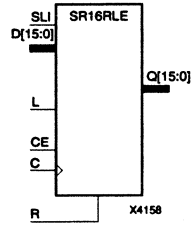
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs				Outputs	
R	CE	SLI	C	Q0	Q15 – Q1
1	X	X	↑	0	0
0	0	X	X	---No Change---	
0	1	1	↑	1	qn-1
0	1	0	↑	0	qn-1

qn-1 = state of referenced output one set-up time prior to active clock transition

SR16RLE

16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR16RLE is a 16-bit shift register with shift-left serial input (SLI), parallel inputs (D15 – D0), parallel outputs (Q15 – Q0), and control inputs – clock enable (CE), load enable (L), and synchronous reset (R). The register ignores clock transitions when L and CE are Low. The synchronous R, when High, overrides all other inputs and resets the data outputs (Q15 – Q0) Low. When L is High and R is Low, data on the data D15 – D0 inputs is loaded into the corresponding Q15 – Q0 bits of the register. When CE is High and L and R are Low, data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and R are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). Registers can be cascaded by connecting the Q15 output of one stage to the Shift Left Input (SLI) of the next stage and connecting clock, CE, L, and R inputs in parallel.

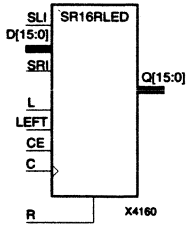
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs						Outputs	
R	L	CE	SLI	D15 – D0	C	Q0	Q15 – Q1
1	X	X	X	X	↑	0	0
0	1	X	X	D15 – D0	↑	d0	dn
0	0	1	SLI	X	↑	SLI	qn-1
0	0	0	X	X	X	--No Change--	

dn or qn-1 = state of referenced input one set-up time prior to active clock transition

SR16RLED

16-Bit Shift Register with Clock Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

SR16RLED is a 16-bit shift register with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D15 – D0), and four control inputs – clock enable (CE), load enable (L), shift left/right (LEFT), and synchronous reset (R). The register ignores clock transitions when CE and L are Low. The synchronous R, when High, overrides all other inputs and resets the data Q15 – Q0 outputs Low. When L is High and R is Low, the data on the D15 – D0 inputs is loaded into the corresponding Q15 – Q0 bits of the register. When CE is High and L and R are Low, data is shifted right or left depending on the state of the LEFT input. If LEFT is High, data on SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on SRI is loaded into Q15 during the Low-to-High clock transition and shifted right (to Q14, Q13, and so forth) during subsequent clock transitions. The truth table indicates the state of the Q15 – Q0 outputs under all input conditions.

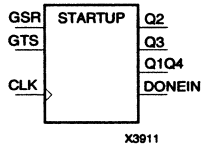
The register is asynchronously reset, outputs Low, when power is applied or when Global Reset (XC2000, XC3000) or Global Set/Reset (XC4000) is active. GR is active-Low; the GSR active level is programmable.

Inputs								Outputs		
R	L	CE	LEFT	SLI	SRI	D15 – D0	C	Q0	Q15	Q14 – Q1
1	X	X	X	X	X	X	↑	0	0	0
0	1	X	X	X	X	D15 – D0	↑	d0	d15	dn
0	0	0	X	X	X	X	X	----No Change-----		
0	0	1	1	SLI	X	X	↑	SLI	q14	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1

dn, qn-1 or qn+1 = state of referenced input one set-up time prior to active clock transition

STARTUP

User Interface to Global Clock, Reset, and 3-State Controls



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

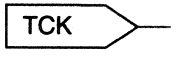
The STARTUP macro is used for Global Set/Reset, global 3-state control, and the user configuration clock. The Global Set/Reset (GSR) input, when High, sets or resets every flip-flop in the device, depending on the initialization state (S or R) of the flip-flop. Following configuration, the global 3-state control (GTS), when High, forces all the IOB outputs into High impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

The configuration clock input (CLK) must be connected to a user clock if the start-up of the device is synchronized with the user clock. Also, "user clock" must be selected in the MakeBits program.

The STARTUP outputs (Q2, Q3, Q1Q4, and DONEIN) display the progress/status of the start-up process following the configuration.

TCK

Boundary-Scan Test Clock Input Pad



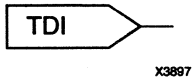
XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The TCK input pad is connected to the boundary-scan test clock, which shifts the serial data and instructions into and out of the boundary-scan data registers. The function of the TCK pad is device configuration dependent and can be used as follows.

- During configuration TCK is connected to the boundary-scan logic.
- After configuration, if boundary-scan is not used, the TCK pad is unrestricted and can be used by the XACT routing tools as an input/output pad.
- After configuration, if boundary-scan is used, the TCK pad can be used for user-logic input by connecting it directly to the user logic.

TDI

Boundary-Scan Test Data Input Pad



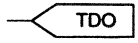
XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The TDI input pad is connected to the boundary-scan TDI input. It loads instructions and data on the Low-to-High TCK transition. The function of the TDI pad is device configuration dependent and can be used as follows.

- During configuration, TDI is connected to the boundary-scan logic.
- After configuration, if boundary-scan is not used, the TDI pad is unrestricted and can be used by the XACT routing tools as an input/output pad.
- After configuration, if boundary-scan is used, the TDI pad can be used for user-logic input by connecting the TDI pad directly to the user logic.

TDO

Boundary-Scan Data Output Pad



X3899

XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The TDO data output pad is connected to the boundary-scan TDO output. It is connected to the external circuit to provide the boundary-scan data for each Low-to-High TCK transition. The function of the TDO pad is device configuration dependent and can be used as follows.

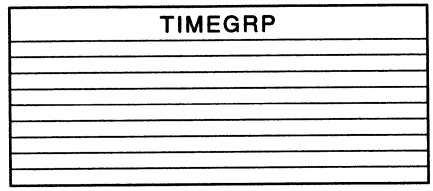
- During configuration, TDO is connected to the boundary-scan logic.
- After configuration, if boundary-scan is not used, the TDO pad can be used as a bidirectional 3-state I/O pad by the XACT routing tools.
- After configuration, if boundary-scan is used, the TDO pad is still used as an output from the boundary-scan logic.

TIMEGRP

Schematic-Level Table of Basic Timing Specification Groups

XC2000	XC3000	XC4000	XC7000
N/A	Primitive	Primitive	N/A

The TIMEGRP primitive table defines timing groups used in "from-to" TIMESPEC statements in terms of other groups. The TIMEGRP table is shown in the following figure.



These groups can include predefined groups, such as "ffs," groups created by using TNM attributes, such as TNM-reg on schematics, and other groups defined by a statement in the TIMEGRP symbol.

The following example statement defines groups in a TIMEGRP symbol.

```
=all_but_regs=ffs:except:regs
```

The table can contain up to 8 statements of any character length, but only 30 characters are displayed in the symbol.

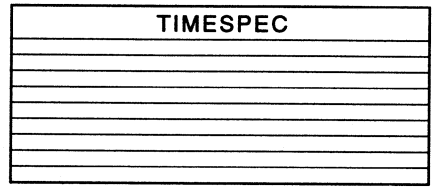
TIMESPEC

Schematic-Level Timing Requirement Table

XC2000	XC3000	XC4000	XC7000
N/A	Primitive	Primitive	Primitive*

* ignored in XC7000 designs

The TIMESPEC primitive is a table that specifies up to eight timing attributes (TS). TS attributes can be any length, but only 30 characters are displayed in the TIMESPEC window. The TIMESPEC table is displayed in the follow figure.



X3866

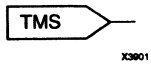
XC4000 OrCAD Only Schematic-Level Timing Requirement Signal Tag

The TS Signal Tag or parameter attaches timing attributes to nets in the schematic.

Refer to the appropriate CAE tool interface user guide for details about using the TIMESPEC primitive and TS Signal Tag.

TMS

Boundary-Scan Test Mode Select Input Pad



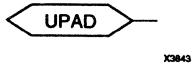
XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

The TMS input pad is connected to the boundary-scan TMS input. It determines which boundary-scan operation is performed. The function of the TMS pad is device configuration dependent and can be used as follows.

- During configuration, TMS is connected to the boundary-scan logic.
- After configuration, if boundary-scan is not used, the TMS pad is unrestricted and can be used by the XACT routing tools as an input/output pad.
- After configuration, if boundary-scan is used, the TMS pad can be used for user-logic input by connecting the TMS pad directly to the user logic.

UPAD

Connects the I/O Node of an IOB to the Internal PLD Circuit

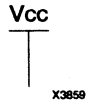


XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	Primitive	Primitive

A UPAD allows the use of any unbonded IOBs in a device. It is used the same way as a IOPAD, except that the signal output is not visible on any external device pins.

VCC

VCC-Connection Signal Tag



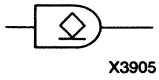
XC2000	XC3000	XC4000	XC7000
Primitive	Primitive	Primitive	Primitive

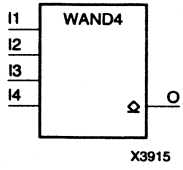
The V_{CC} signal tag or parameter forces a net or input function to a logic High level. A net tied to V_{CC} cannot have any other source.

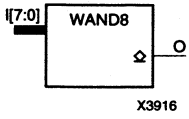
When the placement and routing software (APR for XC2000, XC3000; PPR for XC4000; or FITNET for XC7000) encounters a net or input function tied to V_{CC}, it removes any logic that is disabled by the V_{CC} signal. The V_{CC} signal is only implemented when the disabled logic cannot be removed.

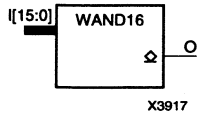
WAND1, WAND4, WAND8, and WAND16

Open-Drain Buffers









Name	XC2000	XC3000	XC4000	XC7000
WAND1	N/A	N/A	Primitive	N/A
WAND4, WAND8, WAND16	N/A	N/A	Macro	N/A

WAND1, WAND4, WAND8, and WAND16 are single and multiple open-drain buffers. Each buffer has an input (I) and an open-drain output (O). When any of the inputs is Low, the output is Low. When all the inputs are High, the output is off. To obtain a High output, add pull-up resistors to the output (O). One pull-up resistor uses the least power, and two pull-up resistors achieve the fastest Low-to-High transition.

To indicate two pull-up resistors, add a DOUBLE parameter to the pull-up symbol attached to the output (O) node. Refer to the appropriate CAE tool interface user guide for details.

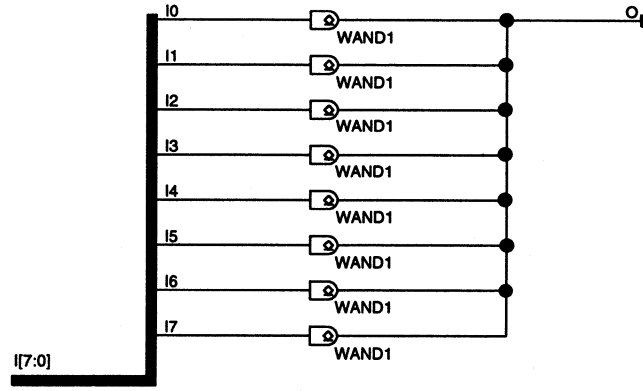
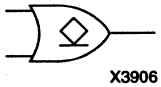


Figure 3-196 WAND8 XC4000 Implementation

WOR2AND

2-Input OR Gate with Wired-AND Open-Drain Buffer Output



XC2000	XC3000	XC4000	XC7000
N/A	N/A	Primitive	N/A

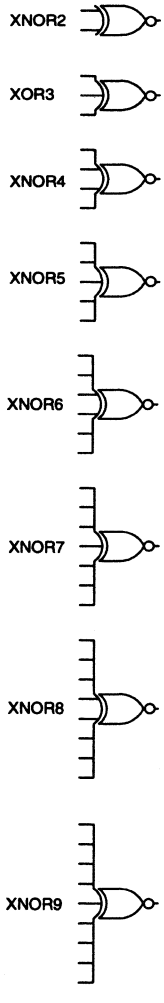
WOR2AND is a 2-input (I1 and I2) OR gate/buffer with an open-drain output (O). It is used in bus applications by tying multiple open-drain outputs together. When both inputs (I1 and I2) are Low, the output (O) is Low. When either input is High, the output is off; wor2and cannot source or sink current. To establish an output High level, tie a pull-up resistor(s) to the output (O). One pull-up resistor uses the least power, two pull-up resistors achieve the fastest Low-to-High speed.

To indicate two pull-up resistors, append a DOUBLE parameter to the pull-up symbol attached to the output (O) node. Refer to the appropriate CAE tool interface user guide for details.

XNOR

2- to 9-Input XNOR Gates with Non-Inverted Inputs

Name	XC2000	XC3000	XC4000	XC7000
XNOR3 – XNOR4	Primitive	Primitive	Primitive	Primitive
XNOR5	Macro	Primitive	Primitive	Primitive
XNOR6 – XNOR9	Macro	Macro	Macro	Primitive*



* XNOR7 – XNOR9 not supported for XC7336 designs

The XNOR function is performed in the Configurable Logic Block (CLB) function generators in XC2000, XC3000, and XC4000. XNOR functions of up to nine inputs are available. All inputs are non-inverting. Because each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.

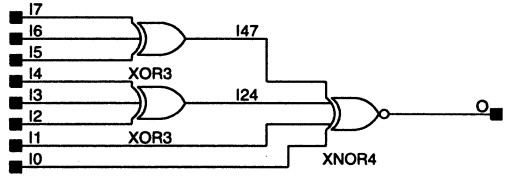


Figure 3-197 XNOR8 XC2000 Implementation

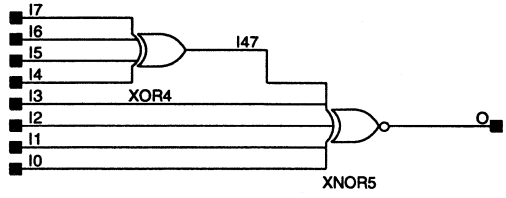


Figure 3-198 XNOR8 XC3000 Implementation

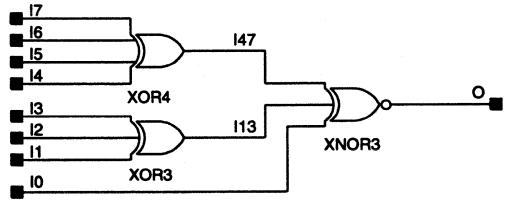
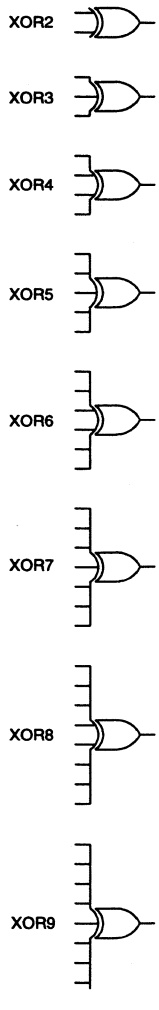


Figure 3-199 XNOR8 XC4000 Implementation

XOR

2- to 9-Input XOR Gates with Non-Inverted Inputs

Name	XC2000	XC3000	XC4000	XC7000
XOR2 – XOR4	Primitive	Primitive	Primitive	Primitive
XOR5	Macro	Primitive	Primitive	Primitive
XOR6 – XOR9	Macro	Macro	Macro	Primitive*



* XOR7 – XOR9 not supported for XC7336 designs

The XOR function is performed in the Configurable Logic Block (CLB) function generators for XC2000, XC3000, and XC4000. XOR functions of up to nine inputs are available. All inputs are non-inverting. Because each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.

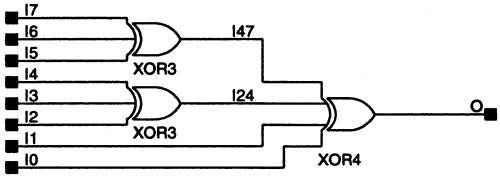


Figure 3-200 XOR8 XC2000 Implementation

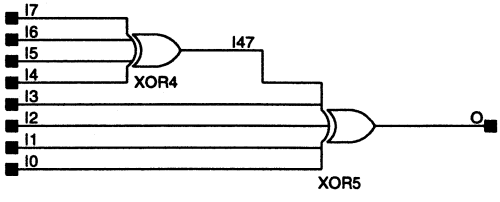


Figure 3-201 XOR8 XC3000 Implementation

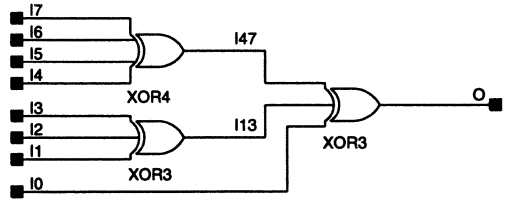
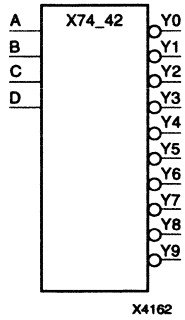


Figure 3-202 XOR8 XC4000 Implementation

X74_42

4- to 10-Line BCD-to-Decimal Decoder with Active-Low Outputs



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_42 decodes the 4-bit BCD number on the data inputs (A – D). Only one of the ten outputs (Y9 – Y0) is active (Low) at a time, which reflects the decimal equivalent of the BCD number on inputs A – D. All outputs are inactive (High) during any one of six illegal states, as shown in the truth table.

Inputs				Outputs
D	C	B	A	Selected (Low) Output*
0	0	0	0	Y0
0	0	0	1	Y1
0	0	1	0	Y2
0	0	1	1	Y3
0	1	0	0	Y4
0	1	0	1	Y5
0	1	1	0	Y6
0	1	1	1	Y7
1	0	0	0	Y8
1	0	0	1	Y9
1	0	1	0	All Outputs High
1	0	1	1	All Outputs High
1	1	0	0	All Outputs High
1	1	0	1	All Outputs High
1	1	1	0	All Outputs High
1	1	1	1	All Outputs High

* Selected output is Low (0) and all others are High.

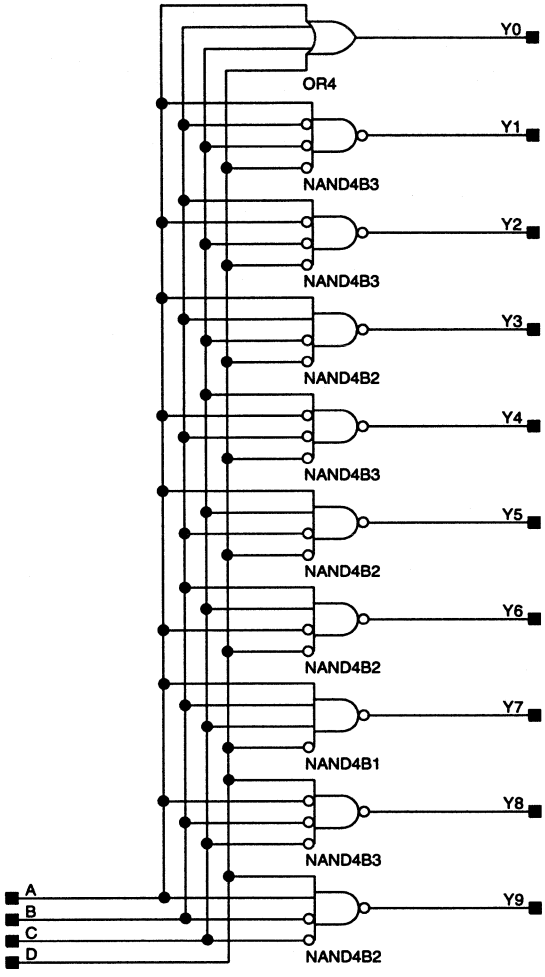
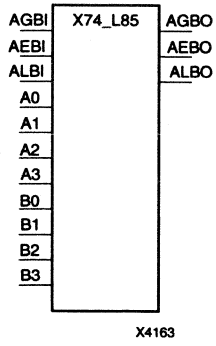


Figure 3-203 X74_42 XC2000/3000/4000 Implementation

X74_L85

4-Bit Expandable Magnitude Comparator



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive*

* not supported for XC7336 designs

X74_L85 is a 4-bit magnitude comparator that compares two 4-bit binary-weighted words A3 – A0 and B3 – B0, where A3 and B3 are the most significant bits. The greater-than output, AGBO, is High when A>B. The less-than output, ALBO, is High when A<B, and the equal output, AEBO, is High when A=B. The expansion inputs, AGBI, ALBI, and AEBI, are the least significant bits. Words of greater length can be compared by cascading the comparators. The AGBO, ALBO, and AEBO outputs of the stage handling less-significant bits are connected to the corresponding AGBI, ALBI, and AEBI inputs of the next stage handling more- significant bits. For proper operation, the stage handling the least significant bits must have AGBI and ALBI tied Low and AEBI tied High.

Inputs							Outputs		
A3, B3	**A2, B2**	**A1, B1**	**A0, B0**	**AGBI**	**ALBI**	**AEBI**	**AGBO**	**ALBO**	**AEBO**
A3>B3	X	X	X	X	X	X	1	0	0
A3<B3	X	X	X	X	X	X	0	1	0
A3=B3	A2>B2	X	X	X	X	X	1	0	0
A3=B3	A2<B2	X	X	X	X	X	0	1	0
A3=B3	A2=B2	A1>B1	X	X	X	X	1	0	0
A3=B3	A2=B2	A1<B1	X	X	X	X	0	1	0
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	1	0	0
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	0	1	0
A3=B3	A2=B2	A1=B1	A0=B0	1	0	0	1	0	0
A3=B3	A2=B2	A1=B1	A0=B0	0	1	0	0	1	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	1	0	0	1
A3=B3	A2=B2	A1=B1	A0=B0	0	1	1	0	1	1
A3=B3	A2=B2	A1=B1	A0=B0	1	0	1	1	0	1
A3=B3	A2=B2	A1=B1	A0=B0	1	1	1	1	1	1
A3=B3	A2=B2	A1=B1	A0=B0	1	1	0	1	1	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	0	0	0	0

For XC7000, outputs differ when A=B and when more than one expansion input (AGBI, ALBI, or AEBI) is high.

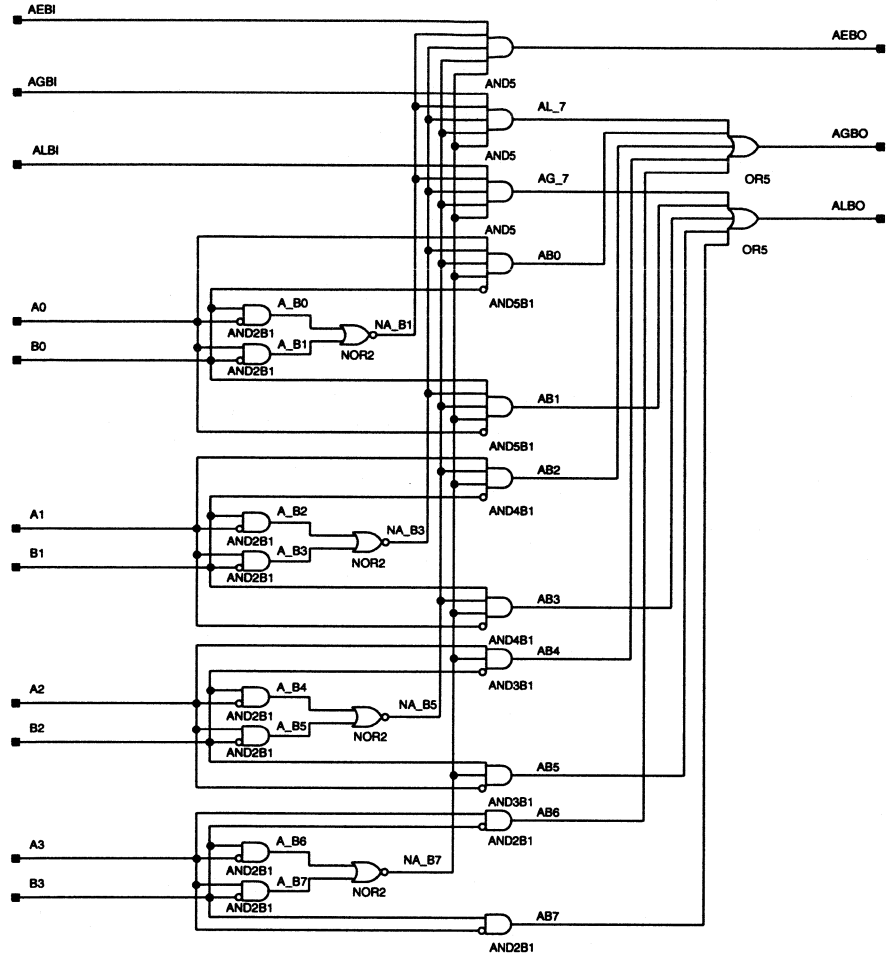
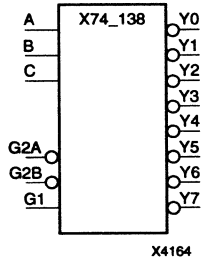


Figure 3-204 X74_L85 XC2000/3000/4000 Implementation

X74_138

3- to 8-Line Decoder/Demultiplexer with Active-Low Outputs and Three Enables



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_138 is an expandable decoder/demultiplexer with one active-High enable input (G1), two active-Low enable inputs (G2A and G2B), and eight active-Low outputs (Y7 – Y0). When G1 is High and G2A and G2B are Low, one of the eight active-Low outputs is selected with a 3-bit binary address on address inputs A, B, and C. The non-selected outputs are High. When G1 is Low or when G2A or G2B is High, all outputs are High.

X74_138 can be used as an 8-output active-Low demultiplexer by tying the data input to one of the enable inputs.

Inputs						Outputs							
C	B	A	G1	G2A	G2B	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	1	0	0	1	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	1	1	1	1	0	1
0	1	0	1	0	0	1	1	1	1	1	0	1	1
0	1	1	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	0	0	1	1	0	1	1	1	1	1
1	1	0	1	0	0	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	1	1	1	1	1	1	1
X	X	X	0	X	X	1	1	1	1	1	1	1	1
X	X	X	X	1	X	1	1	1	1	1	1	1	1
X	X	X	X	X	1	1	1	1	1	1	1	1	1

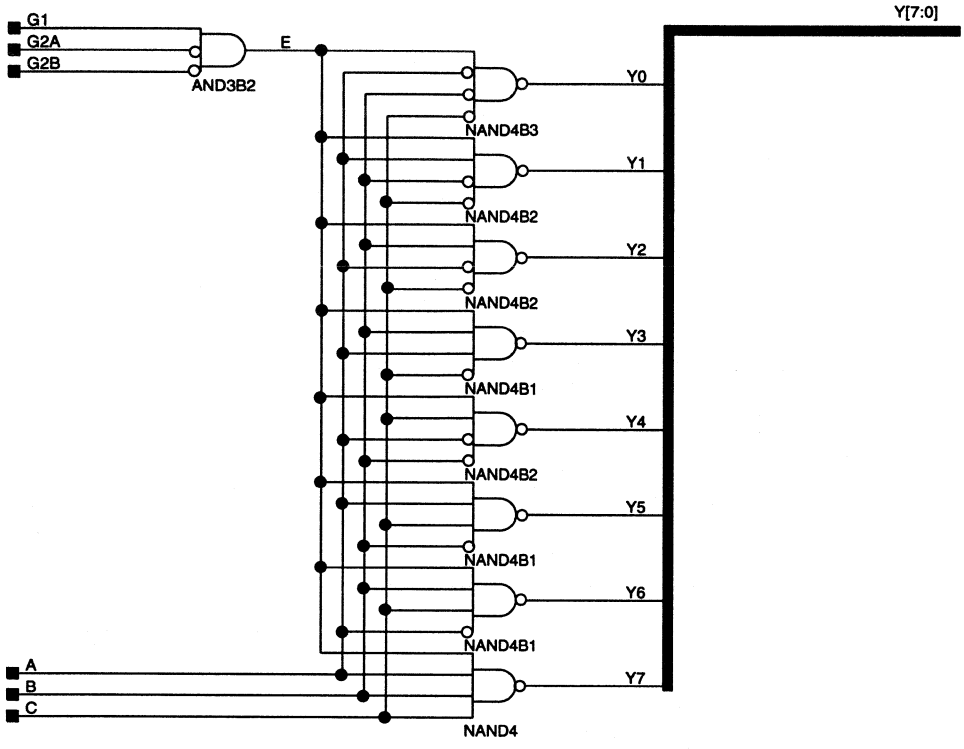
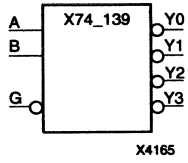


Figure 3-205 X74_138 XC2000/3000/4000 Implementation

X74_139

2- to 4-Line Decoder/Demultiplexer with Active-Low Outputs and Active-Low Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_139 implements one half of a standard 74139 dual 2- to 4-line decoder/demultiplexer. When the active-Low enable input (G) is Low, one of the four active-Low outputs (Y3 – Y0) is selected with the 2-bit binary address on the A and B address input lines. B is the High-order address bit. The non-selected outputs are High. Also, when G is High all outputs are High.

X74_139 can be used as a 4-output active-Low demultiplexer by tying the data input to G.

Inputs			Outputs			
G	B	A	Y3	Y2	Y1	Y0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	X	X	1	1	1	1

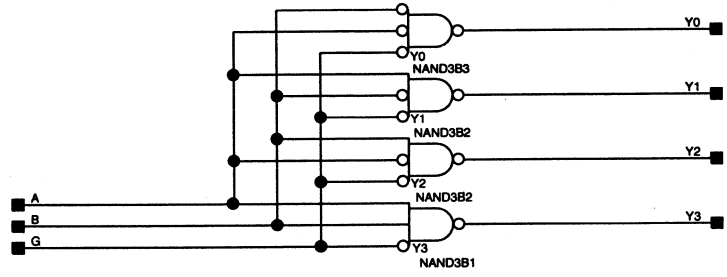
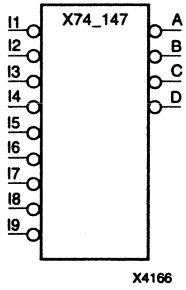


Figure 3-206 X74_139 XC2000/3000/4000 Implementation

X74_147

10- to 4-Line Priority Encoder with Active-Low Inputs and Outputs



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_147 is a 10-line-to-BCD-priority encoder that accepts data from nine active-Low inputs (I9 – I1) and produces a binary-coded decimal (BCD) representation on the four active-Low outputs A, B, C, and D. The data inputs are weighted, so when more than one input is active, only the one with the highest priority is encoded, with I9 having the highest priority. Only nine inputs are provided, because the implied "zero" condition requires no data input. "Zero" is encoded when all data inputs are High.

Inputs									Outputs			
I9	I8	I7	I6	I5	I4	I3	I2	I1	D	C	B	A
1	1	1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	X	1	1	0	1
1	1	1	1	1	1	0	X	X	1	1	0	0
1	1	1	1	1	0	X	X	X	1	0	1	1
1	1	1	1	0	X	X	X	X	1	0	1	0
1	1	1	0	X	X	X	X	X	1	0	0	1
1	1	0	X	X	X	X	X	X	1	0	0	0
1	0	X	X	X	X	X	X	X	0	1	1	1
0	X	X	X	X	X	X	X	X	0	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1

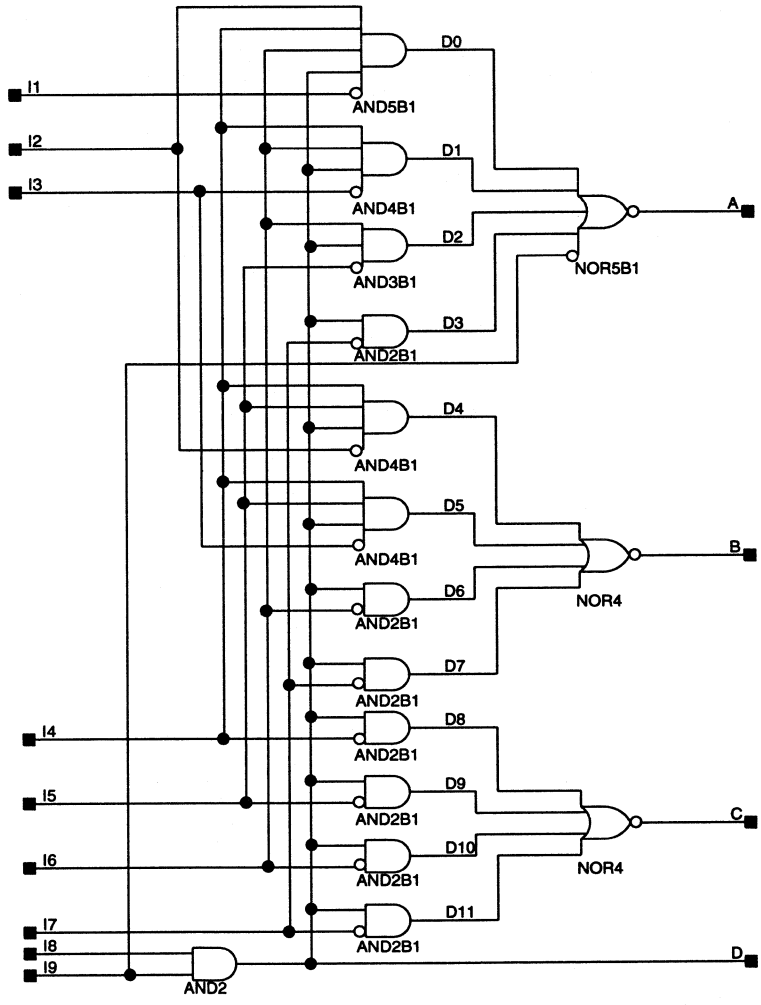
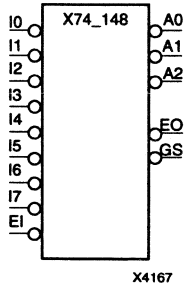


Figure 3-207 X74_147 XC2000/3000/4000 Implementation

X74_148

8- to 3-Line Cascadable Priority Encoder with Active-Low Inputs and Outputs



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_148 8-input priority encoder accepts data from eight active-Low inputs (I7 – I0) and produces a binary representation on the three active-Low outputs (A2 – A0). The data inputs are weighted, so when more than one of the inputs is active, only the input with the highest priority is encoded, I7 having the highest priority. The active-Low group signal (GS) is Low whenever one of the data inputs is Low and the active-Low enable input (EI) is Low.

The active-Low enable input (EI) and active-Low enable output (EO) are used to cascade devices and retain priority control. The EO of the highest priority stage is connected to the EI of the next-highest priority stage. When EI is High, the data outputs and EO are High. When EI is Low, the encoder output represents the highest-priority Low data input, and the EO is High. When EI is Low and all the data inputs are High, the EO output is Low to enable the next-lower priority stage.

Inputs									Outputs				
EI	I7	I6	I5	I4	I3	I2	I1	I0	A2	A1	A0	GS	EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	0	1	1	1	0	1
0	1	1	1	1	1	1	0	X	1	1	0	0	1
0	1	1	1	1	1	0	X	X	1	0	1	0	1
0	1	1	1	1	0	X	X	X	1	0	0	0	1
0	1	1	1	0	X	X	X	X	0	1	1	0	1
0	1	1	0	X	X	X	X	X	0	1	0	0	1
0	1	0	X	X	X	X	X	X	0	0	1	0	1
0	0	X	X	X	X	X	X	X	0	0	0	0	1

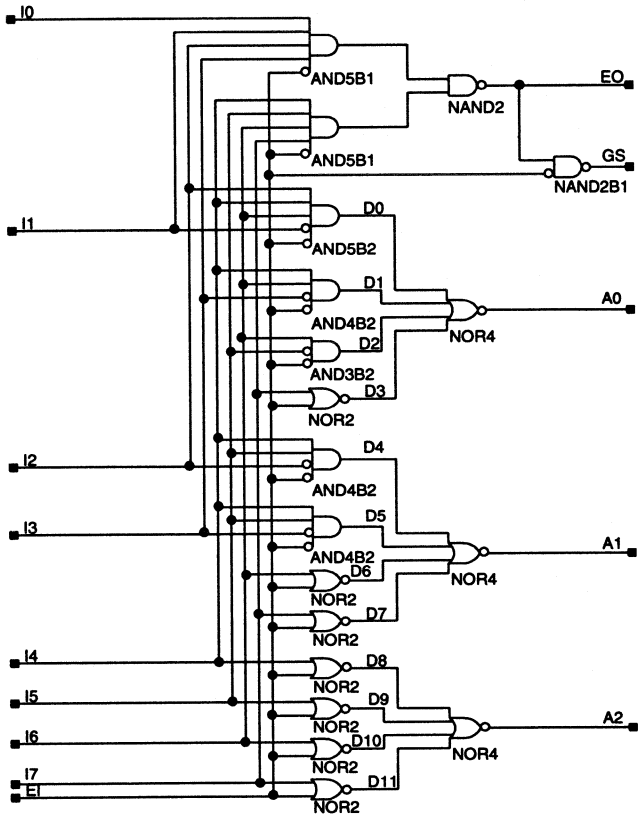
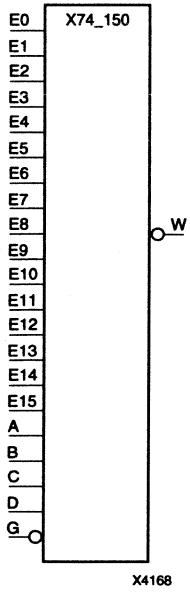


Figure 3-208 X74_148 XC2000/3000/4000 Implementation

X74_150

16-to-1 Multiplexer with Active-Low Enable and Output



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the active-Low enable input (G) is Low, the X74_150 multiplexer chooses one data bit from 16 sources (E15 – E0) under the control of select inputs A, B, C, and D. The active-Low output (W) reflects the inverse of the selected input, as shown in the truth table. When the enable input (G) is High, the output (W) is High.

Inputs					Outputs
G	D	C	B	A	Selected Input Appears (Inverted) on W
1	X	X	X	X	1
0	0	0	0	0	$\overline{E0}$
0	0	0	0	1	$\overline{E1}$
0	0	0	1	0	$\overline{E2}$
0	0	0	1	1	$\overline{E3}$
0	0	1	0	0	$\overline{E4}$
0	0	1	0	1	$\overline{E5}$
0	0	1	1	0	$\overline{E6}$
0	0	1	1	1	$\overline{E7}$
0	1	0	0	0	$\overline{E8}$
0	1	0	0	1	$\overline{E9}$
0	1	0	1	0	$\overline{E10}$
0	1	0	1	1	$\overline{E11}$
0	1	1	0	0	$\overline{E12}$
0	1	1	0	1	$\overline{E13}$
0	1	1	1	0	$\overline{E14}$
0	1	1	1	1	$\overline{E15}$

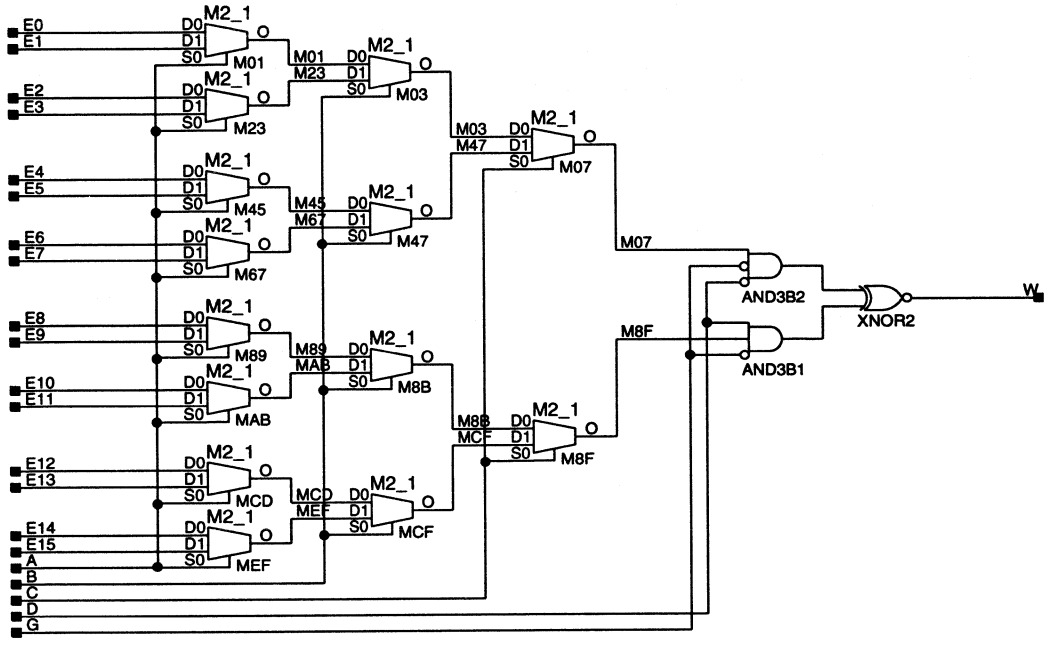
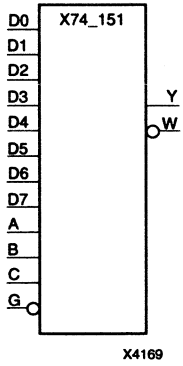


Figure 3-209 X74_150 XC2000/3000/4000 Implementation

X74_151

8-to-1 Multiplexer with Active-Low Enable and Complementary Outputs



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the active-Low enable (G) is Low, the X74_151 multiplexer chooses one data bit from eight sources (D7 – D0) under control of the select inputs A, B, and C. The output (Y) reflects the state of the selected input, and the active-Low output (W) reflects the inverse of the selected input as shown in the truth table. When G is High, the Y output is Low, and the W output is High.

Inputs				Outputs	
G	C	B	A	Y	W
1	X	X	X	1	0
0	0	0	0	D0	$\overline{D0}$
0	0	0	1	D1	$\overline{D1}$
0	0	1	0	D2	$\overline{D2}$
0	0	1	1	D3	$\overline{D3}$
0	1	0	0	D4	$\overline{D4}$
0	1	0	1	D5	$\overline{D5}$
0	1	1	0	D6	$\overline{D6}$
0	1	1	1	D7	$\overline{D7}$

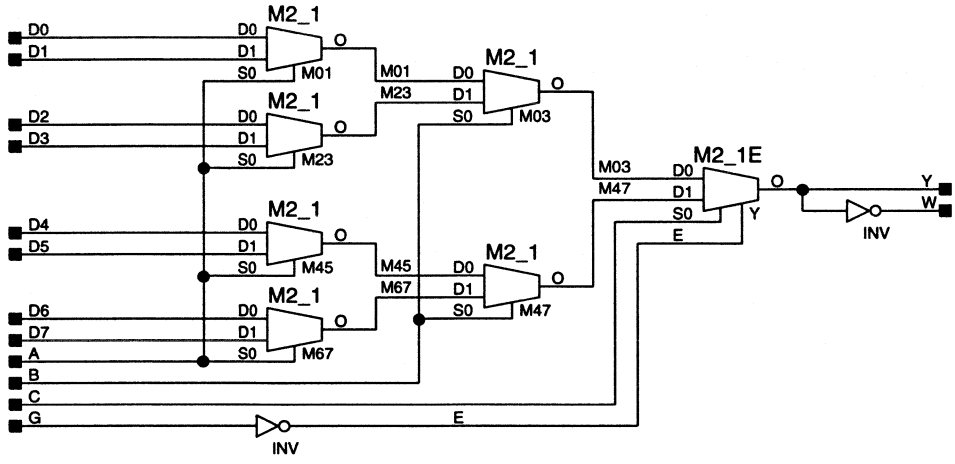
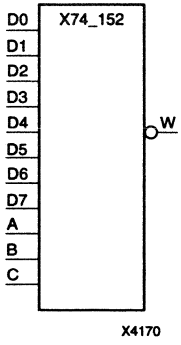


Figure 3-210 X74_151 XC2000/3000/4000 Implementation

X74_152

8-to-1 Multiplexer with Active-Low Output



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_152 multiplexer chooses one data bit from eight sources (D7 – D0) under control of the select inputs A, B, and C. The active-Low output (W) reflects the inverse of the selected data input, as shown in the truth table.

Inputs			Outputs
C	B	A	W
0	0	0	$\overline{D0}$
0	0	1	$\overline{D1}$
0	1	0	$\overline{D2}$
0	1	1	$\overline{D3}$
1	0	0	$\overline{D4}$
1	0	1	$\overline{D5}$
1	1	0	$\overline{D6}$
1	1	1	$\overline{D7}$

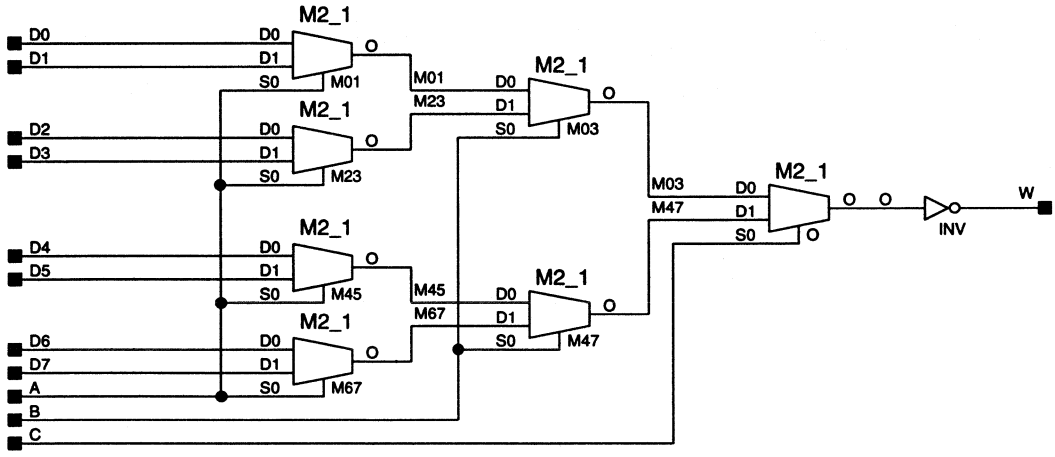
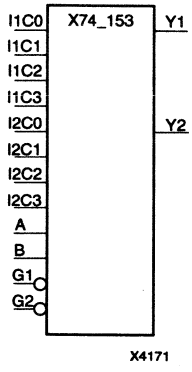


Figure 3-211 X74_152 XC2000/3000/4000 Implementation

X74_153

Dual 4-to-1 Multiplexer with Active-Low Enables and Common Select Input



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the active-Low enable inputs G1 and G2 are Low, the data output Y1, reflects the data input chosen by select inputs A and B from data inputs I1C3 – I1C0. The data output Y2 reflects the data input chosen by select inputs A and B from data inputs I2C3 – I2C0. When G1 or G2 is High, the corresponding output, Y1 or Y2 respectively, is Low.

Inputs			Outputs
G	B	A	Y
1	X	X	0
0	0	0	IC0
0	0	1	IC1
0	1	0	IC2
0	1	1	IC3

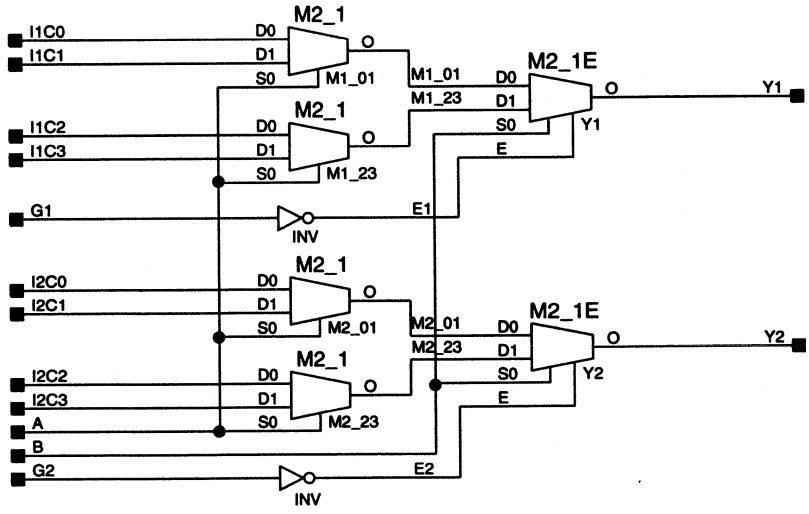
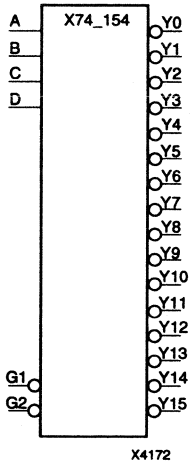


Figure 3-212 X74_153 XC2000/3000/4000 Implementation

X74_154

4- to 16-Line Decoder/Demultiplexer with Two Enables and Active-Low Outputs



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the active-Low enable inputs G1 and G2 of the X74_154 decoder/demultiplexer are Low, one of 16 active-Low outputs, Y15 – Y0, is selected under the control of four binary address inputs A, B, C, and D. The non-selected inputs are High. Also, when either input G1 or G2 is High, all outputs are High.

The X74_154 can be used as a 16-to-1 demultiplexer by tying the data input to one of the G inputs and tying the other G input Low.

Inputs						Outputs							
G1	G2	D	C	B	A	Y15	Y14	Y13	Y12	Y11	Y10	Y9	... Y0
1	X	X	X	X	X	1	1	1	1	1	1	1	... 1
X	1	X	X	X	X	1	1	1	1	1	1	1	... 1
0	0	1	1	1	1	0	1	1	1	1	1	1	... 1
0	0	1	1	1	0	1	0	1	1	1	1	1	... 1
0	0	1	1	0	1	1	1	0	1	1	1	1	... 1
-	-	-	-	-	-	-	-	-	-	-	-	-	... -
-	-	-	-	-	-	-	-	-	-	-	-	-	... -
-	-	-	-	-	-	-	-	-	-	-	-	-	... -
0	0	0	0	0	0	1	1	1	1	1	1	1	... 0

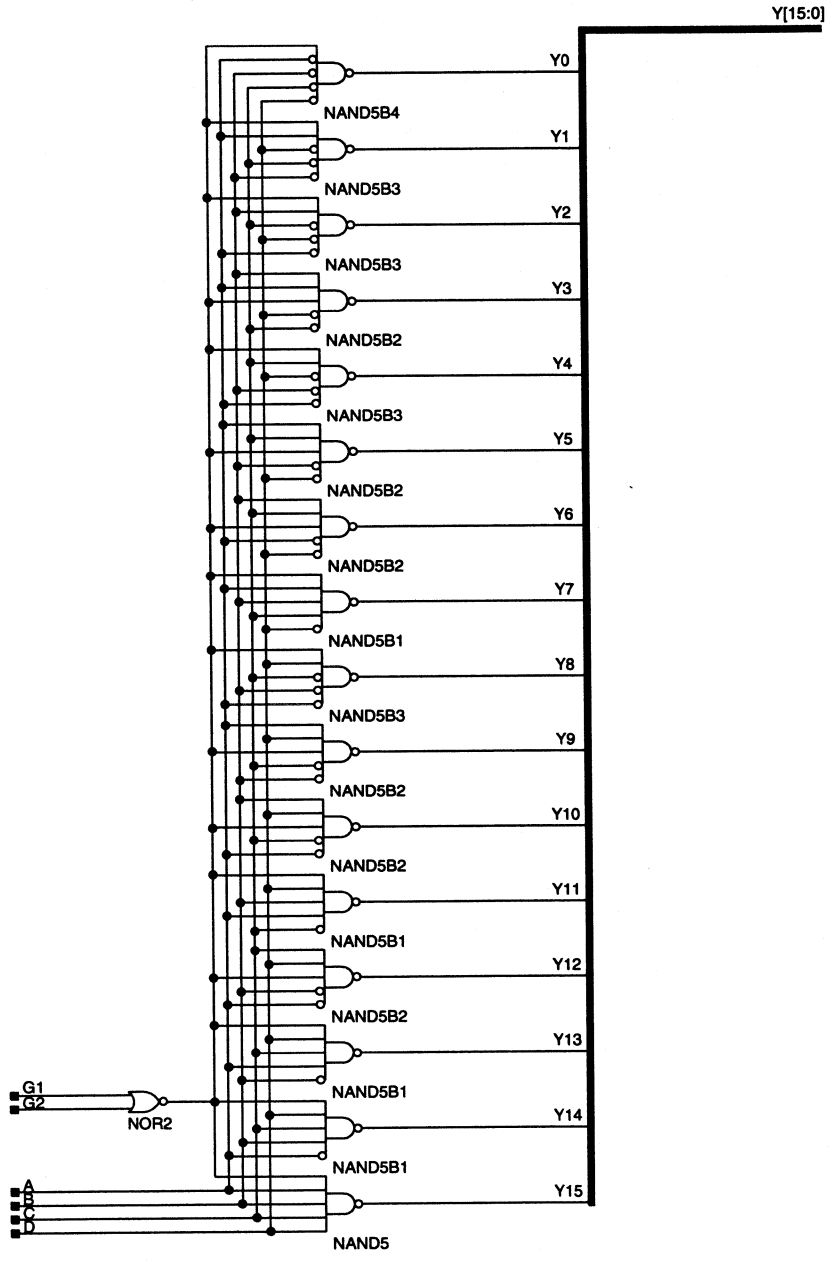
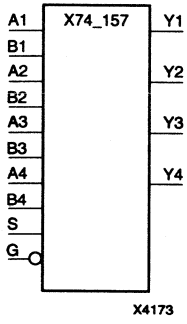


Figure 3-213 X74_154 XC2000/3000/4000 Implementation

X74_157

Quadruple 2-to-1 Multiplexer with Common Select and Active-Low Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the active-Low enable input (G) is Low, a 4-bit word is selected from one of two sources (A3 – A0 or B3 – B0) under the control of the select input (S) and is reflected on the four outputs (Y4 – Y1). When S is Low, the outputs reflect A3 – A0; when S is High, the outputs reflect B3 – B0. When G is High, the outputs are Low.

Inputs				Outputs
G	S	B	A	Y
1	X	X	X	0
0	1	1	X	1
0	1	0	X	0
0	0	X	1	1
0	0	X	0	0

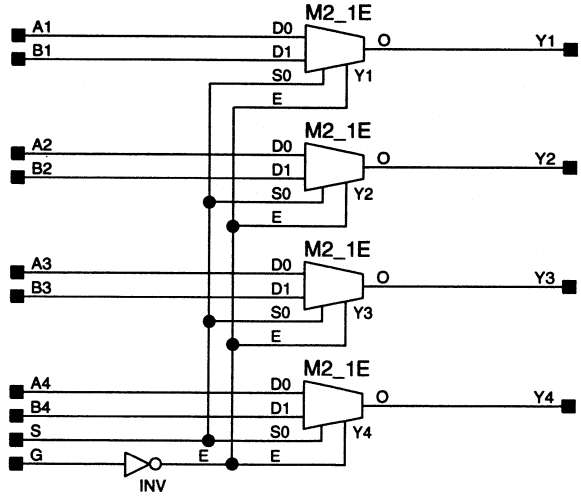
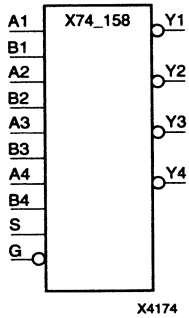


Figure 3-214 X74_157 XC2000/3000/4000 Implementation

X74_158

Quadruple 2-to-1 Multiplexer with Common Select, Active-Low Enable, and Active-Low Outputs



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the active-Low enable (G) is Low, a 4-bit word is selected from one of two sources (A3 – A0 or B3 – B0) under the control of the common select input (S). The inverse of the selected word is reflected on the active-Low outputs (Y4 – Y1). When S is Low, $\overline{A3}$ – $\overline{A0}$ appear on the outputs; when S is High, $\overline{B3}$ – $\overline{B0}$ appear on the outputs. When G is High, the outputs are High.

Inputs				Outputs
G	S	B	A	Y
1	X	X	X	1
0	1	1	X	0
0	1	0	X	1
0	0	X	1	0
0	0	X	0	1

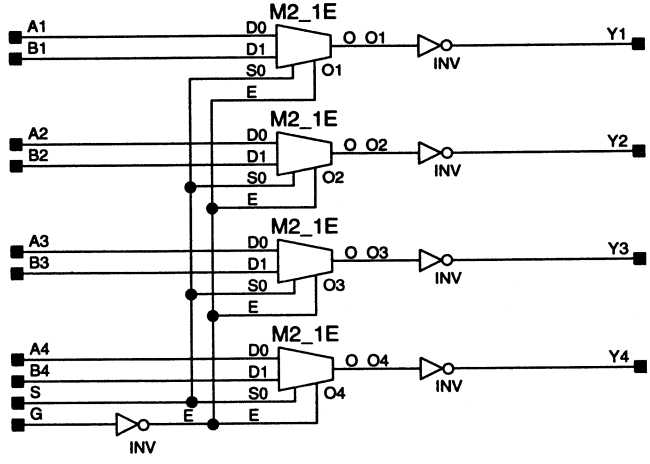
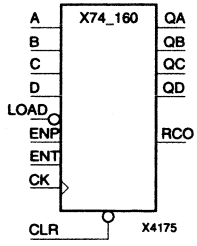


Figure 3-215 X74_158 XC2000/3000/4000 Implementation

X74_160

4-Bit BCD Counter with Parallel and Trickle Enables, Active-Low Load Enable, and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_160 is a 4-stage, 4-bit, synchronous, loadable, resettable, cascadable, binary-coded decimal (BCD) counter. The active-Low asynchronous clear (CLR), when Low, overrides all other inputs and resets the data (QD, QC, QB, QA) and ripple carry-out (RCO) outputs Low during the Low-to-High clock (C) transition. When the active-Low load enable input (LOAD) is Low, parallel clock enable (ENP), and trickle clock enable (ENT) are overridden and data on inputs A, B, C, and D are loaded into the counter during the Low-to-High clock transition. The data outputs (QD, QC, QB, QA) increment when ENP, ENT LOAD, and CLR are High during the Low-to-High clock transition. The counter ignores clock transitions when ENP or ENT are Low and LOAD is High. RCO is High when QD, QA, and ENT are High and QC and QB are Low.

Inputs						Outputs	
CLR	LOAD	ENP	ENT	D – A	CK	QD – QA	RCO
0	X	X	X	X	X	0	0
1	0	X	X	D – A	↑	d – a	RCO
1	1	0	X	X	X	No Chg	RCO
1	1	X	0	X	X	No Chg	0
1	1	1	1	X	↑	Inc	RCO

$RCO = (QD \bullet \overline{QC} \bullet \overline{QB} \bullet QA \bullet ENT)$

d – a = state of referenced input one set-up time prior to active clock transition

The carry-lookahead design allows cascading of large counters without extra gating. Both ENT and ENP must be High to count. ENT is fed forward to enable RCO, which produces a High output pulse with the approximate duration of the QA output. The following figure illustrates a carry-lookahead design.

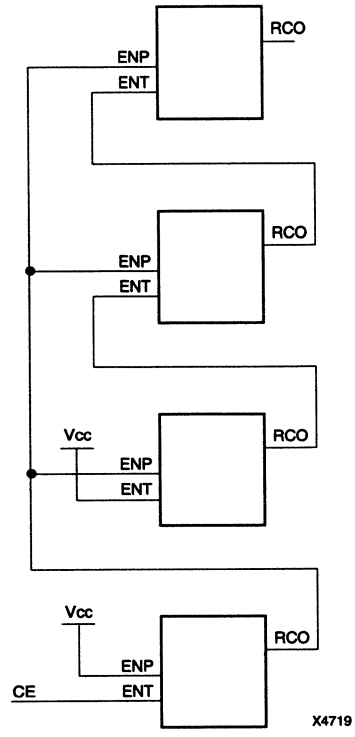


Figure 3-216 Carry-Lookahead Design

The RCO output of the first stage of the ripple carry is connected to the ENP input of the second stage and all subsequent stages. The RCO output of the second stage and all subsequent stages is connected to the ENT input of the next stage. The ENT of the second stage is always enabled/tied to VCC. CE is always connected to the ENT input of the first stage. This cascading method allows the first stage of the ripple carry to be built as a prescaler. In other words, the first stage is built to count very fast.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles.

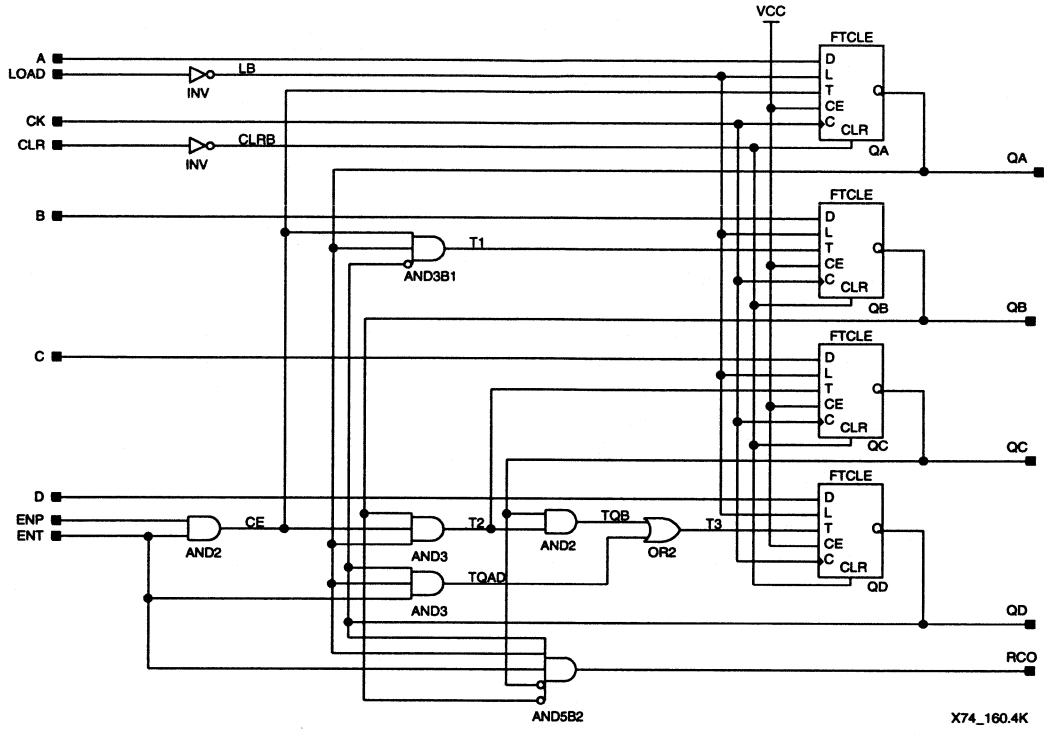
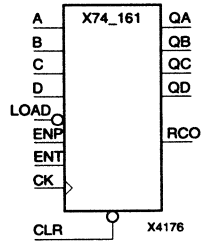


Figure 3-217 X74_160 XC2000/3000/4000 Implementation

X74_161

4-Bit Counter with Parallel and Trickle Enables Active-Low Load Enable and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_161 is a 4-stage, 4-bit, synchronous, loadable, resettable, cascadable binary counter. The active-Low asynchronous clear (CLR), when Low, overrides all other inputs and resets the data outputs (QD, QC, QB, QA) and the ripple carry-out output (RCO) Low. When the active-Low load enable (LOAD) is Low and CLR is High, parallel clock enable (ENP) and trickle clock enable (ENT) are overridden and the data on inputs A, B, C, and D is loaded into the counter during the Low-to-High clock (C) transition. The data outputs (QD, QC, QB, QA) increment when LOAD, ENP, ENT, and CLR are High during the Low-to-High clock transition. The counter ignores clock transitions when LOAD is High and ENP or ENT are Low. RCO is High when QD – QA and ENT are High.

Inputs						Outputs	
CLR	LOAD	ENP	ENT	D – A	CK	QD – QA	RCO
0	X	X	X	X	X	0	0
1	0	X	X	D – A	↑	d – a	RCO
1	1	0	X	X	X	No Chg	RCO
1	1	X	0	X	X	No Chg	0
1	1	1	1	X	↑	Inc	RCO

RCO = (QD•QC•QB•QA•ENT)

d – a = state of referenced input one set-up time prior to active clock transition

The carry-lookahead design accommodates large counters without extra gating. Both the ENT and ENP inputs must be High to count. ENT is fed forward to enable RCO, which produces a High output with the approximate duration of the QA output. The following figure illustrates a carry-lookahead design.

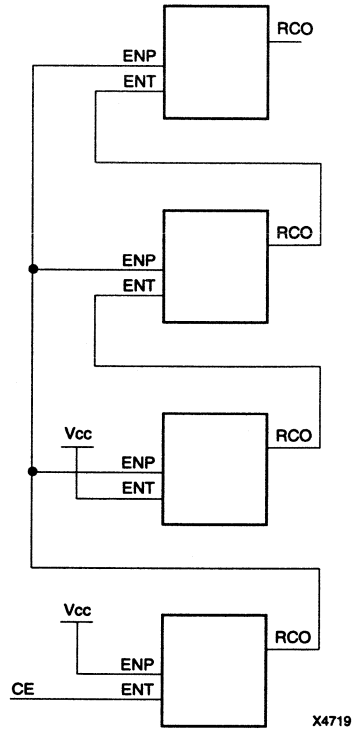


Figure 3-218 Carry-Lookahead Design

The RCO output of the first stage of the ripple carry is connected to the ENP input of the second stage and all subsequent stages. The RCO output of the second stage and all subsequent stages is connected to the ENT input of the next stage. The ENT of the second stage is always enabled/tied to VCC. CE is always connected to the ENT input of the first stage. This cascading method allows the first stage of the ripple carry to be built as a prescaler. In other words, the first stage is built to count very fast.

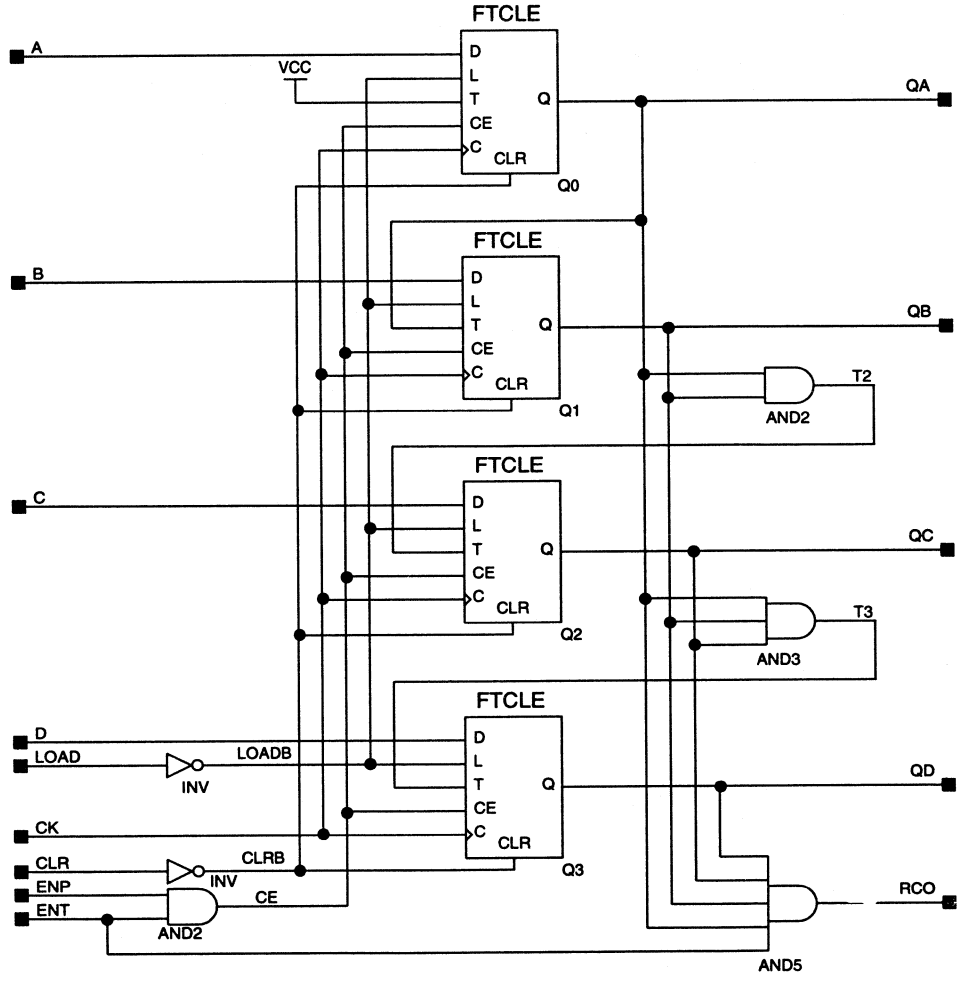
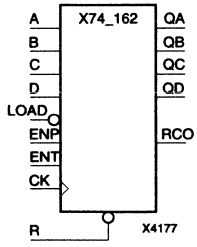


Figure 3-219 X74_161 XC2000/3000/4000 Implementation

X74_162

4-Bit Counter with Parallel and Trickle Enables and Active-Low Load Enable and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_162 is a 4-stage, 4-bit, synchronous, loadable, resettable, cascadable binary-coded decimal (BCD) counter. The active-Low synchronous reset (R), when Low, overrides all other inputs and resets the data (QD, QC, QB, QA) and ripple carry-out (RCO) outputs Low during the Low-to-High clock (C) transition. When the active-Low load enable input (LOAD) is Low, parallel clock enable (ENP) and trickle clock enable (ENT) are overridden and data on inputs A, B, C, and D is loaded into the counter during the Low-to-High clock transition. The data outputs (QD, QC, QB, QA) increment when ENP, ENT, LOAD, and R are High during the Low-to-High clock transition. The counter ignores clock transitions when ENP or ENT are Low and LOAD is High. RCO is High when QD, QA, and ENT are High and QC and QB are Low.

Inputs						Outputs	
R	LOAD	ENP	ENT	D – A	CK	QD – QA	RCO
0	X	X	X	X	↑	0	0
1	0	X	X	D – A	↑	d – a	RCO
1	1	0	X	X	X	No Chg	RCO
1	1	X	0	X	X	No Chg	0
1	1	1	1	X	↑	Inc	RCO

$RCO = (QD \bullet \overline{QC} \bullet \overline{QB} \bullet QA \bullet ENT)$

d – a = state of referenced input one set-up time prior to active clock transition

The carry-lookahead design accommodates cascading large counters without extra gating. Both ENT and ENP must be High to count. The ENT is fed forward to enable RCO, which produces a High output pulse with the approximate duration of the QA output. The following figure illustrates a carry-lookahead design.

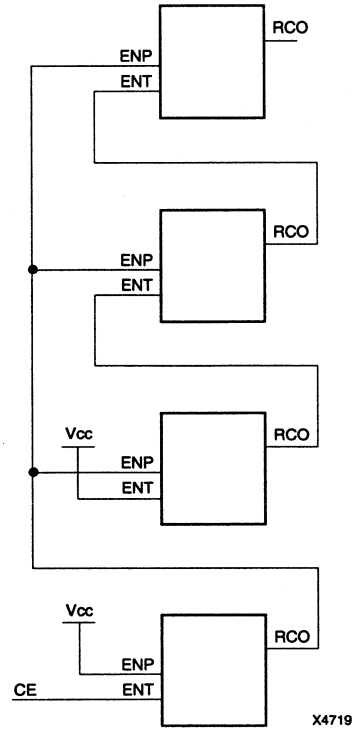


Figure 3-220 Carry-Lookahead Design

The RCO output of the first stage of the ripple carry is connected to the ENP input of the second stage and all subsequent stages. The RCO output of the second stage and all subsequent stages is connected to the ENT input of the next stage. The ENT of the second stage is always enabled/tied to VCC. CE is always connected to the ENT input of the first stage. This cascading method allows the first stage of the ripple carry to be built as a prescaler. In other words, the first stage is built to count very fast.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles.

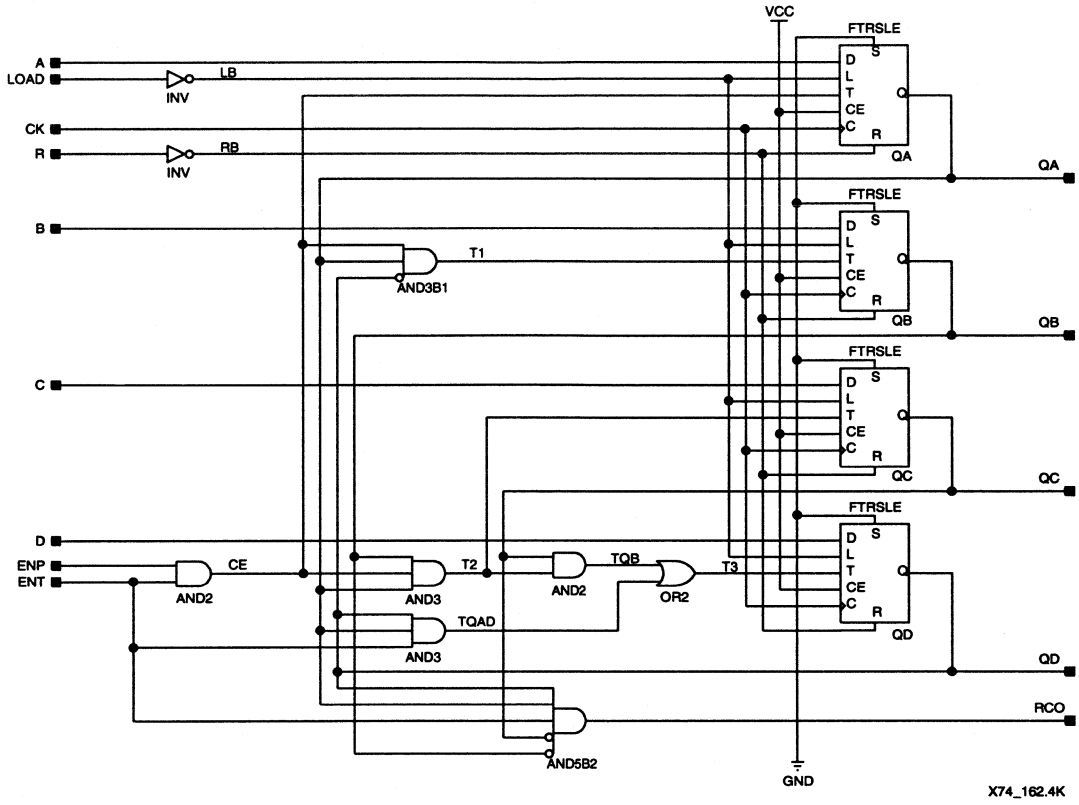
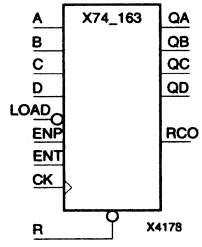


Figure 3-221 X74_162 XC2000/3000/4000 Implementation

X74_163

4-Bit Counter with Parallel and Trickle Enables, Active-Low Load Enable, and Synchronous Reset



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_163 is a 4-stage, 4-bit, synchronous, loadable, resettable, cascadable binary counter. The active-Low synchronous reset (R), when Low, overrides all other inputs and resets the data outputs (QD, QC, QB, QA) and the ripple carry-out output (RCO) Low. When the active-Low load enable (LOAD) is Low and R is High, parallel clock enable (ENP) and trickle clock enable (ENT) are overridden and the data on inputs (A, B, C, D) is loaded into the counter during the Low-to-High clock (C) transition. The outputs (QD, QC, QB, QA) increment when LOAD, ENP, ENT, and R are High during the Low-to-High clock transition. The counter ignores clock transitions when LOAD is High and ENP or ENT are Low; RCO is High when QD – QA and ENT are High.

Inputs						Outputs	
R	LOAD	ENP	ENT	D – A	CK	QD – QA	RCO
0	X	X	X	X	↑	0	0
1	0	X	X	D – A	↑	d – a	RCO
1	1	0	X	X	X	No Chg	RCO
1	1	X	0	X	X	No Chg	0
1	1	1	1	X	↑	Inc	RCO

RCO = (QD•QC•QB•QA•ENT)

d – a = state of referenced input one set-up time prior to active clock transition

The carry-lookahead design accommodates large counters without extra gating. Both the ENT and ENP inputs must be High to count. ENT is propagated forward to enable RCO, which produces a High output with the approximate duration of the QA output. The following figure illustrates a carry-lookahead design.

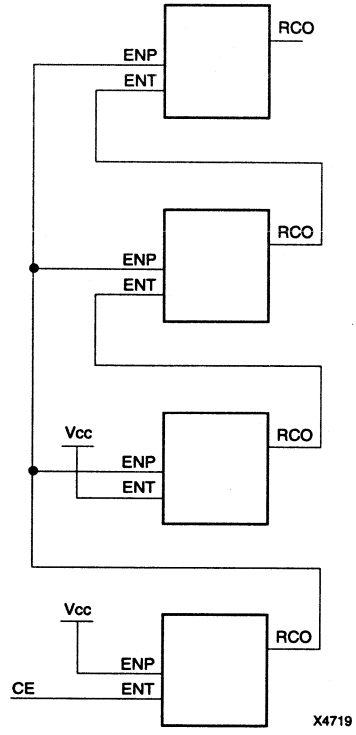


Figure 3-222 Carry-Lookahead Design

The RCO output of the first stage of the ripple carry is connected to the ENP input of the second stage and all subsequent stages. The RCO output of the second stage and all subsequent stages is connected to the ENT input of the next stage. The ENT of the second stage is always enabled/tied to VCC. CE is always connected to the ENT input of the first stage. This cascading method allows the first stage of the ripple carry to be built as a prescaler. In other words, the first stage is built to count very fast.

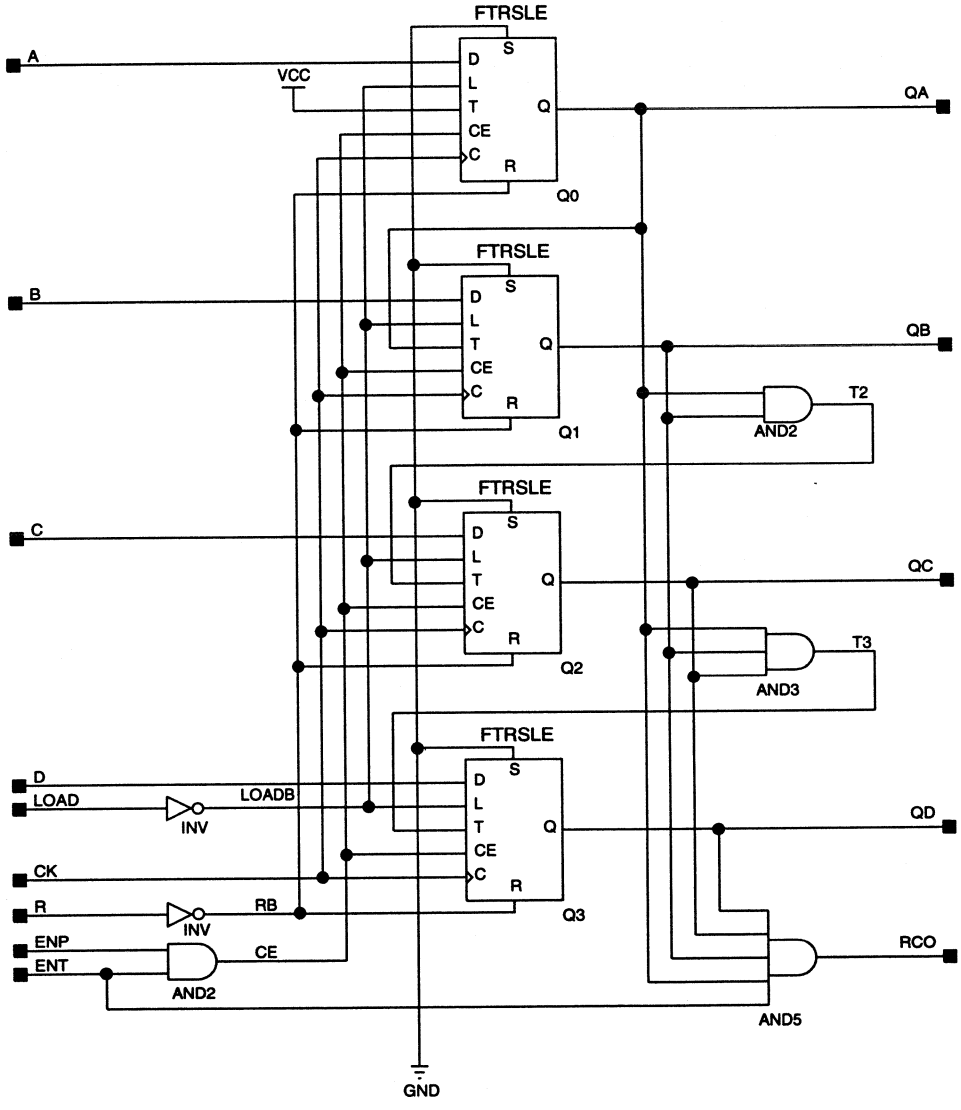
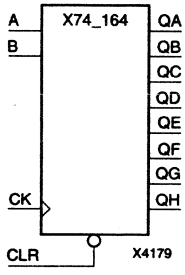


Figure 3-223 X74_163 XC2000/3000/4000 Implementation

X74_164

8-Bit Serial-In Parallel-Out Shift Register with Active-Low Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_164 is an 8-bit, serial input (A and B), parallel output (QH – QA) shift register with an active-Low asynchronous clear (CLR) input. The asynchronous CLR, when Low, overrides the clock input and sets the data outputs (QH – QA) Low. When CLR is High, the AND function of the two data inputs (A and B) is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the QA output. During subsequent Low-to-High clock transitions, with CLR High, the data is shifted to the next-highest bit position as new data is loaded into QA (A and B→QA, QA→QB, QB→QC, and so forth).

Registers can be cascaded by connecting the QH output of one stage to the A input of the next stage, by tying B High, and by connecting the clock and CLR inputs in parallel.

Inputs				Outputs	
CLR	A	B	CK	QA	QB – QH
0	X	X	X	0	0
1	1	1	↑	1	qA – qG
1	0	X	↑	0	qA – qG
1	X	0	↑	0	qA – qG

qA – qG = state of referenced output one set-up time prior to active clock transition

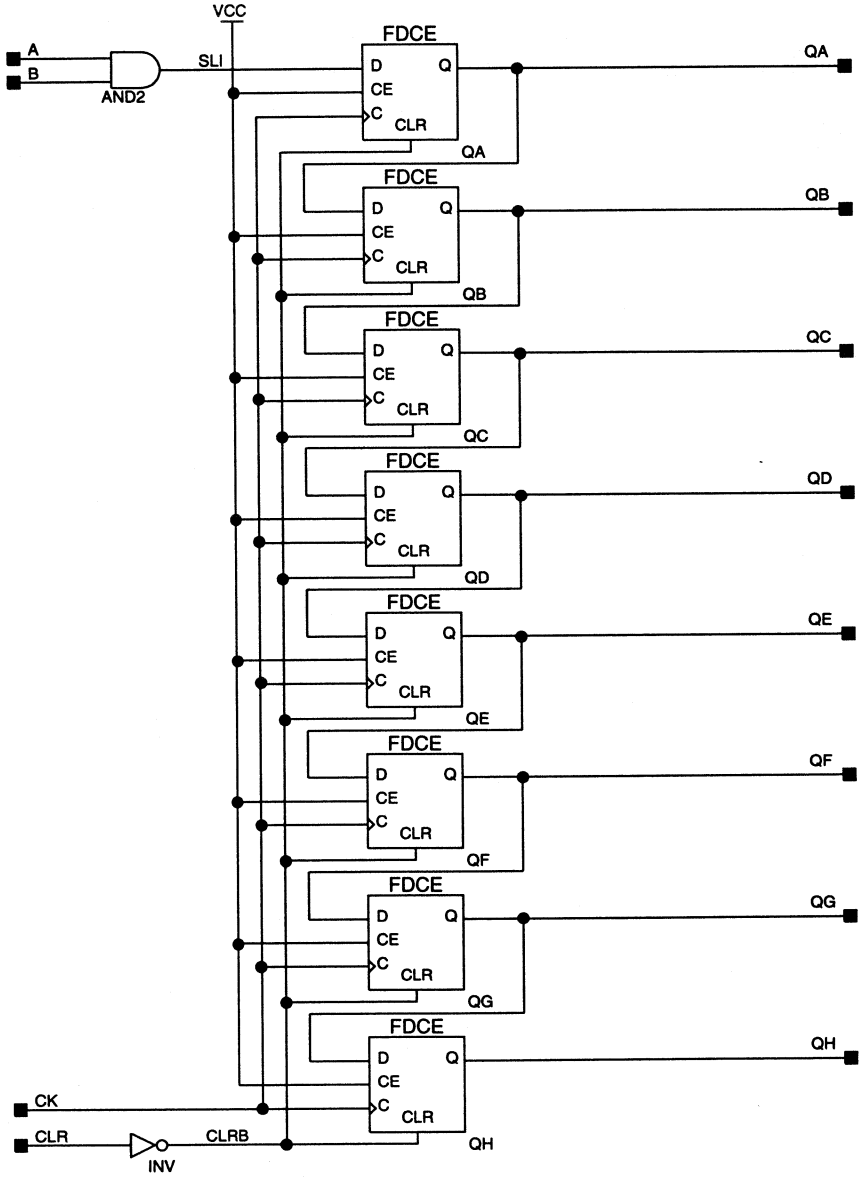
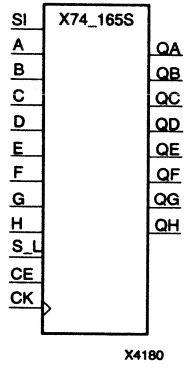


Figure 3-224 X74_164 XC2000/3000/4000 Implementation

X74_165S

8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_165S is an 8-bit shift register with serial-input (SI), parallel-inputs (H – A), parallel-outputs (QH – QA), and two control inputs – clock enable (CE) and active-Low shift/load enable (S_L). When S_L is Low, data on the H – A inputs is loaded into the corresponding QH – QA bits of the register on the Low-to-High clock (C) transition. When CE and S_L are High, data on the SI input is loaded into the first bit of the register during the Low-to-High clock transition. During subsequent Low-to-High clock transitions, with CE and S_L High, the data is shifted to the next-highest bit position (shift right) as new data is loaded into QA (SI→QA, QA→QB, QB→QC, and so forth). The register ignores clock transitions when CE is Low and S_L is High.

Registers can be cascaded by connecting the QH output of one stage to the SI input of the next stage and connecting clock, CE, and S_L inputs in parallel.

Inputs					Outputs	
S_L	CE	SI	A – H	CK	QA	QB – QH
0	X	X	A – H	↑	qa	qb – qh
1	0	X	X	X	--No Change---	
1	1	SI	X	↑	si	qA – qG

si, qn represent state of referenced input or output one set-up time prior to active clock transition.

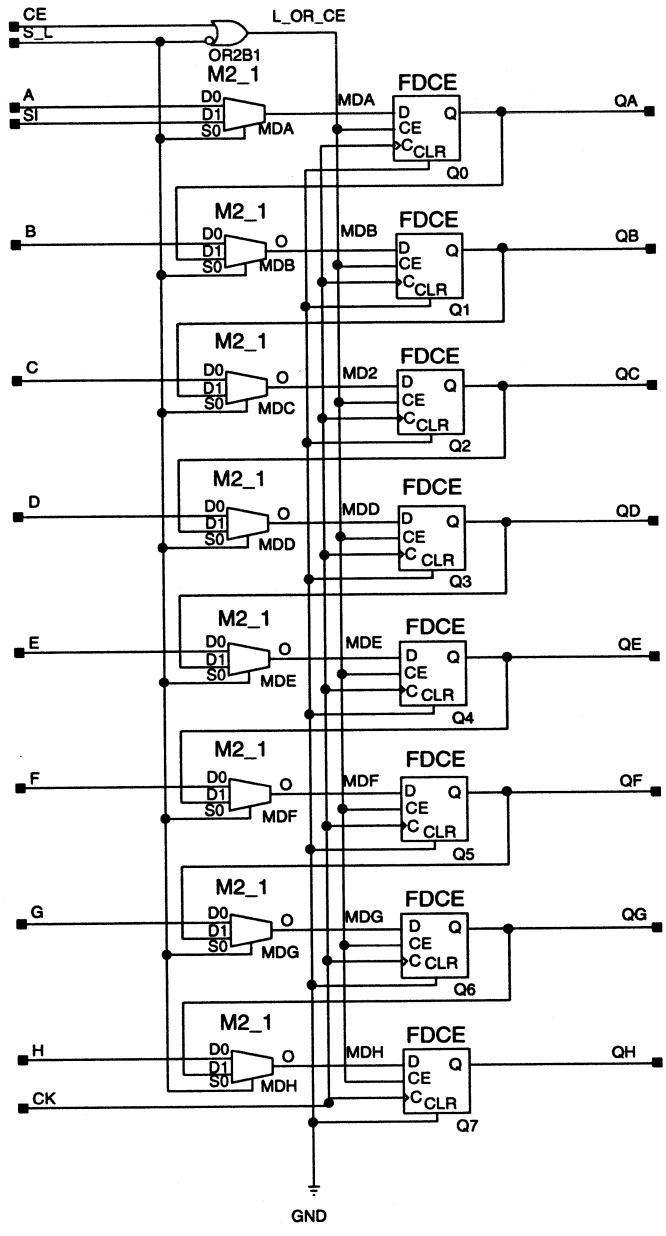
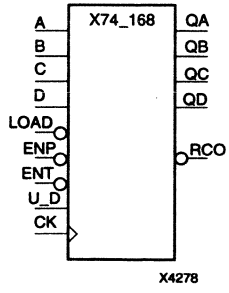


Figure 3-225 X74_165S XC2000/3000/4000 Implementation

X74_168

4-Bit BCD Bidirectional Counter with Parallel and Trickle Clock Enables and Active-Low Load Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_168 is a 4-stage, 4-bit, synchronous, loadable, cascadable, bidirectional binary-coded-decimal (BCD) counter. The data on the D – A inputs is loaded into the counter when the active-Low load enable (LOAD) is Low during the Low-to-High clock (C) transition. The LOAD input, when Low, has priority over parallel clock enable (ENP), trickle clock enable (ENT), and the bidirectional (U_D) control. The outputs (QD – QA) increment when U_D and LOAD are High and ENP and ENT are Low during the Low-to-High clock transition. The outputs decrement when LOAD is High and ENP, ENT, and U_D are Low during the Low-to-High clock transition. The counter ignores clock transitions when LOAD and either ENP or ENT are High.

Inputs						Outputs	
LOAD	ENP	ENT	U_D	A – D	CK	QA – QD	RCO
0	X	X	X	A – D	↑	qa – qd	RCO
1	0	0	1	X	↑	Inc	RCO
1	0	0	0	X	↑	Dec	RCO
1	1	0	X	X	X	No Chg	RCO
1	X	1	X	X	X	No Chg	1

$RCO = (Q3 \bullet \overline{Q2} \bullet \overline{Q1} \bullet Q0 \bullet U\_D \bullet \overline{ENT}) + (\overline{Q3} \bullet \overline{Q2} \bullet \overline{Q1} \bullet \overline{Q0} \bullet \overline{U\_D} \bullet \overline{ENT})$

qa – qd = state of referenced input one set-up time prior to active clock transition

The active-Low ripple carry-out output (RCO) is Low when QD, QA, and U_D are High and QC, QB, and ENT are Low. RCO is also Low when all outputs, ENT and U_D are Low. The following figure illustrates a carry-lookahead design.

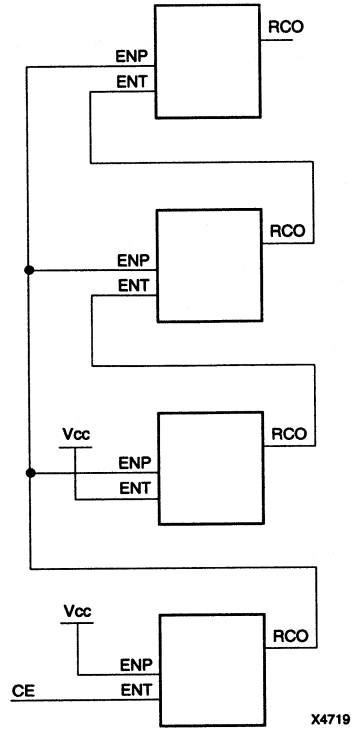


Figure 3-226 Carry-Lookahead Design

The RCO output of the first stage of the ripple carry is connected to the ENP input of the second stage and all subsequent stages. The RCO output of second stage and all subsequent stages is connected to the ENT input of the next stage. The ENT of the second stage is always enabled/tied to VCC. CE is always connected to the ENT input of the first stage. This cascading method allows the first stage of the ripple carry to be built as a prescaler. In other words, the first stage is built to count very fast.

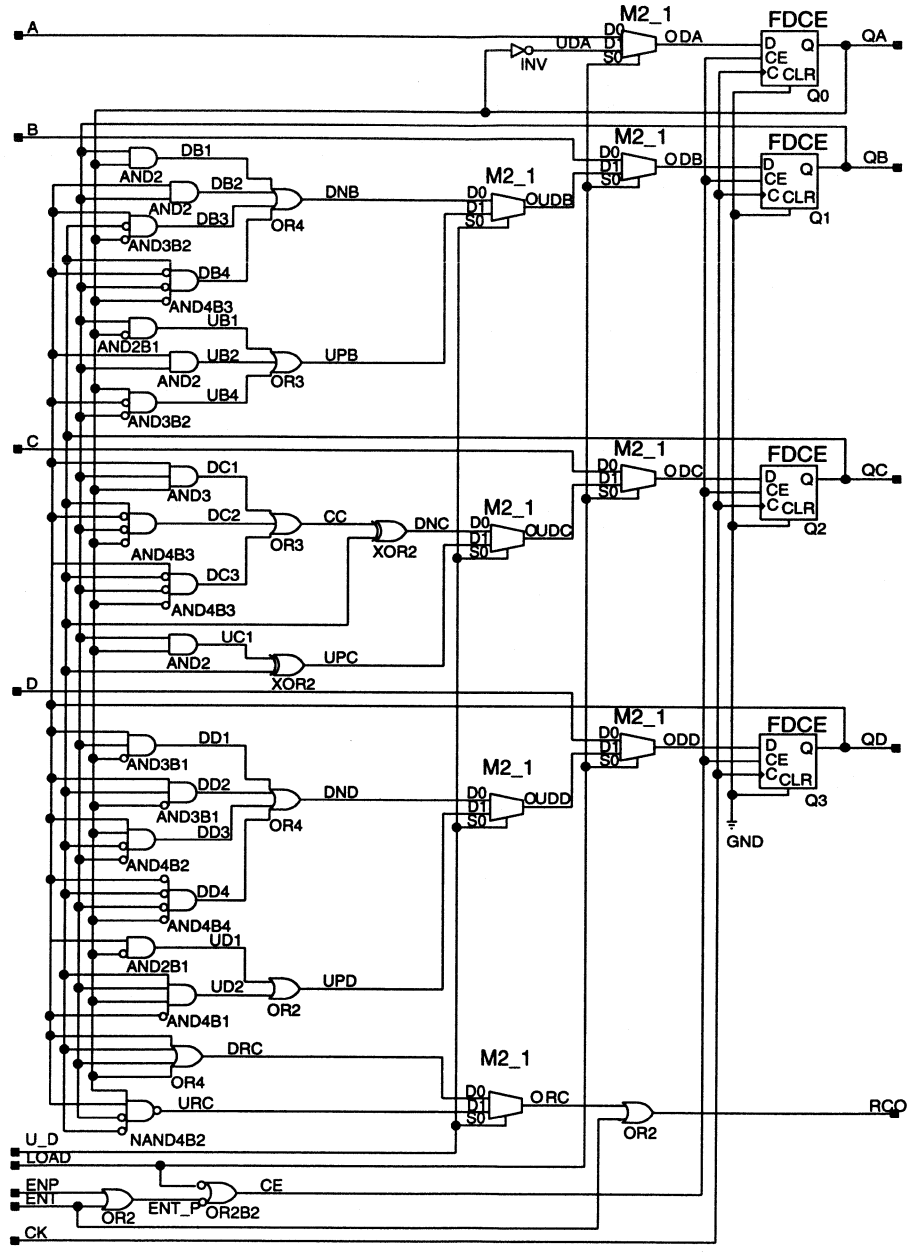
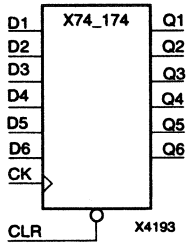


Figure 3-227 X74_168 XC2000/3000/4000 Implementation

X74_174

6-Bit Data Register with Active-Low Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The active-Low asynchronous clear input (CLR), when Low, overrides the clock and resets the six data outputs (Q6 – Q1) Low. When CLR is High, the data on the six data inputs (D6 – D1) is transferred to the corresponding data outputs on the Low-to-High clock (C) transition.

Inputs			Outputs
CLR	D6 – D1	CK	Q6 – Q1
0	X	X	0
1	D6 – D1	↑	d6 – d1

dn = state of referenced input one set-up time prior to active clock transition

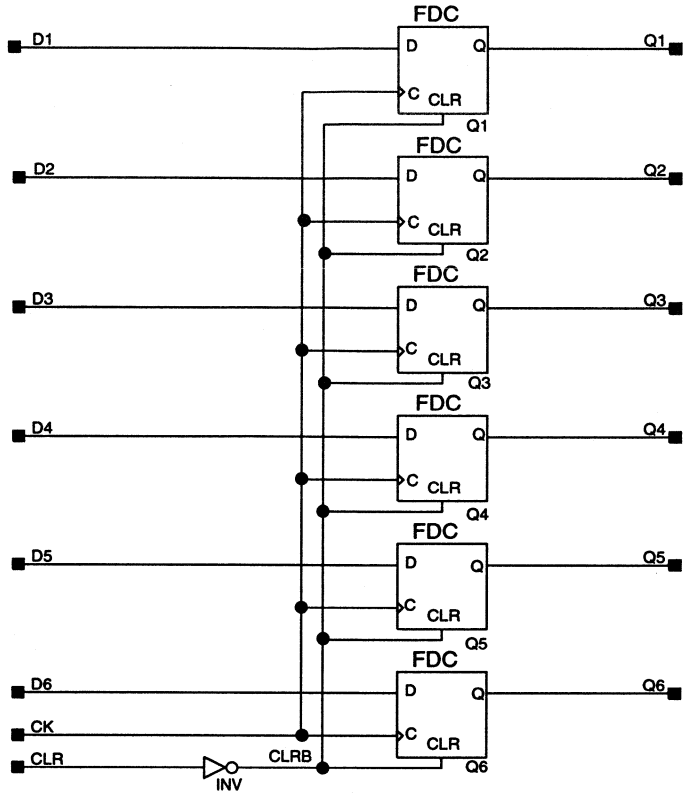
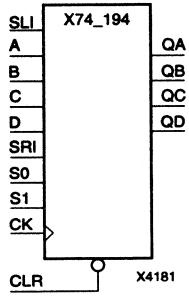


Figure 3-228 X74_174 XC2000/3000/4000 Implementation

X74_194

4-Bit Loadable Bidirectional Serial/Parallel-In Parallel-Out Shift Register



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_194 is a 4-bit shift register with shift-right serial input (SRI), shift-left serial input (SLI), parallel inputs (D – A), parallel outputs (QD – QA), two control inputs (S1, S0), and active-Low asynchronous clear (CLR). The shift register performs the following functions.

- Clear — When CLR is Low, all other inputs are ignored and outputs QD – QA go to logic state zero during the Low-to-High clock transition.
- Load — When S1 and S0 are High, the data on inputs D – A is loaded into the corresponding output bits QD – QA during the Low-to-High clock transition.
- Shift Right — When S1 is Low and S0 is High, the data is shifted to the next-highest bit position (right) as new data is loaded into QA (SRI→QA, QA→QB, QB→QC, and so forth).
- Shift Left — When S1 is High and S0 is Low, the data is shifted to the next-lowest bit position (left) as new data is loaded into QD (SLI→QD, QD→QC, QC→QB, and so forth).

Registers can be cascaded by connecting the QD output of one stage to the SRI input of the next stage, the QA output of one stage to the SRI input of the next stage, and connecting clock, S1, S0, and CLR inputs in parallel.

Inputs							Outputs			
CLR	**S1**	**S0**	**SRI**	**SLI**	**A – D**	**CK**	**QA**	**QB**	**QC**	**QD**
0	X	X	X	X	X	X	0	0	0	0
1	0	0	X	X	X	X	-----No Change--------			
1	1	1	X	X	A – D	↑	a	b	c	d
1	0	1	SRI	X	X	↑	sri	qa	qb	qc
1	1	0	X	SLI	X	↑	qb	qc	qd	sli

Lowercase letters represent state of referenced input or output one set-up time prior to active clock transition.

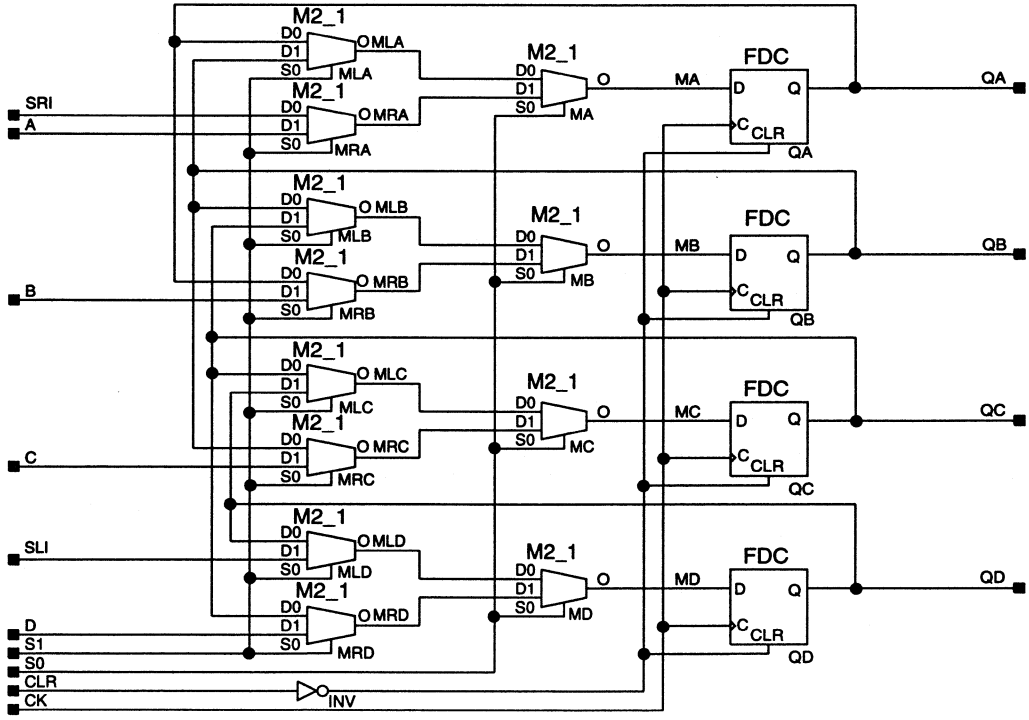
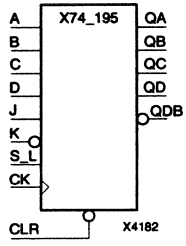


Figure 3-229 X74_194 XC2000/3000/4000 Implementation

X74_195

4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_195 is a 4-bit shift register with shift-right serial inputs (J and K), parallel inputs (D – A), parallel outputs (QD – QA) and QDB, shift/load control input (S_L), and active-Low asynchronous clear (CLR). Asynchronous CLR, when Low, overrides all other inputs and resets data outputs QD – QA Low and QDB High. When S_L is Low and CLR is High, data on the D – A inputs is loaded into the corresponding QD – QA bits of the register during the Low-to-High clock (C) transition. When S_L and CLR are High, the first bit of the register (QA) responds to the J and K inputs during the Low-to-High clock transition, as shown in the truth table. During subsequent Low-to-High clock transitions, with S_L and CLR High, the data is shifted to the next-highest bit position (shift right) as new data is loaded into QA (J, K→QA, QA→QB, QB→QC, and so forth).

Registers can be cascaded by connecting the QD and QDB outputs of one stage to the J and K inputs, respectively, of the next stage and connecting clock, S_L and CLR inputs in parallel.

Inputs						Outputs				
CLR	S_L	J	K	A – D	CK	QA	QB	QC	QD	QDB
0	X	X	X	X	X	0	0	0	0	1
1	0	X	X	A – D	↑	a	b	c	d	$\overline{d}$
1	1	0	0	X	↑	0	qa	qb	qc	$\overline{qc}$
1	1	1	1	X	↑	1	qa	qb	qc	$\overline{qc}$
1	1	0	1	X	↑	qa	qa	qb	qc	$\overline{qc}$
1	1	1	0	X	↑	$\overline{qa}$	qa	qb	qc	$\overline{qc}$

Lowercase letters represent state of referenced input or output one set-up time prior to active clock transition.

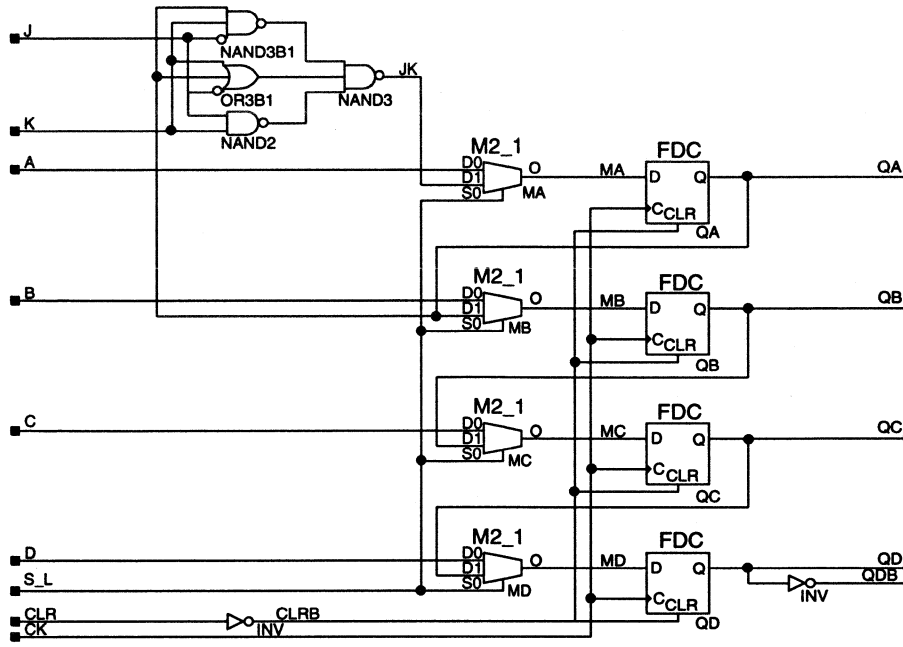
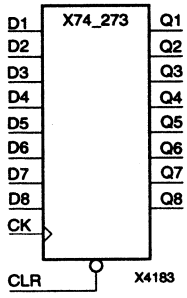


Figure 3-230 X74_195 XC2000/3000/4000 Implementation

X74_273

8-Bit Data Register with Active-Low Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

The active-Low asynchronous clear (CLR), when Low, overrides all other inputs and resets the data outputs (Q8 – Q1) Low. When CLR is High, the data on the data inputs (D8 – D1) is transferred to the corresponding data outputs (Q8 – Q1) during the Low-to-High clock transition.

Inputs			Outputs
CLR	D8 – D1	CK	Q8 – Q1
0	X	X	0
1	D8 – D1	↑	d8 – d1

dn = state of referenced input one set-up time prior to active clock transition

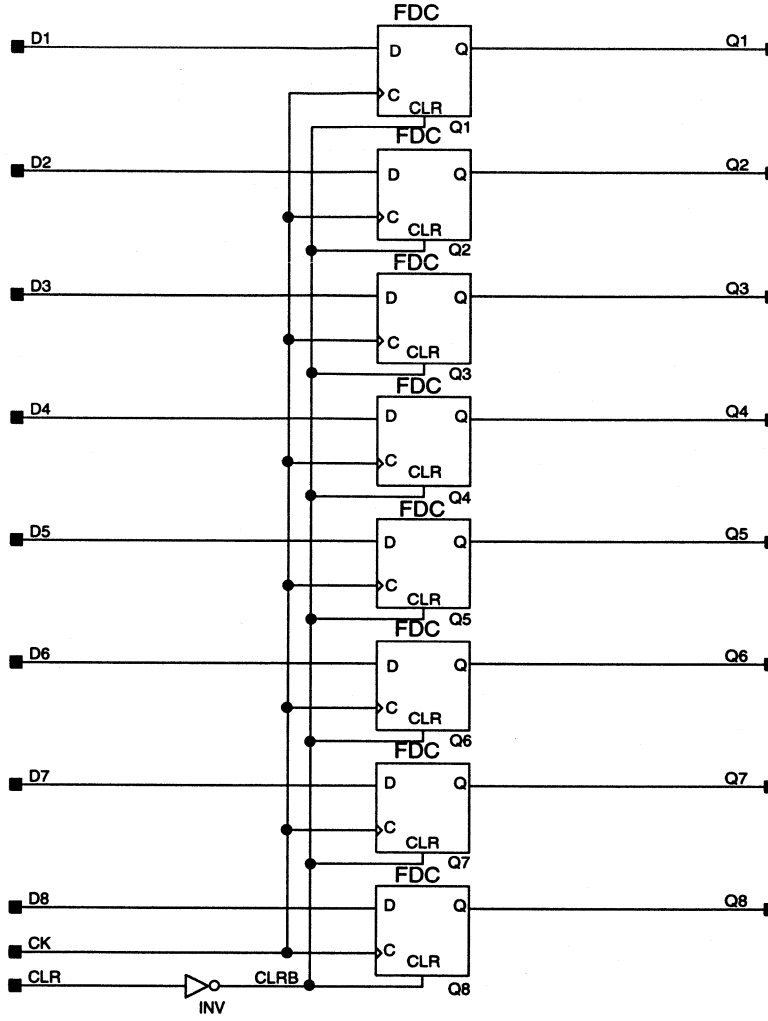
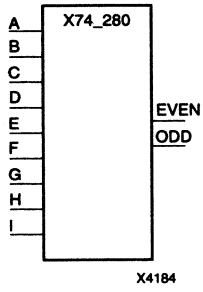


Figure 3-231 X74_273 XC2000/3000/4000 Implementation

X74_280

9-Bit Odd/Even Parity Generator/Checker



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive*

* not supported for XC7336 designs

X74_280 parity generator/checker compares up to nine data inputs (I – A) and provides both even (EVEN) and odd parity (ODD) outputs. The EVEN output is High when an even number of inputs is High. The ODD output is High when an odd number of inputs is High.

Expansion to larger word sizes is accomplished by tying the ODD outputs of up to nine parallel components to the data inputs of one more X74_280; all other inputs are tied to ground.

Inputs	Outputs	
Number of Ones on A – I	EVEN	ODD
0, 2, 4, 6, or 8	1	0
1, 3, 5, 7, or 9	0	1

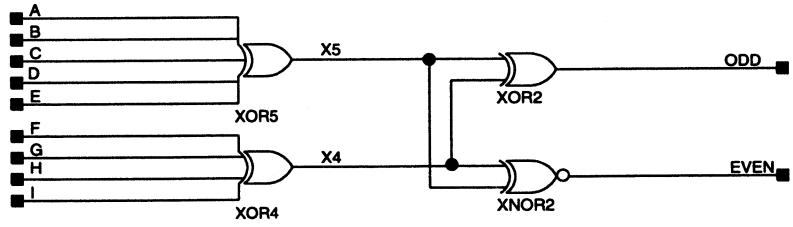
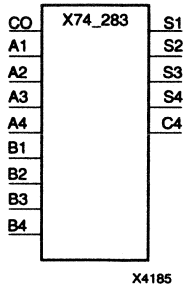


Figure 3-232 X74_280 XC2000/3000/4000 Implementation

X74_283

4-Bit Full Adder with Carry-In and Carry-Out



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive*

* not supported for XC7336 designs

X74_283, a 4-bit full adder with carry-in and carry-out, adds two 4-bit words (A4 – A1 and B4 – B1) and a carry-in (C0) and produces a binary sum output (S4 – S1) and a carry-out (C4).

16(C4)+8(S4)+4(S3)+2(S2)+S1=8(A4+B4)+4(A3+B3)+2(A2+B2)+(A1+B1)+CO, where "+" = addition.

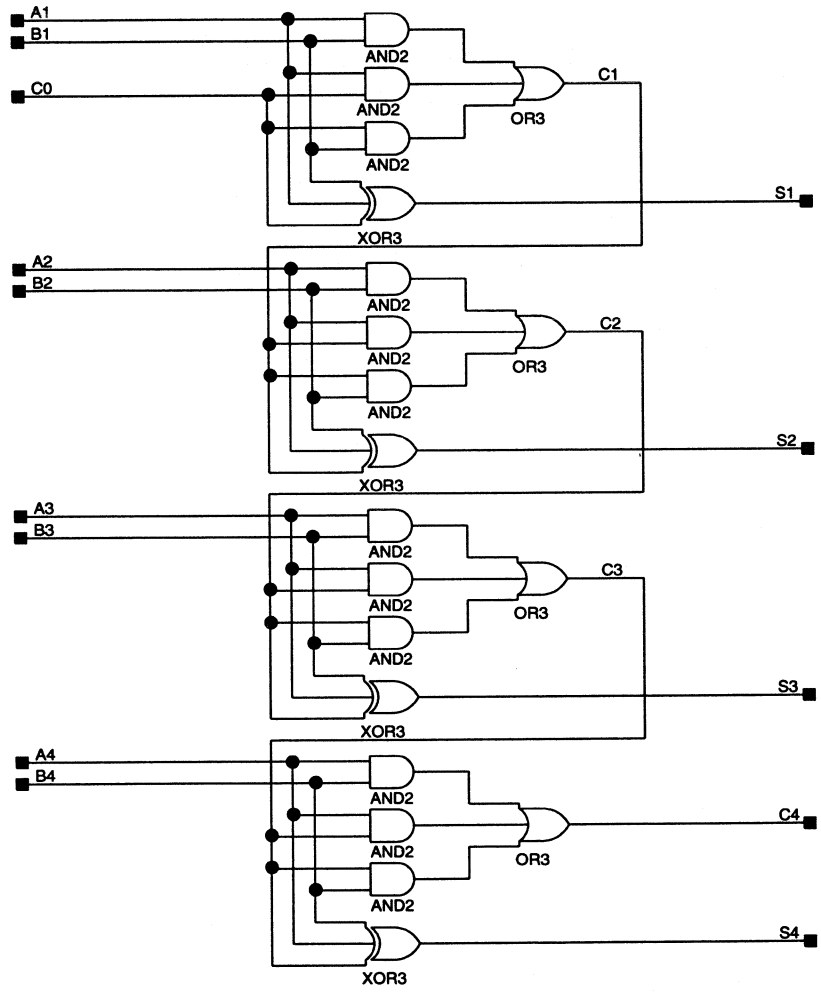
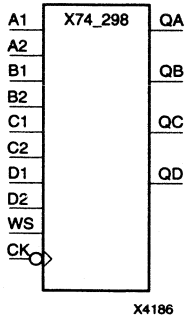


Figure 3-233 X74_283 XC2000/3000/4000 Implementation

X74_298

Quadruple 2-Input Multiplexer with Storage and Negative-Edge Clock



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive*

* not supported for XC7336 designs

X74_298 selects 4-bits of data from two sources (D1 – A1 or D2 – A2) under the control of a common word select input (WS). When WS is Low, D1 – A1 is chosen, and when WS is High, D2 – A2 is chosen. The selected data is transferred into the output register (QD – QA) during the High-to-Low transition of the negative-edge triggered clock (CK). For XC7000, the CK input cannot be driven by a FastCLK signal (from BUFG).

Inputs				Outputs
WS	**A1 – D1**	**A2 – D2**	**CK**	**QA – QD**
0	A1 – D1	X	↓	a1 – d1
1	X	A2 – D2	↓	a2 – d2

an – dn = state of referenced input one set-up time prior to active clock transition

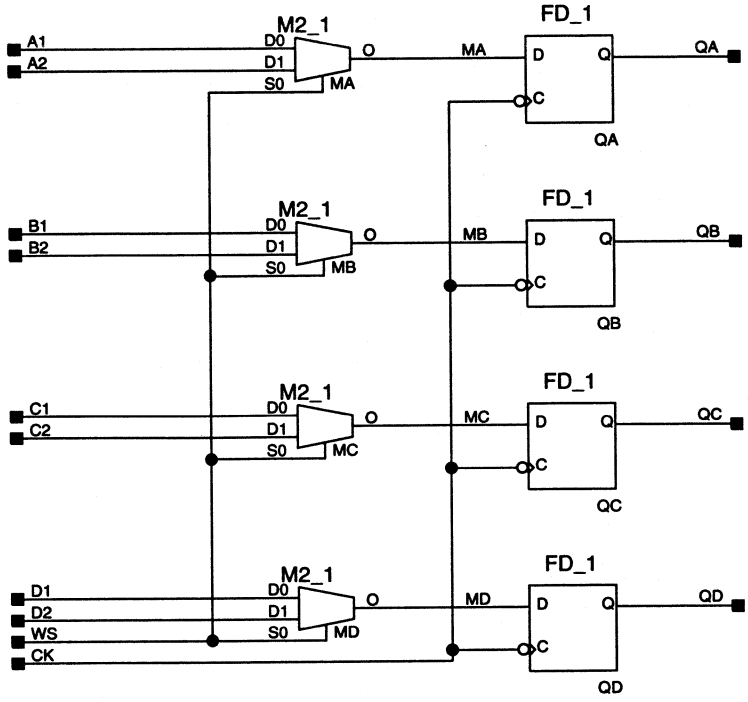
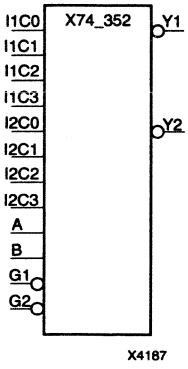


Figure 3-234 X74_298 XC2000/3000/4000 Implementation

X74_352

Dual 4-to-1 Multiplexer with Active-Low Enables and Outputs



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_352 comprises two 4-to-1 multiplexers with separate enables (G1 and G2) but with common select inputs (A and B). When an active-Low enable (G1 or G2) is Low, the multiplexer chooses one data bit from the four sources associated with the particular enable (I1C3 – I1C0 for G1 and I2C3 – I2C0 for G2) under the control of the common select inputs (A and B). The active-Low outputs (Y1 and Y2) reflect the inverse of the selected data as shown in truth table. Y1 is associated with G1 and Y2 is associated with G2. When an active-Low enable is High, the associated output is High.

Inputs							Outputs
G	**B**	**A**	**IC0**	**IC1**	**IC2**	**IC3**	**Y**
1	X	X	X	X	X	X	1
0	0	0	IC0	X	X	X	$\overline{IC0}$
0	0	1	X	IC1	X	X	$\overline{IC1}$
0	1	0	X	X	IC2	X	$\overline{IC2}$
0	1	1	X	X	X	IC3	$\overline{IC3}$

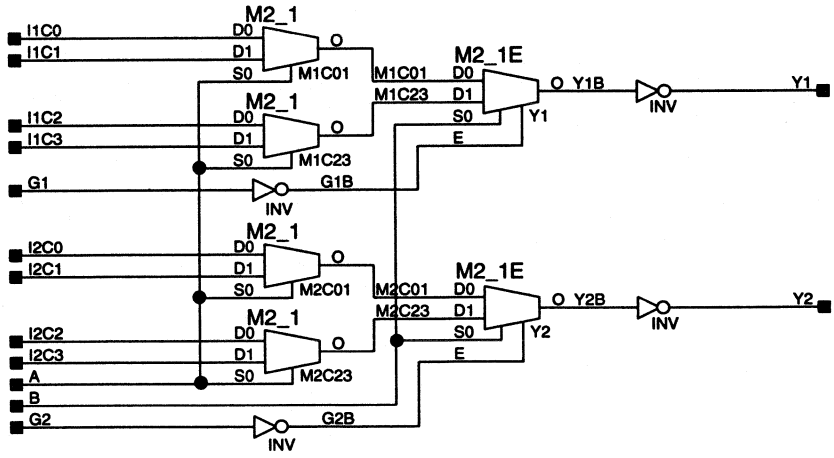
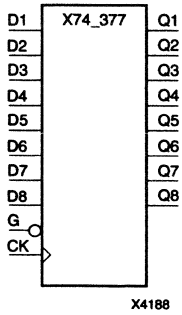


Figure 3-235 X74_352 XC2000/3000/4000 Implementation

X74_377

8-Bit Data Register with Active-Low Clock Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

When the active-Low clock enable (G) is Low, the data on the eight data inputs (D8 – D1) is transferred to the corresponding data outputs (Q8 – Q1) during the Low-to-High clock (CK) transition. The register ignores clock transitions when G is High.

Inputs			Outputs
G	**D8 – D1**	**CK**	**Q8 – Q1**
1	X	X	No Change
0	D8 – D1	↑	d8 – d1

dn = state of referenced input one set-up time prior to active clock transition

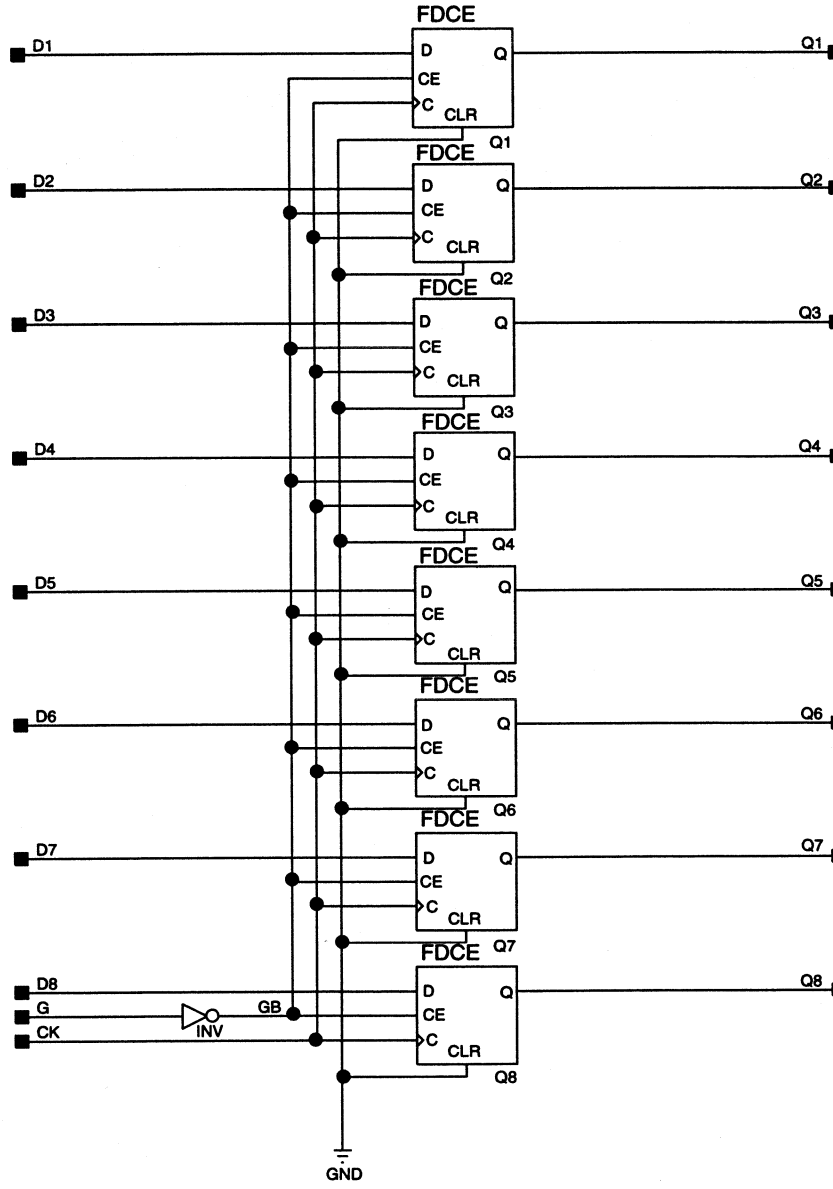
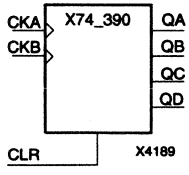


Figure 3-236 X74_377 XC2000/3000/4000 Implementation

X74_390

4-Bit BCD/Bi-Quinary Ripple Counter with Negative-Edge Clocks and Asynchronous Clear



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive*

* not supported for XC7336 designs

X74_390 is a cascadable, resettable binary-coded decimal (BCD) or bi-quinary counter that can be used to implement cycle lengths equal to whole and/or cumulative multiples of 2 and/or 5. In BCD mode, the output QA is connected to negative-edge clock input (CKB), and data outputs (QD – QA) increment during the High-to-Low clock transition as shown in the truth table, provided asynchronous clear (CLR) is Low. In bi-quinary mode, output QD is connected to the negative-edge clock input (CKA). As shown in the truth table, in bi-quinary mode, QA supplies a divide-by-five output and QB supplies a divide-by-two output, provided asynchronous CLR is Low. When asynchronous CLR is High, the other inputs are overridden, and data outputs (QD – QA) are reset Low.

Larger ripple counters are created by connecting the QD output (BCD mode) or QA output (biquinary mode) of the first stage to the appropriate clock input of the next stage and connecting the CLR inputs in parallel. For XC7000, CKA and CKB cannot be driven by a FastCLK signal from (BUFG).

Count	BCD				Bi-Quinary			
	QD	QC	QB	QA	QD	QC	QB	QA
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	1	0	0
3	0	0	1	1	0	1	1	0
4	0	1	0	0	1	0	0	0
5	0	1	0	1	0	0	0	1
6	0	1	1	0	0	0	1	1
7	0	1	1	1	0	1	0	1
8	1	0	0	0	0	1	1	1
9	1	0	0	1	1	0	0	1

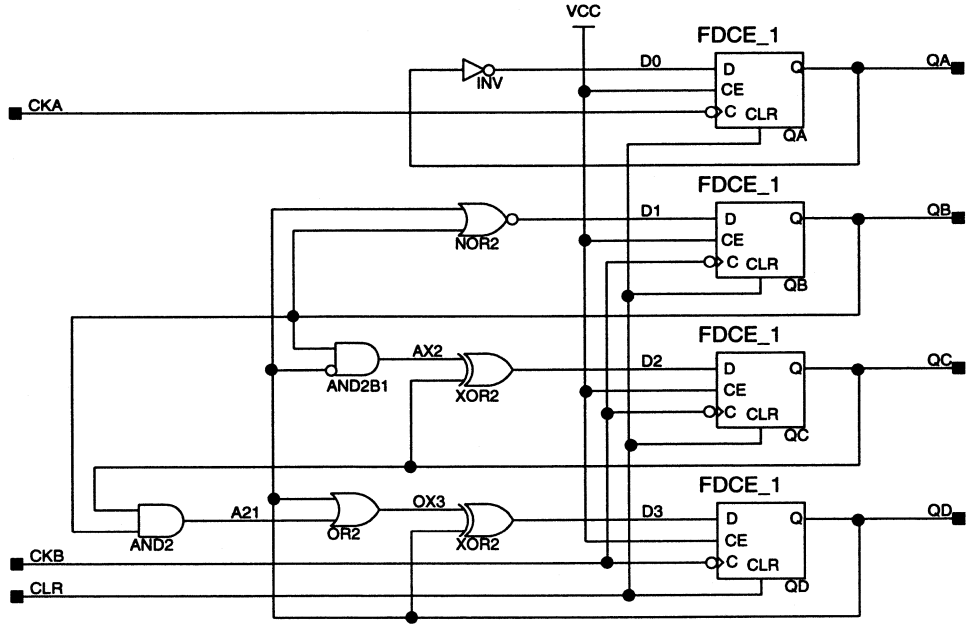
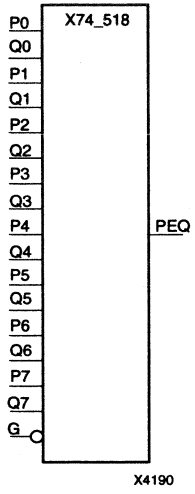


Figure 3-237 X74_390 XC2000/3000/4000 Implementation

X74_518

8-Bit Identity Comparator with Active-Low Enable



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_518 is an 8-bit identity comparator with 16 data inputs for two 8-bit words (P7 – P0 and Q7 – Q0), data output (PEQ), and active-Low enable (G). When G is High, the PEQ output is Low. When G is Low and the two input words are equal, PEQ is High. Equality is determined by a bit comparison of the two words. When any of the two equivalent bits from the two words are not equal, PEQ is Low.

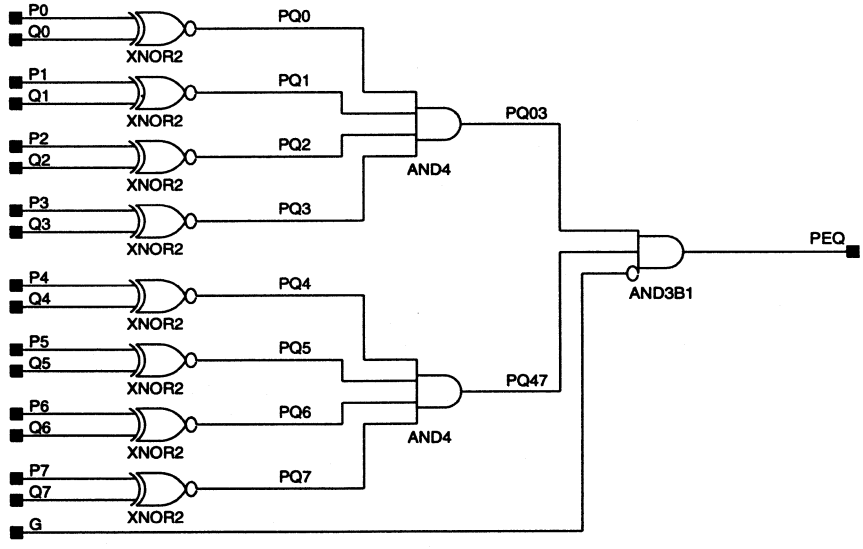
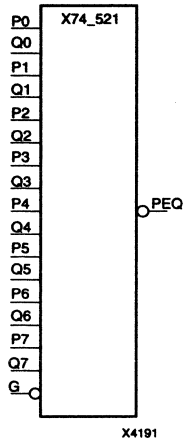


Figure 3-238 X74_518 XC2000/3000/4000 Implementation

X74_521

8-Bit Identity Comparator with Active-Low Enable and Output



XC2000	XC3000	XC4000	XC7000
Macro	Macro	Macro	Primitive

X74_521 is an 8-bit identity comparator with 16 data inputs for two 8-bit words (P7 – P0 and Q7 – Q0), active-Low data output (PEQ), and active-Low enable (G). When G is High, the PEQ output is High. When G is Low and the two input words are equal, PEQ is Low. X74_521 does a bit comparison of the two words to determine equality. When any of the two equivalent bits from the two words are not equal, PEQ is High.

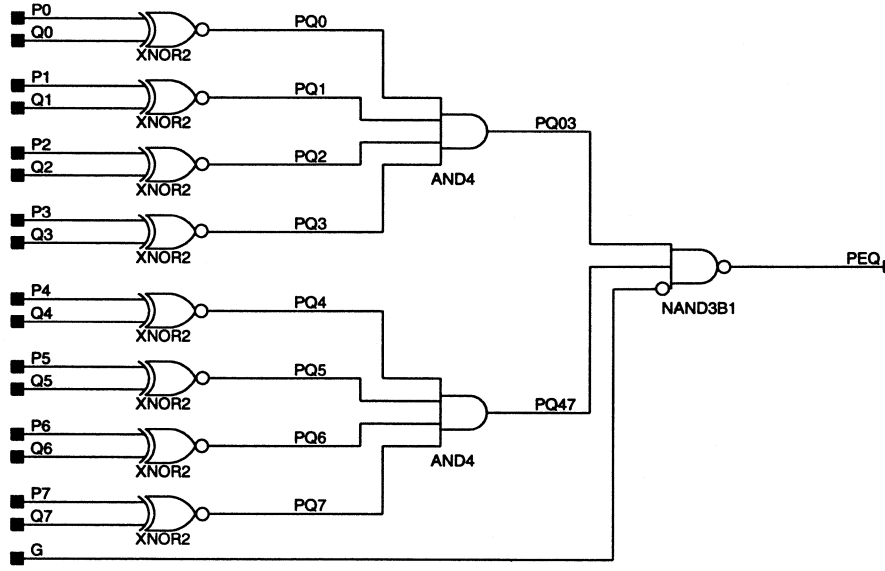
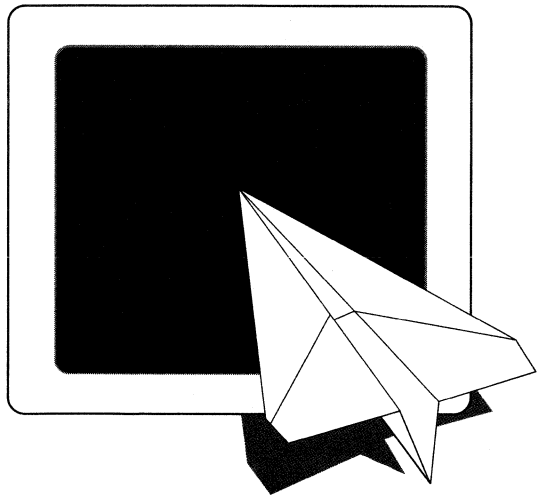


Figure 3-239 X74_521 XC2000/3000/4000 Implementation



Attributes, Constraints, and Carry Logic

XACT Libraries Guide

XACT Libraries Guide — April, 1994 (0401098 01)

Printed in U.S.A.

Chapter 4

Attributes, Constraints, and Carry Logic

This chapter lists and describes all the attributes and constraints that you can use with your schematic entry software or enter in a constraints file. In particular, it describes the relative location (RLOC) constraint. It also describes PPR placement constraints, relationally placed macros (RPMs), and carry logic.

Attributes are instructions placed on symbols or nets in an FPGA or EPLD schematic to indicate their placement, implementation, naming, directionality, and so forth. This information is used by the design implementation software during placement and routing of a design. Constraints, which are a type, or subset, of attributes, are used only to indicate where an element should be placed.

All the attributes listed in this chapter are available in the schematic entry tools directly supported by Xilinx unless otherwise noted, but some may not be available in textual entry methods such as VHDL.

Attributes applicable only to a certain schematic entry tool are described in the documentation for that tool. For third-party interfaces, consult the interface user guides for information on which attributes are available and how they are used.

Attributes

There are three types of attributes discussed in this section:

- Component attributes, which affect only the component instances on which they are placed.
- Global attributes, which affect the entire design. These attributes apply to EPLD devices only.
- Net attributes, which affect individual component outputs or inputs and are represented by attributes applied to nets.

In some software programs, particularly Mentor Graphic's, attributes are called properties, but their functionality is the same as that of attributes. In this document, they are referred to as attributes.

There are two types of Mentor Graphics properties: in one, a property is equal to a value, for example, LOC=AA; in the other, the property name and the value are the same, for example, DECODE. In the first type, "attribute" refers to the property; in the second, "attribute" refers to the property and the value.

The attributes in this section are listed in alphabetical order.

BASE

Architectures

The BASE attribute applies to the XC2000 and XC3000 families only.

Description

The BASE attribute defines the base configuration of a CLB or an IOB. For an IOB primitive, it should always be set to IO. For a CLB primitive, it can be one of three modes in which the CLB function generator operates.

In XC2000 devices, these three modes are the following:

- F mode allows any function of up to four variables to be implemented, where one of the inputs can be the Q output of the flip-flop in the CLB.
- FG mode allows two three-input functions to be implemented, where the input can be chosen from the four inputs to the CLB and the Q output of the flip-flop in the CLB.
- FGM mode is similar to FG mode except that the inputs must be chosen from four inputs to the CLB or the Q feedback. The B input to the CLB acts as the control for a multiplexer between the two four-input functions.

The three modes are very similar in XC3000 devices:

- F mode allows the CLB to implement any one function of up to five variables.
- FG mode gives the CLB any two functions of up to four variables.

Of the two sets of four variables, one input (A) must be common, two (B and C) can be either independent inputs or feedback from the Qx and Qy outputs of the flip-flops within the CLB, and the fourth can be either of the two other inputs to the CLB (D and E).

- FGM mode is similar to FG, but the fourth input must be the D input. The E input is then used to control a multiplexer between the two four-input functions, allowing some six- and seven-input functions to be implemented.

The following two figures illustrate the CLB and IOB base configurations of the XC2000 and XC3000 families, respectively. In these drawings, BASE F, FG, and FGM are for CLBs; BASE IO is for IOBs.

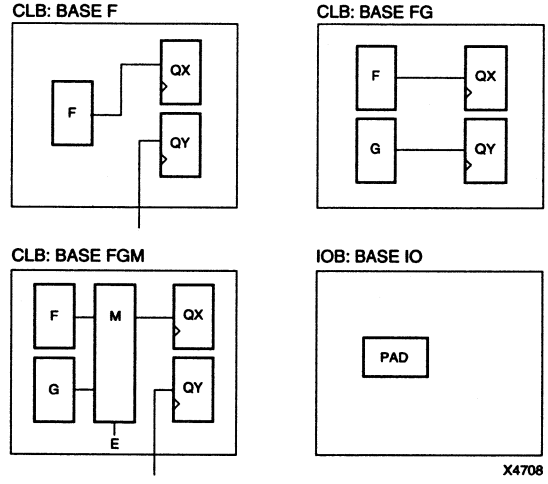


Figure 4-1 Base Configurations for XC3000 CLB and IOB Primitives

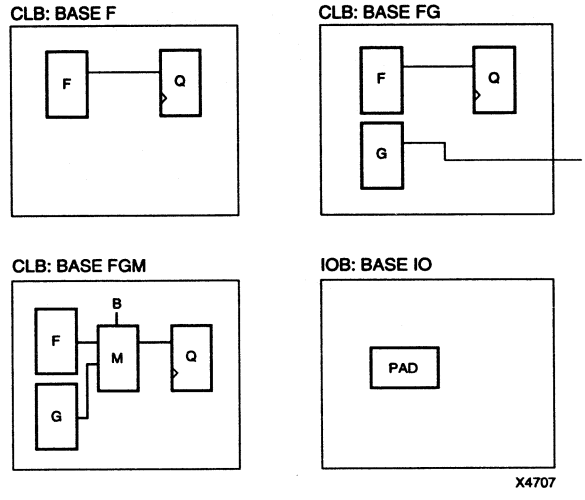


Figure 4-2 Base Configurations for XC2000 CLB and IOB Primitives

Syntax

The syntax of the BASE attribute is the following:

BASE=*mode*

where *mode* can be F, FG, or FGM for a CLB, or IO for an IOB.

BLKNM

Architectures

The BLKNM attribute applies to all FPGA families.

Description

The BLKNM attribute assigns LCA block names to qualifying primitives and logic elements. If the same BLKNM attribute is assigned to more than one instance, the software attempts to map them into the same LCA block. Conversely, two symbols with different BLKNM names are not mapped into the same block. Placing

similar BLKNMs on instances that do not fit within one LCA block creates an error.

Specifying identical BLKNM attributes on FMAP and/or HMAP symbols tells the software to group the associated function generators into a single CLB. Using BLKNM, you can partition a complete CLB without constraining the CLB to a physical location on the device.

For an XC4000 CLB, the maximum number of elements that can be assigned the same block name is two flip-flops, two FMAPs, and one HMAP. For an XC3000 CLB, the maximum number of elements that can be assigned the same block name is two flip-flops or one CLBMAP. For an XC2000 CLB, the maximum number is one flip-flop or one CLBMAP.

BLKNM attributes, like LOC constraints, are specified from the schematic. Hierarchical paths are not prefixed to BLKNM attributes, so BLKNM attributes for different CLBs must be unique throughout the entire design. See the section on the HBLKNM attribute for information on attaching hierarchy to block names.

Use the BLKNM attribute to attach a name to the following symbols:

- XC4000 flip-flop primitives (FDCE, FDPE)
- XC3000 flip-flop primitives (FDCE)
- XC2000 flip-flop and latch primitives (FDCP, LDCP)
- I/O buffers, flip-flops, and latches (IBUF, OBUF, OBUFT, ILD, IFD, OFD, OFDT)
- PAD primitives (PAD, IPAD, OPAD, BPAD, UPAD, PADU)
- I/O block primitives (IOB symbols)
- Configurable logic blocks (CLB symbols)
- 3-state buffers (BUFT symbols)
- Mapping control symbols (CLBMAP, FMAP, HMAP)

Syntax

The syntax of the BLKNM attribute is the following:

BLKNM=*blockname*

where *blockname* is a valid LCA block name for that type of symbol. For a list of prohibited block names, see the "Naming Conventions" section of each user interface manual.

For information on assigning hierarchical block names, see the HBLKNM attribute description in this chapter.

Example

Suppose that you want to map together two flip-flops within one CLB. You give both the BLKNM=FFGRP1 attribute. You then translate, place, and route the design. When you examine it in EditLCA, you see that both flip-flops reside within a CLB named FFGRP1.

CAP

Architectures

The CAP attribute applies to the XC4000H family only.

Description

You can specify an XC4000H output driver as operating in either resistive (RES) or capacitive, "softedge" (CAP) mode. In resistive mode, the output is faster and draws more power. Use this mode when the output is attached to purely resistive loads, or when ground bounce is not expected to be a problem with the output.

The CAP attribute allows you to specify capacitive mode. Use capacitive mode when connecting an output to a capacitive mode, or when ground bounce is predicted to be a problem with the output. In capacitive mode, the pull-down transistor is slowly turned off as the output is pulled to ground, minimizing the likelihood of ground bounce.

See the section on the RES attribute for more information.

Use the CAP attribute on the following symbols:

- IOB output symbols OBUF, OBUFT
- IOB pads OPAD, IOPAD, UPAD
- Special function access symbols TDI, TMS, TCK

Syntax

The CAP attribute has the following syntax:

```
CAP
```

CLOCK_OPT

Architectures

The CLOCK_OPT attribute applies to the XC7200 and XC7300 families only.

Description

The CLOCK_OPT global attribute controls FastCLK optimization for the entire design. FastCLK optimization changes a product-term clock to a FastCLK global signal, which reduces the number of universal interconnect matrix (UIM) inputs and product terms required by each function block.

Syntax

Use the following syntax with the CLOCK_OPT attribute:

```
CLOCK_OPT={on|off}
```

The On setting enables FastCLK optimization; the Off setting inhibits it. On is the default.

CMOS

Architectures

The CMOS attribute applies to the XC4000H family only.

Description

The CMOS attribute configures output drivers on the XC4000H to drive to CMOS-compatible levels. Similarly, it configures IOBs to have CMOS-compatible input thresholds.

To configure output drive levels, attach the CMOS attribute to any of the following output symbols: OBUF, OBUFT, OUTFF/OFD, OUTFFT/OFDT.

To configure input threshold levels, attach the CMOS attribute to any of the following input symbols: IBUF, INFF/IFD, INLAT/ILD, INREG.

See the section on the TTL attribute for more information.

Syntax

The syntax of the CMOS attribute is the following:

```
CMOS
```

CONFIG

Architectures

The CONFIG attribute applies to XC2000 and XC3000 families only.

Description

The CONFIG attribute specifies logic element inputs and the storage element function for a CLB or IOB symbol.

CONFIG attributes can only be attached to IOB and CLB symbols.

Syntax

Use the following syntax for the CONFIG attribute:

CONFIG=*tag*: [*value*] : [*value*]

where *tag* and *value* are derived from the following tables.

Table 4-1 XC2000 CLB Configuration Options

Tag	BASE F	BASE FG	BASE FGM*
X	F, Q	F, G, Q	M, Q
Y	F, Q	F, G, Q	M, Q
Q	FF, LATCH	FF, LATCH	FF, LATCH
SET	A, F	A, F	A, M
RES	D, F	D, G	D, M
CLK	K, C, F, NOT	K, C, G, NOT	K, C, M, NOT
F	A, B, C, D, Q	A, B, C, D, Q	A, B, C, D, Q
G	None	A, B, C, D, Q	A, B, C, D, Q

*For BASE FGM, M=F if B=1, and M=G if B=0.

Table 4-2 XC2000 IOB Configuration Options

Tag	BASE IO
I	PAD, Q
BUF	ON, TRI

Table 4-3 XC3000 CLB Configuration Options

Tag	BASE F	BASE FG	BASE FGM*
X	F, QX	F, QX	M, QX
Y	F, QY	G, QY	M, QY
DX	DI, F	DI, F, G	DI, M
DY	DI, F	DI, F, G	DI, M
CLK	K, NOT	K, NOT	K, NOT
RSTDIR	RD	RD	RD
ENCLK	EC	EC	EC
F	A,B,C,D,E,QX, QY	A,B,C,D,E,QX, QY	A,B,C,D,QX, QY
G	None	A,B,C,D,E,QX, QY	A,B,C,D,QX, QY

*For BASE FGM, M=F if E=0, and M=G if E=1.

Table 4-4 XC3000 IOB Configuration Options

Tag	BASE IO
IN	I, IQ, IKNOT, FF, LATCH, PULLUP
OUT	O, OQ, NOT, OKNOT, FAST
TRI	T, NOT

Example

Following is an example of a valid XC2000 CLB CONFIG attribute value:

```
X:Q Y:G CLK:K:NOT Q:FF SET:A RES:D
```

Here is an example of a valid XC3000 CLB CONFIG attribute value:

```
X:QX Y:QY DX:F DY:G CLK:K ENCLK:EC
```

DECODE

Architectures

The DECODE attribute applies to the XC4000 family only.

Description

The DECODE attribute defines where a wired-AND (WAND) instance is placed, either within a BUFT or an edge decoder. If the DECODE attribute is placed on a single-input WAND1 gate, the gate is implemented as an input to a wide-edge decoder in an XC4000 design.

Syntax

The syntax of the DECODE attribute is the following:

```
DECODE
```

DECODE attributes can only be attached to a WAND1 symbol.

DOUBLE

Architectures

The DOUBLE attribute applies to the XC3000 family only.

Description

The DOUBLE attribute specifies double pull-up resistors on the horizontal longline. On XC3000 parts, there are internal nets that can be set as 3-state with two programmable pull-up resistors available per line. If the DOUBLE attribute is placed on a PULLUP symbol, both pull-ups are used, enabling a fast, high-power line. If the DOUBLE attribute is not placed on a pull-up, only one pull-up is used, resulting in a slower, lower-power line.

When the DOUBLE attribute is present, the speed of the distributed logic is increased, as is the power consumption of the part. When only half of the longline is used, there is only one pull-up at each end of the longline.

While the DOUBLE attribute can be used for the XC4000 family, it is not recommended. PPR activates both pull-up resistors if the entire longline (not a half-longline) is used.

Syntax

The syntax of the DOUBLE attribute is the following:

DOUBLE

The DOUBLE attribute can only be attached to a BUFT symbol.

EQUATE_F and EQUATE_G

Architectures

The EQUATE_F and EQUATE_G attributes apply to the XC2000 and XC3000 families only.

Description

The EQUATE_F and EQUATE_G attributes set the logic equations describing the F and G function generators of a CLB, respectively.

Syntax

The syntax of the EQUATE_F and EQUATE_G attributes is the following:

EQUATE_F or **EQUATE_G**

The following table lists the Boolean operators used in the logic equations.

Table 4-5 Valid XC2000 and XC3000 Boolean Operators

Symbol	Boolean Equivalent
~	NOT
*	AND
@	XOR
+	OR
( )	Group expression

Example

Here are two examples illustrating the use of the EQUATE_F attribute:

```
EQUATE_F=F=((~A*B)+D))@Q

F=A@B+(C*~D)
```

FAST

Architectures

The FAST attribute applies to XC3000, XC3000A/L, XC4000, and XC4000A families only.

Description

The FAST slew-rate attribute is attached to an output buffer, output flip-flop, or pad to increase the speed of an IOB output. It produces a faster output but may increase noise and power consumption.

The FAST attribute can be attached to the following symbols:

- IOB symbols OBUF, OBUFT, OFD, OFDI, OFDT, OFDTI, OPAD, IOPAD, UPAD
- Special function access symbols TDI, TMS, TCK

Syntax

The syntax of the FAST attribute is the following:

```
FAST
```

FILE

Architectures

The FILE attribute applies to all FPGA families.

Description

The FILE attribute is placed on symbols that do not have underlying schematics. It references the XNF file containing the Xilinx netlist for

the logic represented by the symbol. When XNFMerge encounters such a symbol, it looks in the design directory for the XNF file and replaces the description of the symbol in the XNF file with the functionality found in the XNF file.

Syntax

Use the following syntax for the FILE attribute:

FILE=*filename*

where *filename* is the name of an XNF file without the .xnf extension.

Example

Suppose that a symbol is created, called new_and2, whose function mimics that of a 2-input AND gate. A Xilinx ABEL file describes the function of the new_and2 symbol and is translated to an XNF file called new_and2.xnf. A FILE attribute is placed on the symbol, and the attribute is given a value of new_and2. The top-level design containing the new_and2 symbol is translated to an XNF file, and the following lines are found within it:

```
SYM, I$2, NEW_AND2, FILE=NEW_AND2
PIN, I1, I, NET_IN1
PIN, I2, I, NET_IN2
PIN, O1, O, NET_OUT1
END
```

The new_and2.xnf file contains the following lines:

```
SYM, I$1, AND2
PIN, 1, I, I1
PIN, 2, I, I2
PIN, O, O, O1
END
```

The top-level file is then processed by XNFMerge, which reads new_and2.xnf and replaces the description of the symbol with the description of the functionality, resulting in the following lines in the top-level design:

```
SYM, I$2/I$1, AND2
PIN, 1, I, NET_IN1
PIN, 2, I, NET_IN2
PIN, O, O, NET_OUT1
END
```

The functionality of the symbol is added to the top-level design, while the connectivity found in the top-level design is maintained.

FOE_OPT

Architectures

The FOE_OPT attribute applies to the XC7200 and XC7300 families only.

Description

The FOE_OPT global attribute controls the optimization of the fast output enable (FOE) for the entire design. FOE optimization generally applies only to BUFE, OBUFE, or 3-state PLD outputs driving an OBUF. FOE optimization changes a product-term 3-state signal to an FOE global control signal. Like FastCLK assignment, it reduces the number of UIM inputs and product terms required by each function block.

Syntax

Use the following syntax with the FOE_OPT attribute:

```
FOE_OPT={on|off}
```

Off inhibits FOE optimization of the entire design, and On, which is the default, activates it.

HBLKNM

Architectures

The HBLKNM attribute applies to all FPGA families.

Description

The HBLKNM attribute assigns hierarchical LCA block names to logic elements and controls grouping in a flattened hierarchical design. When elements on different levels of a hierarchical design carry the same block name and the design is flattened, XNFMerge prefixes a hierarchical path name to the HBLKNM value.

Like BLKNM, the HBLKNM attribute forces function generators and flip-flops into the same CLB. Symbols with the same HBLKNM attribute map into the same CLB, if possible. However, using HBLKNM instead of BLKNM has the advantage of adding hierarchy path names during translation, and therefore the same HBLKNM attribute and value can be used on elements within different instances of the same macro.

Use the HBLKNM attribute to attach a name to the following symbols:

- XC4000 flip-flop primitives (FDCE, FDOP)
- XC3000 flip-flop primitives (FDCE)
- XC2000 flip-flop and latch primitives (FDCP, LDCP)
- I/O buffers, flip-flops, and latches (IBUF, OBUF, OBUFT, ILD, IFD, OFD, OFDT)
- PAD primitives (PAD, IPAD, OPAD, BPAD, UPAD, PADU)
- I/O block primitives (IOB symbols)
- Configurable logic blocks (CLB symbols)
- 3-state buffers (BUFT symbols)
- Mapping control symbols (CLBMAP, FMAP, HMAP)

Syntax

The syntax of the HBLKNM attribute is the following:

HBLKNM=*blockname*

where *blockname* is a valid LCA block name for that type of symbol. For a list of prohibited block names, see the "Naming Conventions" section of each user interface manual.

Example

A schematic is created that contains a four-input function and a flip-flop. The logic function is mapped using an FMAP symbol. Both the FMAP and the flip-flop are given the attribute HBLKNM=GROUP1. A symbol is created to represent the schematic, and both are given the name of FUNC. Another schematic is then created, and four instances of FUNC are placed on it. Because hierarchy is taken into account when the design is translated, the software recognizes four distinct groups, as opposed to one large group called GROUP1, and each instance of FUNC is mapped into a separate CLB.

HU_SET

Architectures

The HU_SET constraint applies to the XC4000 and XC4000A/H families only.

Description

Like the H_SET constraint, the HU_SET constraint is defined by the design hierarchy. However, it also allows you to specify a set name. It is possible to have only one H_SET constraint within a given hierarchical element (macro) but by specifying set names, you can specify several HU_SET sets.

XNFMerge hierarchically qualifies the name of the HU_SET as it flattens the design and attaches the hierarchical names as prefixes. The difference between an HU_SET constraint and an H_SET constraint is that an HU_SET has an explicit user-defined and hierarchically qualified name for the set, but an H_SET constraint has only an implicit hierarchically qualified name generated by the design-flattening program. An HU_SET set "starts" with the symbols

that are assigned the HU_SET constraint, but an H_SET set "starts" with the instantiating macro one level above the symbols with the RLOC constraints.

For detailed information about this attribute, refer to the "Relative Location (RLOC) Constraints" section later in this chapter.

Syntax

To designate a design element as a member of a HU_SET set, apply the following syntax to a design element:

HU_SET=*name*

where *name* is the identifier for the set; it must be unique among all the sets in the design.

INIT

Architectures

The INIT attribute applies to the XC4000 and XC4000A/H families only.

Description

The INIT attribute initializes ROMs.

On a ROM, the INIT attribute gives an initial value to the contents of the ROM. Either four or eight hexadecimal digits are required, depending on the width of the ROM.

Syntax

Use the following syntax to implement the INIT attribute:

INIT=*value*

For ROMs, *value* can be four or eight hexadecimal digits, depending on whether the ROM is a 16- or 32-word-deep ROM, respectively.

LOC

Architectures

The LOC constraint applies to all families.

Description for FPGAs

For FPGAs, the LOC constraint defines where a symbol can be placed within an FPGA. It specifies the absolute placement of a design element on the FPGA die. It can be a single location, a range of locations, or a list of locations. The LOC constraint can only be specified from the schematic. However, statements in a constraints file can also be used to direct placement.

The LOC constraint can be used on the following elements:

- BUFTs
- Elements that map into a CLB: flip-flops, FMAPs, HMAPs, CLBMAPs, CLBs
- Elements that map into an IOB: pads, IBUFs, OBUFs, INFFs, OUTFFs, and so forth
- For XC4000 only, WANDs and clock buffers

If a LOC constraint is placed on a macro symbol, XNFMerge passes it down onto every symbol of the appropriate type underneath that macro. For example, if LOC=CLB_R3C7 is placed on a macro, that LOC constraint is passed to flip-flops and map symbols but not to BUFTs.

You can use the LOC constraint to assign a specific LCA location to the following symbols:

- All flip-flop and latch primitives
- Xilinx soft macros (only flip-flops are affected)
- User-created symbols (only flip-flops are affected)
- Input buffers, output buffers, or pad symbols
- Clock buffers (ACLK, GCLK, BUFGP, BUFGS)
- I/O block primitives (IOB symbols) — XC2000, XC3000, XC3000A/L, XC3100, and XC3100A only

- Configurable logic blocks (CLB symbols) — XC2000, XC3000, XC3000A/L, XC3100, and XC3100A only
- 3-state buffers (BUFT symbols) — XC3000, XC3000A/L, XC3100, XC3100, and XC4000 only
- XC3000 horizontal longline pull-up resistors (PULLUP symbols)
- XC4000 wide-edge decoders (WAND*n* and DECODE*n* symbols)
- Mapping control symbols (CLBMAP, FMAP, HMAP)

You can ignore LOC constraints in the design or in various parts of the design by using the Ignore_xnf_locs option in XNFPrep and PPR.

You can specify multiple LOC constraints for the same symbol by using a semicolon (;) to separate each LOC within the field. It specifies that the symbols be placed or prohibited from being placed in any of the locations specified. Also, you can specify an area in which to place a symbol or group of symbols.

The legal names are a function of the target LCA part type. However, to find the correct syntax for specifying a target location, you can load an empty part into the XACT Design Editor (XDE). Place the cursor on any block to display its location in the lower left corner of the screen. Do not include the pin name such as .I, .O, or .T as part of the location.

You can use the LOC constraint for logic that uses multiple CLBs, IOBs, soft macros, or other symbols. To do this, use the LOC attribute on a soft macro symbol, which passes the location information down to the logic on the lower level. However, the location restrictions are only applied to the flip-flops within the logic block or to mapping symbols or 3-state buffers in user-created macros.

Description for EPLDs

For EPLDs, use the LOC=*pinname* attribute on a PAD symbol to assign the signal to a specific pin. The PAD symbols are IPAD, OPAD, IOPAD, and UPAD.

Pin assignments are unconditional; that is, the software does not attempt to relocate a pin if it cannot achieve the specified assignment. You can apply the LOC constraint to as many PAD symbols in your design as you like. However, each pin assignment further constrains

the software as it automatically allocates logic and I/O resources to internal nodes and I/O pins with no LOC constraints.

To save all resulting pin assignments so they are preserved the next time you modify and re-integrate the design, use the PinSave command in the XDM Translate menu. This command saves the pin assignments to a *design_name*.vmf file. You can override individual pin assignments saved in the VMF file by changing or adding LOC=*pinname* attributes in the schematic.

Note: Pin assignment using the LOC attribute is not supported for bus pad symbols such as OPAD8.

Syntax for FPGAs

The syntax for specifying single LOC constraints for FPGAs is the following:

LOC=*location*

where *location* is a legal LCA location for the LCA part type.

You can specify areas of CLBs or BUFTs using the LOC constraint. Specify the upper left and lower right corners of an area in which logic is to be placed. Use a colon (:) to separate the two boundaries.

LOC=*location* **:** *location*

Conversely, you can also prohibit the placement of logic into a particular CLB or IOB by using the following syntax. Single locations or an entire area can be prohibited.

LOC<>*location*

LOC<>*location* **:** *location*

LOC= and LOC<> constraints can be used on the same symbol. If multiple LOC= constraints are placed on a single symbol or group of symbols, such as a macro, they are interpreted by the software as "ORing" each of the constraints together. Multiple LOC<> constraints are interpreted as "ANDing" the constraints together. The convention for specifying multiple LOC constraints is to separate each of them with a semicolon (;). Examples are shown in the "Examples" section, following.

Syntax for EPLDs

For EPLDs, the LOC syntax is the following:

LOC=*pinname*

where the pin name is P*nn* for PC packages; *nn* is a pin number. The pin name is *rc* (row number and column number) for PG packages. Examples are LOC=P24 and LOC=G2.

Examples

This section gives several examples of the LOC syntax for FPGAs.

Single LOC Constraints

Examples of the syntax for single LOC constraints are given in Table 4-6.

Table 4-6 Single LOC Constraint Examples

Attribute	Description
`LOC=P12`	Place I/O at location P12.
`LOC=B`	Place decode logic or I/O on the bottom edge.
`LOC=TL`	Place decode logic or I/O on the top left edge, or global buffer in the top left corner.
`LOC=AA` (XC2000 and XC3000 only)	Place logic in CLB AA.
`LOC=TBUF.AC.2` (XC2000 and XC3000 only)	Place BUFT in TBUF above and one column to the right of CLB AC.
`LOC=CLB_R3C5` (XC4000 only)	Place logic in the CLB in row 3, column 5.
`LOC=CLB_R4C5.ffx` (XC4000 only)	Place CLB flip-flop in the X flip-flop of the CLB in row 4, column 5.
`LOC=CLB_R4C5.F` (XC4000 only)	Place CLB function generator in the F generator of CLB-R4C5.

Attribute	Description
LOC=TBUF_R2C1.1 (XC4000 only)	Place BUFT in row 2, column 1, along the top.
LOC=TBUF_R*C0 (XC4000 only)	Place BUFT in any row in column 0.

Area LOC Constraints

Examples of LOC constraints used to specify area are given in Table 4-7.

Table 4-7 Area LOC Constraint Examples

Attribute	Description
LOC=AA:FF (XC2000 and XC3000 only)	Place CLB logic anywhere in the top left corner of the LCA bounded by row F and column F.
LOC=CLB_R1C1:CLB_R5C5 (XC4000 only)	Place logic in the top left corner of the LCA in a 5 x 5 area bounded by row 5 and column 5.
LOC=TBUF_R1C1:TBUF_R2C8 (XC4000 only)	Place BUFT anywhere in the area bounded by row 1, column 1 and row 2, column 8.

Prohibit LOC Constraints

Examples of the correct syntax for prohibiting locations are shown in Table 4-8.

Table 4-8 Prohibit LOC Constraint Examples

Attribute	Description
LOC<>T	Do not place I/O or decoder on the top edge.
LOC<>A* (XC2000 and XC3000 only)	Do not place logic anywhere in the top row.

Attribute	Description
LOC<>CLB_R5C*.ffy (XC4000 only)	Do not place the CLB flip-flop in the Y flip-flop of any CLB in row 5.
LOC<>CLB_R1C1:CLB_R5C5 (XC4000 only)	Do not place the logic in any CLB in the top left corner extending to row 5, column 5.
LOC<>TBUF_R*C0 (XC4000 only)	Do not place BUFT anywhere in column 0.

Multiple LOC Constraints

Examples of multiple LOC constraints are provided in Table 4-9.

Table 4-9 Multiple LOC Constraint Examples

Attribute	Description
LOC<>*A;LOC<>*D (XC2000 and XC3000 only)	Do not place flip-flop in first or fourth column of CLBs
LOC=T:LOC=L	Place I/O or decoder (XC4000) on the top or left edge.
LOC=CLB_R1C1:CLB_R5C5; LOC<>CLB_R5C5 (must be specified in one continuous line) (XC4000 only)	Place CLB logic in the top left corner of the LCA in a 5 x 5 area, but not in the CLB in row 5, column 5.

CLB Placement Examples

You can assign soft macros and flip-flops to a single CLB location, a list of CLB locations, or a rectangular block of CLB locations. You can also specify the exact function generator or flip-flop within a CLB. CLB locations are identified as CLB_R#C# for an XC4000, or *nn* for an XC2000 or XC3000, where *nn* is a two-letter designator. The upper left CLB is CLB_R1C1 or AA.

The following examples illustrate the format of CLB constraints. Enter LOC= or LOC<> and the pin or CLB location. If the target symbol represents a soft macro, the LOC constraint is applied to all appropriate symbols (flip-flops, maps) contained in that macro. If the

indicated logic does not fit into the specified blocks, an error is generated.

The following statements place logic in the designated CLB.

```
LOC=AA                    (XC2000 and XC3000)
LOC=CLB_R1C1              (XC4000)
```

The following statements prohibit the placement of logic in the designated CLB.

```
LOC<>AA                   (XC2000 and XC3000)
LOC<>CLB_R1C1             (XC4000)
```

The following statements place logic within the first column of CLBs. The asterisk (*) is a wildcard character.

```
LOC=*A                    (XC2000 and XC3000)
LOC=CLB_R*C1              (XC4000)
```

The next two statements place logic in any of the three designated CLBs. There is no significance to the order of the LOC statements.

```
LOC=AA;LOC=AB;LOC=AC                    (XC2000 and XC3000)
LOC=CLB_R1C1;LOC=CLB_R1C2;LOC=CLB_R1C3  (XC4000)
```

The following statements place logic within the rectangular block defined by the first specified CLB in the upper left corner and the second specified CLB in the lower right corner.

```
LOC=AA:HE                 (XC2000 and XC3000)
LOC=CLB_R1C1:CLB_R8C5     (XC4000)
```

The next statement places logic in the X flip-flop of CLB_R2C2. For the Y flip-flop, use the FFY tag.

```
LOC=CLB_R2C2.FFX          (XC4000)
```

IOB Placement Examples

You can assign I/O pads, buffers, and registers to an individual IOB location or to a specified die edge or half-edge. IOB locations are identified by the corresponding package pin designation or by the edge of the FPGA array.

The following examples illustrate the format of IOB constraints. Specify either LOC= or LOC<> and the pin location. If the target symbol represents a soft macro containing only I/O elements, for

example, INFF8, the LOC constraint is applied to all I/O elements contained in that macro. If the indicated I/O elements do not fit into the specified locations, an error is generated.

The following statement places the I/O element in location P13. For PGA packages, the letter-number designation is used, for example, B3.

```
LOC=P13
```

The next statement places I/O elements in IOBs along the top edge of the die. For the other three die edges, use B (bottom), L (left), or R (right).

```
LOC=T
```

The following statement places I/O elements in IOBs along the top half of the left edge of the die. The first letter in this code represents the die edge, and the second letter represents the desired half of that edge. Other legal half-edge values are LB (left bottom), RT (right top), RB (right bottom), TL (top left), TR (top right), BL (bottom left), and BR (bottom right).

```
LOC=LT
```

The next statement prohibits the placement of I/O elements on the left edge of the die.

```
LOC<>L
```

Note: The edges referred to in these constraints are die edges, which do not necessarily correspond to package edges. View the device in EditLCA to determine which pins are on which die edge.

BUFT Placement Examples

You can assign internal 3-state buffers (BUFTs) to an individual BUFT location, a list of BUFT locations, or a rectangular block of BUFT locations. In XC4000, BUFT locations are identified by the adjacent CLB. Thus, TBUF_R1C1.1 is just above CLB_R1C1, and TBUF_R1C1.2 is just below it in an XC4000 part. In XC2000 and XC3000, BUFT locations are not as straightforward. View the device in EditLCA to determine the exact BUFT names.

BUFT constraints all refer to locations with a prefix of TBUF, which is the name of the physical element on the device.

The following examples illustrate the format of BUFT LOC constraints. Specify either LOC= or LOC<> and the BUFT location.

The following statements place the BUFT in the designated location.

```
LOC=TBUF.AA.1          (XC2000 and XC3000)
LOC=TBUF_R1C1.1        (XC4000)
```

The next statements place BUFTs at any location in the first column of BUFTs. The asterisk (*) is a wildcard character.

```
LOC=TBUF.*A            (XC2000 and XC3000)
LOC=TBUF_R*C0          (XC4000)
```

The following statements place BUFTs within the rectangular block defined by the first specified BUFT in the upper left corner and the second specified BUFT in the lower right corner.

```
LOC=TBUF.AA:TBUF.BH        (XC2000 and XC3000)
LOC=TBUF_R1C1:TBUF_R2C8    (XC4000)
```

The following statements prohibit the placement of BUFTs at any location in the first row of BUFTs.

```
LOC<>TBUF.A*           (XC2000 and XC3000)
LOC<>TBUF_R1C*         (XC4000)
```

Global Buffer Placement Examples (XC4000 Only)

You can assign global buffers (BUFGP and BUFGS) to one of the four corners of the die. Specify either LOC= or LOC<> and the global buffer location. The following example illustrates the format of global buffer constraints.

```
LOC=TL
```

This statement places the global buffer in the top left corner of the die. For the other three corners, use TR (top right), BL (bottom left), and BR (bottom right).

You cannot assign placement to the GCLK or ACLK buffers in the XC2000 and XC3000 families, since there is only one of each, and their placements are fixed on the die.

Decode Logic Placement Examples (XC4000 Only)

In an XC4000 design, you can assign the decode logic to a specified die edge or half-edge. All elements of a single decode function must lie along the same edge.

The following example illustrates the format of decode constraints. Specify either LOC= or LOC<> and the decode logic symbol location. If the target symbol represents a soft macro containing only decode logic, for example, DECODE8, the LOC constraint is applied to all decode logic contained in that macro. If the indicated decode logic does not fit into the specified decoders, an error is generated.

```
LOC=L
```

This statement places the decoder logic along the left edge of the die. For the other three edges, use T (top), B (bottom), or R (right).

LOGIC_OPT

Architectures

The LOGIC_OPT attribute applies to the XC7200 and XC7300 families only.

Description

The LOGIC_OPT global attribute controls the default logic optimization for the entire design.

Syntax

The syntax for this attribute is the following:

```
LOGIC_OPT={on|off}
```

To inhibit logic optimization for the whole design, set this attribute to Off. The default is On. You can override the global setting for individual symbols using the OPT=on or OPT=off component attribute.

LOWPWR

Architectures

The LOWPWR attribute applies to the XC7300 family only.

Description

You can use the LOWPWR attribute as either a global or component attribute. When used as a component attribute, it determines the power setting of the macrocells used by an individual symbol. When used as a global attribute, it makes low power the global default power setting.

This attribute is ignored if it is assigned to a symbol that uses no macrocells, such as an inverter, AND/OR gate (when optimized), input register, and so on.

Syntax

To make low power the setting of the macrocells used by an individual symbol, use the following syntax:

```
LOWPWR={on|off}
```

To make low power the global default power setting, place the following syntax in the schematic:

```
LOWPWR=ALL
```

The default is LOWPWR=off, indicating a high-speed power setting for all macrocells used in the design unless otherwise specified.

MAP

Architectures

The MAP attribute applies to all FPGA families.

Description

The MAP attribute is placed on an FMAP, HMAP, or CLBMAP to specify whether pin swapping and the merging of other functions with the logic in the map are allowed. If pin swapping is allowed, the net connections to the pins on the CLB may differ from the

connections to the map symbol. If merging with other functions is allowed, other logic can also be placed within the CLB, if space allows.

Syntax

The syntax of the MAP attribute is the following:

```
MAP={PLC|PUC|PLO|PUO}
```

where the keywords have the following meanings:

- PLC means that the CLB pins are locked, and the CLB is closed.
- PLO means that the CLB pins are locked, and the CLB is open.
- PUC means that the CLB pins are unlocked, and the CLB is closed.
- PUO means that the CLB pins are unlocked, and the CLB is open.

"Locked" in these definitions means that the software cannot swap signals among the pins on the CLB; "unlocked" indicates that it can. "Open" means that the software can add or remove logic from the CLB; conversely, "closed" indicates that the software cannot add or remove logic from the function specified by the MAP symbol.

The default is PUC.

Example

A two-input function is mapped using an FMAP. Upon reaching the place and route stage of the design, the software determines that additional logic could be merged into the function generator containing the first function. If the MAP attribute value is PLO or PUO, the logic is merged into the function generator. If the MAP attribute value is PLC or PUC, the logic is not merged into the function generator. The software also determines that routing can be improved if the first and second pins on the function generator containing the 2-input function are swapped. If the MAP attribute is PUC or PUO, the pins are swapped. If the MAP attribute value is PLC or PLO, the pins are not swapped.

MEDFAST and MEDSLOW

Architectures

The MEDFAST and MEDSLOW attributes apply to the XC4000A family only.

Description

MEDFAST and MEDSLOW specify the slew rate of an XC4000A output driver. MEDFAST decreases output transition time and is slightly faster than MEDSLOW, possibly resulting in more noise and power consumption that an output driver specified as MEDSLOW.

The MEDFAST and MEDSLOW attributes can be attached to the I/O symbols and the special function access symbols TDI, TMS, and TCK.

Syntax

The syntax of the MEDFAST and MEDSLOW attributes is the following:

`MEDFAST` or **`MEDSLOW`**

MINIMIZE

Architectures

The MINIMIZE attribute applies to the XC7200 and XC7300 families only.

Description

The MINIMIZE global attribute determines whether or not the software minimizes the logic for the whole design. If the logic is minimized, any redundant or non-effective logic found in any user-specified equation files is eliminated through Boolean minimization.

Syntax

The syntax of the MINIMIZE attribute is the following:

```
MINIMIZE={on|off}
```

where On allows logic minimization, and Off inhibits it. The default is On.

MRINPUT

Architectures

The MRINPUT attribute applies to the XC7300 family only.

Description

The MRINPUT global attribute in an XC7354 or XC7336 design changes the master reset pin to an ordinary input pin. If this attribute is set to On, the EPLD device is initialized only on power-up.

Syntax

The syntax of the MRINPUT attribute is the following:

```
MRINPUT={on|off}
```

The On setting changes the master reset pin to an ordinary input pin.

The default is Off.

Net

Architectures

Net attributes apply to all families except where noted in the following paragraphs.

Description

Attaching attributes to nets affects the mapping, placement, and/or routing of the LCA design. Net attributes can be any of the following values:

- C Critical (all FPGA families)

  The C net attribute flags a net as critical so the software tries to route the net earlier than others. See also W, the weight net attribute.

Note: The use of the C (critical) and W (weight net) attributes is *not* recommended. In many cases, their use can degrade rather than improve routability and performance.

- F (XC7300 only)

  The F net attribute in an XC7300 device specifies that the macrocell implementing a component output should be placed in a fast function block (FFB). When placed on the output of an IBUF, the F attribute specifies that the input signal is to use the FastInput (FI) path when the signal is used in a fast function block.

  The F attribute is not valid on outputs of components that require features only present in high-density function blocks, such as PLFB9, ADD, ADSU, ACC, COMPM, LD, FDCP, FDCPE, XOR7, XOR8, and XOR9.

Note: The BUFE symbol can be assigned to FFB only when driving an OBUF, and it must allow FOE optimization.

- G G Output (XC2000 and XC2000L only on flip-flop clock pins and latch enable pins)

  Any CLB clocks driven by this net are connected to the G function output.

- H (XC7300 only)

  The H net attribute in an XC7300 device specifies that the macrocell implementing a component output should be placed in a high-density function block.

  The H attribute is not valid on outputs of a PLFFB9 or any of the input/output buffer symbols.

- I C Input (XC2000 and XC2000L only on flip-flop clock pins and latch enable pins)

  Any CLB clocks driven by this net are connected to the C input pin.

- K K Input (XC2000 and XC2000L only on flip-flop clock pins and latch enable pins)

  Any CLB clocks driven by this net are connected to the K input.

- L Longline (XC2000, XC3000, and XC3100 only)

  The APR router attempts to use a longline to route this net; a longline is useful for nets with high fan-out that need low skew.

- N Non-critical (all FPGA families)

  The N attribute flags a net as non-critical so the routing software gives this signal low priority. See also W, the weight net attribute.

Note: The use of the N (non-critical) and W (weight net) attributes is *not* recommended. In many cases, their use can degrade rather than improve routability and performance.

- P Pin-lock (XC2000 and XC3000 only on CLBMAP primitives; XC4000 only on FMAPs and HMAPs)

  The P attribute specifies that the signal should not be moved from the CLB pin to which it is assigned. It is useful for aligning CLB inputs with a specified longline.

- S Save (all FPGA families)

  The S attribute prevents the removal of unconnected signals, which is useful when using the map-then-merge method on lower-level hierarchy. If you do not have the S attribute on a net, any signal not connected to logic and/or an I/O primitive is removed.

- W Weight Net (all FPGA families)

  The W attribute indicates the routing order of the specified net by assigning it a net weight. For XC4000 and XC3000A/L (PPR) designs, legal values are 1-99, with 0 being equivalent to the N (non-critical) attribute and 100 being equivalent to the C (critical) attribute. For XC2000 and XC3000 devices (APR), a value of 0 or 1

means non-critical, 10 or higher means critical, and net weights of 2 through 9 are not graded.

Note: The use of the C (critical) or N (non-critical) and W (weight net) attributes is *not* recommended. In many cases, their use can degrade rather than improve routability and performance.

- X Explicit or External (all FPGA families)

  With this attribute, XNFMAP or PPR ensures that a net is not mapped inside the combinational logic of a CLB, which would make the net "disappear." For example, an external net between a logic gate and a flip-flop forces the software to place the combinational logic and the flip-flop in different CLBs. This mapping may make the mapping of the design less efficient, but it guarantees that the flagged net exists at a CLB output, which allows the signal to be probed in XDE.

Syntax

Methods of entering this attribute vary by user interface. Consult the appropriate user interface guide for instructions.

NODELAY

Architectures

The NODELAY attribute applies to the XC4000 and XC4000A families only.

Description

The default configuration of IOB flip-flops in XC4000 and XC4000A designs includes an input delay that results in no external hold time on the input data path. However, this delay can be removed by placing the NODELAY attribute on input flip-flops or latches, resulting in a smaller setup time but a positive hold time.

The NODELAY attribute can be attached to the I/O symbols and the special function access symbols TDI, TMS, and TCK.

Syntax

The syntax of the NODELAY attribute is the following:

```
NODELAY
```

OPT

Architectures

The OPT attribute applies to the XC7200 and XC7300 families only.

Description

The OPT attribute controls the optimization of all macrocells used by a symbol.

If you build combinational logic using low-level gates and multiplexers, the logic optimizer attempts to pack all logic bounded between device I/O pins and registers into a single macrocell.

The logic optimizer optimizes components forward into components connected to their outputs. It also moves forward any logic, whether combinational or sequential, that is buffered by a 3-state buffer. However, logic that itself contains a 3-state control is not moved forward.

The OPT=off attribute prevents any logic in a component from optimizing forward.

The OPT attribute has no effect on any symbol that contains no macrocell logic, such as an input/output buffer.

Syntax

The syntax of the OPT attribute is the following:

```
OPT={on|off}
```

OPT=on allows optimization of macrocell logic; OPT=off inhibits optimization. The default is the value of the LOGIC_OPT attribute, which is On unless otherwise specified.

PLD

Architectures

The PLD attribute applies to XC7200 and XC7300 families only.

Description

The PLD attribute is placed on a PLD symbol to specify the name of the file containing the logic equations for that PLD. Use it on custom primitive symbols and the following PLDs: PL20V8, PL22V10, PL20PIN, PL24PIN, PL48PIN, PLFB9, and PLFFB9.

All PLD components in your schematic design must be assigned the PLD attribute. Running XEMake automatically assembles all equation files named by all PLD=*filename* attributes found in the schematic. If you do not use XEMake, you must assemble each PLD file in the design using PLUSASM before you run the FITNET command.

Like PLDs, user-specified (custom) primitives are defined by PLUSASM equation files. The PLD=*filename* attribute is not required but can be applied as a convenient way to have your equation file automatically assembled when XEMake is invoked. If you omit the PLD attribute, FITNET will expect to find a bitmap file for the symbol (*symbol_name*.vmh) in your local CLIB subdirectory.

Syntax

Following is the syntax of the PLD attribute:

PLD=*filename*

Do not specify the *filename* extension. You must specify this *filename* as the first parameter of the CHIP statement inside the equation file, as described in the "PLUSASM Language Reference" section of the *XEPLD Reference Guide*. Here is an example:

```
CHIP filename PL22V10
```

PRELOAD_OPT

Architectures

The PRELOAD_OPT attribute applies to XC7200 and XC7300 families only.

Description

The PRELOAD_OPT global attribute allows the XEPLD software to change the preload values in the design to match the preload values supported by specified device resources such as fast function blocks and input registers. The XEPLD software can therefore map your design most efficiently, using the device resources most suited to the elements of your design. Unless you specify PRELOAD_OPT=off, the software is free to change the initial register states of any component, including PLD (custom) components defined in PLUSASM. Use PRELOAD=off to preserve the initial states specified in this manual for library components and in the PRLD equations in your PLUSASM file for PLD or custom components.

You can set a high or low preload for high-density function blocks. The preload value of fast function blocks depends on the use of Set or Reset. Input register preload values are fixed at 1, except for those on the XC7272, which are undefined.

Syntax

The syntax of the PRELOAD_OPT attribute is the following:

```
PRELOAD_OPT={on|off}
```

The On setting, which is the default, allows XEPLD to change the preload values; Off preserves all preload values defined in the library and specified in your PLD equation files.

REG_OPT

Architectures

The REG_OPT attribute applies to XC7200 and XC7300 families only.

Description

The REG_OPT global attribute controls input register optimization for the entire design. Input register optimization reduces the number of macrocells in a design by moving simple FD registers connected to IBUFs into a pad register, provided that the IBUF has no other fanouts. The clock by which the input register is controlled must be a FastCLK or an input that can be assigned to a FastCLK pin.

Syntax

Use the following the syntax with the REG_OPT option:

```
REG_OPT={on|off}
```

To inhibit input register optimization, set this attribute to Off. To enable this optimization, set it to On, which is the default.

RES

Architectures

The RES attribute applies to the XC4000H family only.

Description

You can specify an XC4000H output driver as operating in either resistive (RES) or capacitive, "softedge" (CAP) mode. In resistive mode, the output is faster and draws more power. Use this mode when the output is attached to purely resistive loads, or when ground bounce is not predicted to be a problem with the output. The RES attribute allows you to specify resistive mode.

Use capacitive mode when connecting an output to a capacitive mode, or when ground bounce is predicted to be a problem with the output. In capacitive mode, the pull-down transistor is slowly turned off as the output is pulled to ground, minimizing the likelihood of ground bounce.

See the section on the CAP attribute for more information.

The RES attribute can be attached to the I/O symbols and the special function access symbols TDI, TMS, and TCK.

Syntax

The syntax of the RES attribute is the following:

RES

RLOC

Architectures

The RLOC constraint applies to XC4000 and XC4000A/H families only.

Description

Relative location (RLOC) constraints group logic elements into discrete sets and allow you to define the location of any element within the set relative to other elements in the set, regardless of eventual placement in the overall design. See the "Relative Location (RLOC) Constraints" section later in this chapter for detailed information about this type of constraint.

Syntax

Use the following syntax with the RLOC constraint:

RLOC=R*row*#C*column*#[**.***extension*]

where the row and column numbers can be any positive integer, including zero.

The optional *.extension* can take all the values that are available with the current absolute LOC syntax: FFX, FFY, F, G, H, 1, and 2. The 1 and 2 values are available for BUFT primitives, and the rest are

available for primitives associated with CLBs. Only extensions for the XC4000 family designs are currently supported.

The RLOC value cannot specify a range or a list of several locations; it must specify a single location.

See the "Relative Location (RLOC) Constraints" section later in this chapter for information on the RLOC syntax.

RLOC_ORIGIN

Architectures

The RLOC_ORIGIN constraint applies to XC4000 and XC4000A/H families only.

Description

An RLOC_ORIGIN constraint fixes the members of a set at exact die locations. This constraint must specify a single location, not a range or a list of several locations. For detailed information about this constraint, refer to the "Relative Location (RLOC) Constraints" section later in this chapter.

The RLOC_ORIGIN constraint is required for a set that includes BUFT symbols.

Syntax

The syntax of the RLOC_ORIGIN constraint is the following:

RLOC_ORIGIN=R*row#***C***column#*

where the row and column numbers are positive non-zero integers.

RLOC_RANGE

Architectures

The RLOC_RANGE constraint applies to XC4000 and XC4000A/H families only.

Description

The RLOC_RANGE constraint is similar to the RLOC_ORIGIN constraint except that it limits the members of a set to a certain range on the die. The range or list of locations is meant to apply to all applicable elements with RLOCs, not just to the origin of the set.

Syntax

The RLOC_RANGE constraint has the following syntax:

RLOC_RANGE=R*row1***#C***col#***:R***row2***#C***col2#**

where the row numbers and the column numbers can be non-zero positive numbers or the wildcard (*) character. This syntax allows three kinds of range specifications, which are defined in the RLOC_RANGE section of the "Relative Location (RLOC) Constraints" section later in this chapter.

TNM

Architectures

The TNM attribute applies to XC3000A/L, XC3100A, and XC4000 families only, and only when XACT-Performance is used.

Description

The TNM attribute tags specific flip-flops, RAMs, pads, and input latches as members of a group to simplify the application of timing specifications to the group.

See the "XACT-Performance Utility" chapter of the *XACT Reference Guide* for detailed information about this attribute.

Syntax

Following is the syntax of the TNM attribute:

TNM=*identifier*

where *identifier* can be any combination of letters, numbers, or underscores.

Do not use reserved words, such as FFS, LATCHES, RAMS, or PADS for TNM identifiers.

TS*identifier*

Architectures

The TS*identifier* attribute applies to XC3000A/L, XC3100A, and XC4000 families only.

Description

TS*identifier* properties beginning with the letters "TS" are placed on the TIMESPEC symbol. The value of the TS*identifier* attribute corresponds to a specific timing specification that can then be applied to paths in the design.

See the "XACT-Performance Utility" chapter of the *XACT Reference Guide* for detailed information about this attribute.

Syntax

The syntax of the TS*identifier* attribute is the following:

TS*identifier*

where *identifier* can be any combination of letters, numbers, or underscores. It is commonly 01, 02, 03, and so forth. In Mentor, it *must* be 01, 02, 03, and so forth.

TTL

Architectures

The TTL attribute applies to the XC4000H family only.

Description

The TTL attribute configures output drivers on the XC4000H to drive to TTL-compatible levels. Similarly, it configures IOBs to have TTL-compatible input thresholds.

To configure output drive levels, attach the TTL attribute to any of the following output symbols: OBUF, OBUFT, OUTFF/OFD, OUTFFT/OFDT.

To configure input threshold levels, attach the TTL attribute to any of the following input symbols: IBUF, INFF/IFD, INLAT/ILD, INREG.

See the section on the CMOS attribute for more information.

Syntax

The syntax of the TTL attribute is the following:

```
TTL
```

UIM_OPT

Architectures

The UIM_OPT attribute applies to the XC7200 and XC7300 families only.

Description

UIM optimization extracts AND expressions and inverters out of macrocell logic functions and moves them into the UIM, which reduces the use of function block resources. The UIM_OPT global attribute turns this type of optimization on or off.

Syntax

The syntax of the UIM_OPT attribute is the following:

```
UIM_OPT={on|off}
```

where On activates UIM optimization, and Off inhibits it. The On setting is the default.

USE_RLOC

Architectures

The USE_RLOC constraint applies to the XC4000 and XC4000A/H families only.

Description

The USE_RLOC constraint turns on or off the RLOC constraint for a specific element or section of a set. For detailed information about this constraint, refer to the "Relative Location (RLOC) Constraints" section later in this chapter.

Syntax

The syntax of the USE_RLOC constraint is the following:

```
USE_RLOC={true|false}
```

where True turns on the RLOC attribute for a specific element, and False turns it off.

U_SET

Architectures

The U_SET constraint applies to the XC4000 and XC4000A/H families only.

Description

The U_SET constraint groups design elements with attached RLOC constraints that are distributed throughout the design hierarchy into a single set. The elements that are members of a U_SET can cross the

design hierarchy; that is, you can arbitrarily select objects without regard to the design hierarchy and tag them as members of a U_SET. For detailed information about this attribute, refer to the "Relative Location (RLOC) Constraints" section later in this chapter.

Syntax

The syntax of the U_SET constraint is the following:

U_SET=*name*

where *name* is the identifier of the set. This name is absolute; you specify it, and it is not prefixed by a hierarchical qualifier.

PPR Placement Constraints

This section describes the legal PPR placement constraints for each type of logic element, such as flip-flops, I/O pads, BUFTs, memories, 3-state buffers, global buffers, and edge decoders in FPGA designs. Individual logic gates, such as AND or OR gates, are mapped into CLB function generators before the constraints are read and therefore cannot be constrained. However, if gates are represented by an FMAP or HMAP symbol, you can place a placement constraint on that symbol.

This section first describes the syntax for using constraints on schematics and in a constraints (CST) file, then it gives examples showing how both kinds of syntax are used to place constraints on the various types of logic elements.

Schematic Syntax

This section describes how to place constraints on symbols on a schematic. You can use LOC, RLOC, BLKNM, and HBLKNM constraints on these symbols; these constraints are described earlier in this chapter in the "Attributes" section. Although you can prohibit individual symbols from being placed in a certain location, you cannot prohibit symbol placement in general.

To specify a single location, use the following syntax:

constraint=location

To specify a list of locations, use this syntax:

constraint=location;constraint=location;constraint=location ...;

The following syntax defines the two corners of a bounding box:

constraint=blockname:blockname

A colon is only used to separate the corners of a bounding box.

A semicolon separates locations.

The < > arrows can be substituted for the equals sign, =, to specify a "prohibit" location constraint.

Here are some examples of location constraints:

```
LOC=CLB_R1C2

LOC=P12

LOC=CLB_R5C6;LOC=CLB_R6C6

LOC=CLB_R2C2:CLB_R3C3

LOC<>CLB_R1C2

LOC<>P7
```

Constraints File Syntax

This section describes how to place constraints on instances and blocks in the constraints file. It also gives the syntax for all the statements that can be placed in the constraints file.

Instances and Blocks

Because the statements in the constraints file concern instances and blocks, these entities are defined here.

An instance is a symbol on the schematic. An instance name is the symbol name as it appears in the XNF file. Instance constraints are used for XC4000 designs.

A block is a CLB or an IOB. A block name depends on the design family used. In XC3000 and XC3000A/L designs, the name is assigned by XNFMAP using BLKNM, HBLKNM, and signal names

associated with the block. In XC4000 designs, you assign the block name with the BLKNM or HBLKNM attribute.

Place Instance Constraints

The Place Instance constraint instructs PPR to place or not place an instance in a specific location.The Place Instance and Notplace Instance constraints cannot be used for XC3000A/L or XC3100A designs, because the design is partitioned into CLBs before PPR processes it. Only the Place Block and Notplace Block statements, described in the next section, are allowed for these families, since these constraints operate on mapped blocks instead of mapped instances.

The general syntax for placing PPR constraints on an instance in the constraints file is the following:

{place|notplace} instance ***instance_name:location;***

where the keywords are the following:

- Place Instance specifies the location of an instance to be placed.
- Notplace Instance prohibits placement of an instance in the specified location.

Instance_name is the name of the instance affected by the keyword.

Location can be one of the following three types of locations:

- Single location
- List of locations separated by spaces
- Locations of two bounding box corners, which must be enclosed in square brackets and separated by spaces.

The syntax that you can use to specify locations is given in the "Statements" section later in this chapter.

Each constraint statement must end with a semicolon(;).

A colon separates the instance name from the location in constraint statements.

Following are some examples showing how instances are constrained:

```
place instance $1I2/$1I3:CLB_R5C3;
place instance $2I3/$2I5/$3I6:P12;
```

Place Block Constraints

The Place Block constraint constrains CLBs or IOBs that have been named by BLKNM or HBLKNM attributes, or by XNFMAP for XC3000, XC3000A/L, or XC3100A designs. The general syntax of this constraint is the following:

{place|notplace} block ***blockname: location;***

where the keywords are the following:

- Place Block specifies the location of a block to be placed.
- Notplace Block prohibits placement of a block in the specified location.

For CLBs, the *blockname* field must match the BLKNM attribute on the individual FMAPs, HMAPs, CLBMAPs, and/or flip-flops. For IOBs, the *blockname* must match the BLKNM attribute on individual I/O elements.

The *location* for CLBs can be a single CLB location, a range of CLBs, or a wildcard constraint. For IOBs, the *location* is a package pin, such as P12 or A6, or a die edge, such as T for top. The syntax that you can use to specify locations is given in the "Statements" section later in this chapter. Each constraint statement must end with a semicolon(;).

A colon separates the instance name from the location in constraint statements.

Here are some examples showing how CLBs are constrained:

```
place block ABC:CLB_R3C7;
place block DEF:[CLB_R1C2 CLB_R5C4];
place block GHI:CLB_R*C3;
```

The Place Block constraint differs from other PPR constraints because the logic is referenced by the LCA block name rather than the XNF symbol's instance name.

Syntactical Conventions

The following syntactical conventions are used in the CST file statements given in the "Statements" section, following:

- Lower-case words are literal.
- `[0-9]` means a range of numbers between 0 and 9, inclusive.
- `[a-z]` means a range of characters between A and Z, inclusive.
- `::=` means "can be composed of."
- `|` indicates alternatives; you must select either one or the other.
- `{}` means that you must choose one of the items enclosed in the brackets.
- `[]` enclose items that are optional. Brackets also enclose a range.
- Items in *italics* are variables for which you substitute a value.
- Items in `Courier`, or `typewriter`, font are to be entered literally.

Wildcards

You can use the wildcard (*) character in constraint statements as follows.

In an instance name, a wildcard character by itself represents every symbol of the appropriate type. For example, the following constraint applies to every BUFT in the design:

```
notplace instance *: TBUF_R1C1.1;
```

If the wildcard character is used as part of a longer instance name, the wildcard represents one or more characters at that position. However, only symbols of the appropriate type are constrained. For example, consider the following constraint:

```
notplace instance cntr/q*: CLB_R7C3;
```

This constraint would apply to a flip-flop named cntr/q7, but not to a BUFT named cntr/q7_data.

In a location, a wildcard character can be used for either the row number or column number. For example, the following constraint

prevents the flip-flop named cntr/q0 from being placed in any CLB in the third column:

```
notplace instance cntr/q0: CLB_R*C3;
```

Wildcard characters cannot be used in dot extensions; for example, CLB_R1C3.* is illegal.

Statements

Following are the statements that you can use in the CST file. It is not recommended that you use the *flag_constraint* and *weight_constraint* statements.

constraints : : = {*place_constraint* | *flag_constraint* | *weight_constraint* | *time_spec* | *time_grp*}

Place Constraints

place_constraint : : = {*place_instance* | *place_block*}

place_instance : : = {`place`|`notplace`} `instance` *instance_list*: *loc_list*;

instance_list : : = *alphanum* [*alphanum alphanum alphanum* . . .]

loc_list : : = *loc_spec* [*loc_spec loc_spec loc_spec* . . .]

loc_spec : : = {*clb_locs* | *pin_locs* | *edge_locs* | *buft_locs*}

clb_locs : : = {*clb_loc*[*bel*] | [*clb_loc clb_loc*] [*bel*]}, where [*clb_loc clb_loc*] is a range from the lowest to the highest.

bel : : = .{`f`|`g`|`ffx`|`ffy`|`1`|`2`}

clb_loc : : = `clb_r` *row_col* `c` *row_col*

row_col : : = *number*, where *number* can be any number between 0 and 99, inclusive.

pin_locs : : = See the appropriate data book for the pin package names, for example, p12, or unbonded pad names, for example, u16.

edge_locs : : = {`t`|`b`|`l`|`r`|`tl`|`tr`|`bl`|`br`|`rt`|`rb`|`lt`|`lb`}

The edge locations corresponding to these terms are given in the "I/O Constraints" section later in this chapter.

buft_locs : : = {*buft_loc* [*bel*] | [*buft_loc buft_loc*] [*bel*] }, where [*buft_loc buft_loc*] is a range from the lowest to the highest.

buft_loc : : = `tbuf_r` *row_col* `c` *row_col*

place_block : : = {`place`|`notplace`} `block` *alphanum*: *block_loc_list*;

block_loc_list : : = *block_loc_spec* [*block_loc_spec block_loc_spec block_loc_spec*. . .]

block_loc_spec : : = {*clb_loc* | [*clb_loc clb_loc*] | *pin_locs* | *edge_locs* | *buft_locs*}, where [*clb_loc clb_loc*] is a range from the lowest to the highest.

alphanum : : = Any combination or number of letters A through Z and any combination or number of numbers 0 through 9 can be used. Letters can be upper case or lower case. Underscores are acceptable. Any characters that can be used in an XNF file are also acceptable; however, *alphanum* must start with a letter.

Flag Constraints

flag_constraint : : = `flag net` {`critical`|`uncritical`} *net_list*;

net_list : : = *alphanum* [*alphanum alphanum alphanum*. . .]

Weight Constraints

weight_constraint : : = `weight net` *net_weight net_list*;

net_weight : : = [0-100]

TIMESPEC Constraints

timespec : : = `TIMESPEC=`"*timespec_line*" ;

timespec_line : : = *tsIdentifier* = *timespec_statement*

tsIdentifier : : = `TS` *tlabel*

tlabel : : = *alphatnm* [*alphatnm* . . .]

alphatnm : : = { [a-z] | [A-Z] | [0-9] | _}

timespec_statement : : = {*default_spec* | *delay_spec* | *link* | `ignore`}

default_spec : : = {*dc2s* | *dp2s* | *dc2p* | *dp2p*}

delay_spec : : = {*from_to* | *c2s* | *p2s* | *c2p* | *p2p*}

link : : = `link:`*tsIdentifier*`[:`*tsIdentifier*`:`*tsIdentifier*`...]`

dc2s : : = `dc2s:`*dmax_delay*`[:`*thi*`]`

dp2s : : = `dp2s:`*dmax_delay*

dc2p : : = `dc2p:`*dmax_delay*

dp2p : : = `dp2p:`*dmax_delay*

from_to : : = `from:`*group*`:to:`*group*`=`*max_delay*

c2s : : = `c2s:`*dmax_delay*`[:`*thi*`]`

p2s : : = `p2s:`*dmax_delay*`[:`*signame*`]`

c2p : : = `c2p:`*dmax_delay*`[:`*signame*`]`

p2p : : = `p2p:`*dmax_delay*`:`*signame*`:`*signame*

dmax_delay : : = {*requirement* | `auto` | `ignore`}

thi : : = *float_number*

group : : = {*tlabel* | *whole_class* | *pattern*}

whole_class : : = {`ffs` | `pads` | `rams` | `latches`}

max_delay : : = {*requirement* | `auto`}

pattern : : = *whole_class* (*signame* {`:`*signame*})

signame : : = {*alphanum* | `*` | `?`} [{*alphanum* | `*` | `?`}...]

requirement : : = {*float_number* [*unit*] | *reference*}

unit : : = {`ns` | `us` | `mhz` | `khz`}

reference : : = *tsIdentifier operator float_number*

operator : : = `*` | `/` In this case, the asterisk is a multiplier, not a wildcard.

float_number : : = *number* [`.`*number*], where *number* can be any number between 0 and 9, inclusive.

TIMEGRP Constraints

timegrp : : = TIMEGRP="*timegrp_line*" ;

timegrp_line : : = *tlabel=derived_group*

derived_group : : = *compound* | *difference* | *rise_fall*

compound : : = *group* [:*group* :*group* . . .]

difference : : = *compound* : except : *compound*

rise_fall : : = *edge* : *group*

edge : : = {rising | falling}

Restrictions

You should observe the following restrictions when using constraints in the CST file.

- Use only *pin_locs* and *edge_locs* to place constraints on IOB elements.
- Use only two-character *edge_locs* on BUFGSs and BUFGPs.
- Since BUFGs can only go in corners, "tl" means the top left corner. For IOBs, "tl" means the top left half-edge.
- Do not use Notplace Instance with decoders or WANDs having a DECODE attribute.
- *BUFT_locs* can only be used on BUFT elements.
- On decoders only, use full- or half-edge constraints; no *pin_locs* are allowed. You can only specify one location.

Determining Symbol Names

In a constraints file, each placement constraint acts upon one or more symbols. Every symbol in a design carries a unique name, which is defined in the input file. Use this name in a constraint statement to identify the symbol.

For each type of constraint described in the following sections, the method of determining the symbol name is explained and examples are given.

Flip-Flop Constraints

Flip-flops can be constrained to a specific CLB, a range of CLBs, a row or column of CLBs, or a specific half-CLB. Flip-flop constraints can be assigned from the schematic or through the CST file.

You cannot use Place Instance constraints on XC3000A/L flip-flops.

From the schematic, attach LOC constraints to the target flip-flop. The constraints are then passed into the XNF and XTF files and read by PPR. For more information on attaching LOC constraints, see the appropriate interface user guide.

In the CST file, a flip-flop is identified by a unique instance name. A flip-flop instance of type DFF can be found in the input file. Assume that the following lines appear in the input file:

```
sym, /top-12/fdrd, dff, init=r
sym, /top-54/fdsd, dff, init=s
```

The instance names of these two flip-flops are /top-12/fdrd and /top-54/fdsd. These names are used in the following examples showing how constraints are applied to flip-flops on the schematic and in the constraints file.

In the following examples, repeating the LOC constraint, separated by a semicolon, specifies multiple locations for an element.

Example 1:

Schematic constraint `loc=clb_r1c5`

Constraints file
```
place instance /top-12/fdrd:
clb_r1c5;
```

Place the flip-flop in the CLB in row 1, column 5. CLB R1C5 is in the upper left corner of the device.

Example 2:

Schematic constraint `loc=clb_r1c1:clb_r5c7`

Constraints file
```
place instance /top-12/fdrd:
[clb_r1c1 clb_r5c7];
```

Place the flip-flop in the rectangular area bounded by the CLB R1C1 in the upper left corner and CLB R5C7 in the lower right.

Example 3:

Schematic constraint `loc=clb_r*c3`

Constraints file
```
place instance /top-12/fdrd
/top-54/fdsd: clb_r*c3;
```

Place the flip-flops in any row of column 3. The wildcard (*) character can be used in place of either the row or column number to specify an entire row or column of CLBs.

From the schematic, the same LOC constraint is attached to both flip-flops.

Example 4:

Schematic constraint `loc=clb_r2c4;loc=clb_r7c9`

Constraints file
```
place instance /top-54/fdsd:
clb_r2c4 clb_r7c9;
```

Place the flip-flop in either CLB R2C4 or CLB R7C9.

Example 5:

Schematic constraint `loc=clb_r3c5.ffy`

Constraints file
```
place instance /top-12/fdrd:
clb_r3c5.ffy;
```

Place the flip-flop in CLB R3C5 and assign the flip-flop output to the XQ pin. Use the FFY tag to indicate the YQ pin of the CLB. If the FFX or FFY tags are specified, the wildcard (*) character cannot be used for the row or column numbers.

Example 6:

Schematic constraint `loc<>clb_r5c*`

Constraints file
```
notplace instance /top-12/fdrd:
clb_r5c*;
```

Do not place the flip-flop in any column of row 5. The wildcard (*) character can be used in place of either the row or column number to specify an entire row or column of CLBs.

ROM and RAM Constraints

A ROM or RAM can be constrained to a specific CLB, a range of CLBs, or a row or column of CLBs. Memory constraints can be assigned from the schematic or through the CST file.

From the schematic, attach the LOC constraints to the memory symbol. The constraints are then passed into the XNF and XTF files and read by PPR. For more information on attaching LOC constraints, see the appropriate interface user guide.

In the constraints file, a memory is identified by a unique instance name. For a memory not created by MemGen, one or more memory instances of type ROM or RAM can be found in the input file. All memory macros larger than 16 x 1 or 32 x 1 are broken down into these basic elements in the XNF file. Examples of non-MemGen memory instances in the XNF file are shown following:

```
sym, /top-7/rq, rom, init=05a3
sym, /top-11-ram64x8/ram-3, ram
```

The instance name of the ROM primitive is /top-7/rq. The instance name of the RAM primitive, which is a piece of a RAM64X8 macro, is /top-11-ram64x8/ram-3. These names are used in the following examples.

A MemGen-created memory is represented by a hierarchical symbol in the XNF file, as shown in this example:

```
sym, /top-17/bigram, bigram, file=bigram
```

The instance name of the MemGen module is /top-17/bigram.

Example 1:

Schematic constraint `loc=clb_r1c5`

Constraints file
```
place instance /top-7/rq:
clb_r1c5;
```

Place the memory in the CLB in row 1, column 5. CLB R1C5 is in the upper left corner of the device. A single-CLB constraint such as this can only be applied to a 16 x 1 or 32 x 1 memory.

Example 2:

Schematic constraint `loc=clb_r2c4;loc=clb_r7c9`

Constraints file
```
place instance /top-7/rq:
clb_r2c4 clb_r7c9
```

Place the memory in either CLB R2C4 or CLB R7C9.

Example 3:

Schematic constraint `loc=clb_r1c1:clb_r5c7`

Constraints file
```
place instance /top-17/bigram/*:
[clb_r1c1 clb_r5c7];
```

Place the MemGen module in the rectangular area bounded by the CLB R1C1 in the upper left corner and CLB R5C7 in the lower right.

From the schematic, attach the LOC constraint to the MemGen symbol.

In the CST file, the `/*` is appended to the end of the MemGen symbol instance name found in the XNF file. The wildcard (*) character is used here to specify all instances that begin with the /top-17/ bigram/ prefix.

Example 4:

Schematic constraint `loc<>clb_r5c*`

Constraints file
```
notplace instance /top11ram64x8*:
clb_r5c*;
```

Do not place the memory in any column of row 5. The wildcard (*) character can be used in place of either the row or column number in the CLB name to specify an entire row or column of CLBs.

From the schematic, the LOC constraint is simply attached to the RAM64X8 macro symbol and is passed through to each individual RAM in the XNF file.

In the CST file, the wildcard (*) character specifies all instances that begin with the /top-11-/ram64x8/ prefix.

Mapping Constraints

FMAP and HMAP Constraints

The FMAP and HMAP symbols control mapping in an XC4000 design. They are similar to the XC2000/XC3000 CLBMAP symbol.

FMAP and HMAP control the mapping of logic into function generators. These symbols do not define logic on the schematic; instead, they specify how portions of logic shown elsewhere on the schematic should be mapped into a function generator.

The FMAP symbol defines mapping into a four-input (F) function generator. PPR assigns this function to an F or G function generator, so you are not required to specify whether it belongs in F or G.

The HMAP symbol defines mapping into a three-input (H) function generator. If the HMAP has two FMAP outputs and, optionally, one normal (non-FMAP) signal as its inputs, PPR normally places all the logic related to these symbols into one CLB.

An example of how to use these symbols in your schematic appears in Figure 4-3 and Figure 4-4.

For the FMAP symbol, as with the CLBMAP primitive, MAP={PUC, PUO, PLC or PLO} is supported, as well as the LOC constraint.

For the HMAP symbol, only MAP=PUC is supported.

You can ignore FMAP and HMAP symbols in the input file by using the PPR Ignore_maps option described in the "PPR" chapter of the *XACT Reference Guide.*

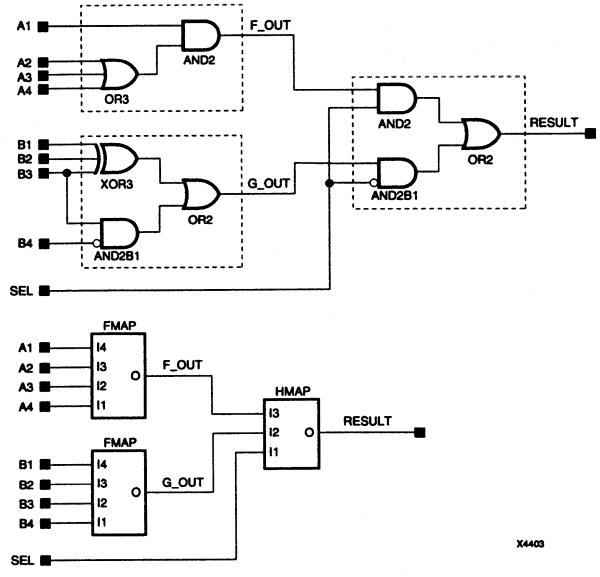


Figure 4-3 FMAP and HMAP Schematics

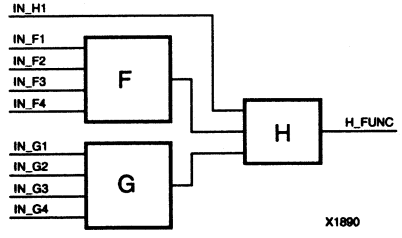


Figure 4-4 PPR Implementation of FMAP and HMAP

Example 1:

Schematic constraint `loc=clb_r7c3`

Constraints file `place instance $1I323: clb_r7c3;`

Place the FMAP or HMAP symbol in the CLB at row 7, column 3.

Example 2:

Schematic constraint `loc=clb_r2c4;loc=clb_r3c4`

Constraints file
```
place instance top/dec0011:
clb_r2c4 clb_r3c4;
```

Place the FMAP or HMAP symbol in either the CLB at row 2, column 4 or the CLB at row 3, column 4.

Example 3:

Schematic constraint `loc=clb_r5c5:clb_r10c8`

Constraints file
```
place instance $3I27: [clb_r5c5
clb_r10c8;
```

Place the FMAP or HMAP symbol in the area bounded by CLB R5C5 in the upper left corner and CLB R10C8 in the lower right.

Example 4:

Schematic constraint `loc=clb_r10c11.f`

Constraints file
```
place instance top/done:
clb_r1011.f;
```

Place the FMAP in the F function generator of CLB R10C11. The .G extension specifies the G function generator. An HMAP can only go into the H function generator, so there is no need to specify this placement explicitly.

CLBMAP Constraints

With the CLBMAP symbol, you can specify logic mapping at the schematic level for all XC3000 designs. It is used in conjunction with standard logic elements, such as gates and flip-flops. It implicitly specifies the configuration of a CLB by defining the signals on its pins. Use the CLBMAP symbol to control mapping when the default mapping is not acceptable.

Enter the CLBMAP symbol on the schematic and assign signals to its pins. XNFMAP processes this symbol and maps the appropriate logic, as defined by the input and output signals, into one CLB. The easiest way to define a CLBMAP is to connect a labeled wire segment

to each pin, which connects that pin to the net carrying the same label.

If a CLBMAP specifies an illegal CLB configuration, XNFMAP ignores the CLBMAP and issues a warning explaining why the CLBMAP is illegal.

A CLBMAP can be either closed or open. A closed CLBMAP must specify both the input and output signals for that CLB. XNFMAP maps a closed CLBMAP exactly as specified, unless the indicated configuration is illegal. XNFMAP does not add any logic to a CLB specified with a closed CLBMAP.

An open CLBMAP specifies only the output signals for the CLB. XNFMAP assigns those signals to the CLB output pins and maps the source logic into the CLB as appropriate. Use an open CLBMAP to specify the function of a CLB without specifying the exact configuration.

Specify whether a CLBMAP is either open or closed by attaching the appropriate MAP attribute to the symbol. See Table 4-10 for the exact conventions.

The pins on a CLBMAP can be either locked or unlocked. Specify whether a CLBMAP has locked or unlocked pins by attaching the appropriate MAP attribute to the symbol. See Table 4-10 for the exact conventions. With an open CLBMAP, only the output pins are locked.

If a CLBMAP is indicated as having unlocked pins, you can lock individual CLBMAP pins by attaching a P (pin-lock) attribute to the corresponding net. On an open CLBMAP, you can assign P attributes only to output pins.

Table 4-10 Map Attributes for CLBMAP Symbols

	Closed CLB	Open CLB
Pins locked	MAP=PLC	MAP=PLO
Pins unlocked	MAP=PUC	MAP=PUO

Example 1:

Schematic constraint `loc=CB`

Constraints file `place block top/cntq7: CB;`

Place the CLBMAP in the CB CLB.

Example 2:

Schematic constraint `loc=AA:EE`

Constraints file `place block reg/bit7: [AA:EE];`

Place the CLBMAP in the area bounded by CLB AA in the upper left corner and CLB EE in the lower right.

CLB Constraints

You can prohibit PPR from using a specific CLB, a range of CLBs, or a row or column of CLBs. Such prohibit constraints can be assigned only through the constraints file. CLBs are prohibited by specifying a Notplace Instance constraint with only a wildcard (*) character as the instance name, as shown in the following examples.

Example 1:

Schematic constraint None

Constraints file `notplace instance *: clb_r1c5;`

Do not place any logic in the CLB in row 1, column 5. CLB R1C5 is in the upper left corner of the device.

Example 2:

Schematic constraint None

Constraints file
```
notplace instance *: [clb_r1c1
clb_r5c7];
```

Do not place any logic in the rectangular area bounded by the CLB R1C1 in the upper left corner and CLB R5C7 in the lower right.

Example 3:

Schematic constraint None

Constraints file
```
notplace instance *: clb_r*c3;
```

Do not place any logic in any row of column 3. The wildcard (*) character can be used in place of either the row or column number to specify an entire row or column of CLBs.

Example 4:

Schematic constraint None

Constraints file
```
notplace instance *: clb_r2c4
clb_r7c9;
```

Do not place any logic in either CLB R2C4 or CLB R7C9.

I/O Constraints

You can constrain I/Os to one edge of the die, half an edge of the die, or a specific IOB. I/O constraints can be assigned from the schematic or through the CST file.

From the schematic, attach LOC constraints to the target PAD symbol. The constraints are then passed into the XNF and XTF files and read by PPR. For more information on attaching LOC constraints, see the appropriate interface user guide.

A pad can be found in the XNF file as an EXT record. Assume the following lines appear in the XNF file.

```
ext, /top-102/data0, i,, blknm=data0
ext, /top-117/q13, o,, blknm=out13
```

For a CST file constraint, the instance names of these I/Os are /top-102/data0_pad and /top-117/q13_pad.

Example 1:

This example uses a pin number to lock to one pin.

Schematic constraint `loc=p17`

Constraints file
```
place instance /top-102/
data0_pad: p17;
```

Place the I/O in the IOB at pin 17. For pin grid arrays, a pin name such as B3 or T1 is used.

Example 2:

The following example uses a letter to lock to a side of the die.

Schematic constraint `loc=t`

Constraints file
```
place instance /top-117/
q13_pad: t;
```

Place the I/O in any IOB on the top edge of the die. Table 4-11 shows the legal edge designations.

Table 4-11 Legal Edge Designations for IOBs

Edge Code	Edge Location
b	Bottom edge
l	Left edge
r	Right edge
t	Top edge

Note: When assigning global location constraints specifically to IOBs and edge decoders, refer to the die locations, *not* the package locations. The package edges do not necessarily correspond to the die edges. The die locations are rotated with respect to the package locations. The only way to see where these edges are is to load the design into EditLCA. See the EditLCA section in the "XACT Design Editor" chapter of the *XACT Reference Guide* for additional information.

Example 3:

This example uses multiple locations.

Schematic constraint `loc=t;loc=b`

Constraints file
```
place instance /top-117/
q13_pad: t b;
```

Place the I/O in any IOB on the top or bottom edges of the die.

Example 4:

A two-letter combination is used to lock to a half-edge in the following example.

Schematic constraint `loc=tl`

Constraints file
```
place instance /top-102/
data0_pad: tl;
```

Place the I/O in any IOB on the left half of the top edge of the die. Table 4-12 shows the legal half-edge designations.

Table 4-12 Legal Half-Edge Designations for IOBs

Half-Edge Code	Edge Location
TL	Left half of top edge
TR	Right half of top edge
BL	Left half of bottom edge
BR	Right half of bottom edge
LT	Top half of left edge
LB	Bottom half of left edge
RT	Top half of right edge
RB	Bottom half of right edge

Example 5:

This example constrains I/Os to the right edge of the IOB.

Schematic constraint `loc=r`

Constraints file
```
place instance /top-117/q*_pad:
r;
```

Place the I/Os in the IOBs on the right edge of the die.

From the schematic, the LOC constraint is attached to all target PAD symbols.

In the CST file, the wildcard (*) character specifies all instances that begin with /top-117/q and end with _pad. It identifies the external signals /top-117/q0_pad, /top-117/q1_pad, and so on.

IOB Constraints

You can prohibit PPR from using a specific IOB. This step might be taken to keep user I/O signals away from semi-dedicated configuration pins. Such prohibit constraints can be assigned only through the CST file.

IOBs are prohibited by specifying a Notplace Instance constraint with only a wildcard (*) character as the instance name, as shown in the following example.

Schematic constraint	None
Constraints file	`notplace instance *: p36 p37 p41;`

Do not place user I/Os in the IOBs at pins 36, 37, or 41. For pin grid arrays, pin names such as D14, C16, or H15 are used.

BUFT Constraints

You can constraint 3-state buffers to a specific BUFT, a range of BUFTs, or a row or column of BUFTs. BUFT constraints can be assigned from the schematic or through the CST file. BUFT constraints all refer to locations with a prefix of TBUF, which is the name of the physical element on the device.

From the schematic, LOC constraints are attached to the target BUFT. The constraints are then passed into the XNF and XTF files and read by PPR. For more information on attaching LOC constraints, see the appropriate user interface guide.

In a constraints file, a BUFT is identified by a unique instance name. A BUFT instance can found in the XNF file. Assume the following lines appear in the XNF file. A BUFT symbol can be translated to the equivalent TBUF type.

```
sym, /top-72/rd0, TBUF
sym, /top-79/ed7, TBUF
```

The instance names of these two BUFTs are /top-72/rd0 and /top-79/ed7. These names are used in the following examples.

Example 1:

This example specifies BUFTs adjacent to a specific CLB.

Schematic constraint `loc=TBUF_r1c5`

Constraints file
```
place instance /top-72/rd0:
TBUF_r1c5;
```

Place the BUFT adjacent to CLB R1C5. You can use either the longline above the row of CLBs or the longline below.

Example 2:

The following example places a specific BUFT.

Schematic constraint `loc=TBUF_r1c5.1`

Constraints file
```
place instance /top-72/rd0:
TBUF_r1c5.1;
```

Place the BUFT adjacent to CLB R1C5. The .1 tag specifies the longline above the row of CLBs. The .2 tag specifies the longline below it.

BUFTs that drive the same signal must carry consistent constraints. If you specify the longline for one BUFT constraint, you must specify it for all constraints on that line of BUFTs. However, not all BUFTs on a line need be constrained. Constraining one BUFT to a specific longline forces the remaining BUFTs onto that line.

Example 3:

The next example specifies a column of BUFTs.

Schematic constraint `loc=TBUF_r*c3`

Constraints file
```
place instance /top-72/rd0
top-79/ed7:TBUF_r*c3;
```

Place BUFTs in column 3 on any row. This constraint might be used to align BUFTs with a common enable signal. You can use the wildcard (*) character in place of either the row or column number to specify an entire row or column of BUFTs.

From the schematic, the same LOC constraint is attached to both BUFTs.

Example 4:

This example specifies a row of BUFTs.

Schematic constraint `loc=TBUF_r7c*`

Constraints file
```
place instance /top-79/ed7:
TBUF_r7c*;
```

Place the BUFT on either the top or bottom horizontal longline in row 7, in any column. You can use the wildcard (*) character in place of either the row or column number to specify an entire row or column of BUFTs.

Edge Decoder Constraints

Edge decoders can only be constrained to a single edge of the die; they cannot be split across two edges of the die. If you use decoder constraints, you must assign all decode inputs for a given function to the same edge. From the schematic, attach LOC constraints to the decode logic — either a DECODE macro or a WAND gate with the DECODE attribute. The constraints are then passed into the XNF and XTF files and read by PPR. For more information on attaching LOC constraints, see the appropriate interface user guide.

Here is an example:

Schematic constraint `loc=T`

Constraints file `place instance dec1/$1I1:T;`

Place the decoder along the top edge of the die. Table 4-13 shows the legal edge designations.

Table 4-13 Legal Edge Designations for Edge Decoders

Edge Code	Edge Location
T	Top edge
B	Bottom edge
L	Left edge
R	Right edge
TL	Left half of top edge

Edge Code	Edge Location
TR	Right half of top edge
BL	Left half of bottom edge
BR	Right half of bottom edge
LT	Top half of left edge
LB	Bottom half of left edge
RT	Top half of right edge
RB	Bottom half of right edge

To constrain decoders to precise positions within a side, constrain the associated pads. However, because PPR determines decoder edges before processing pad constraints, it is not enough to constrain the pads alone. To constrain decoders to a specific die side, use the following rule. For every output net that you want to constrain, specify the side for at least one of its input decoders (WAND gates), using one of the following.

```
LOC=L    LOC=T
LOC=R    LOC=B
```

Global Buffer Constraints

You can constrain a global buffer — BUFGP or BUFGS — to a corner of the die. From the schematic, attach LOC constraints to the global buffer symbols. The constraints are then passed into the XNF and XTF files and read by PPR. For more information on attaching LOC constraints, see the appropriate interface user guide.

Here is an example:

Schematic constraint `loc=TL`

Constraints file `place instance buf1:TL;`

Place the global buffer in the top left corner of the die. Table 4-14 shows the legal corner designations.

Table 4-14 Legal Corner Designations for Global Buffers

Corner Code	Corner Location
TL	Top left corner
TR	Top right corner
BL	Bottom left corner
BR	Bottom right corner

If a global buffer is sourced by an external signal, the dedicated IOB for that buffer must not be used by any other signal. For example, if a BUFGP is constrained to TL, the PGCK1 pin must be used to source it, and no other I/O can be assigned to that pin.

Relative Location (RLOC) Constraints

This section describes the relative location (RLOC) constraint, RLOC sets, and RLOC set constraints and modifiers.

Description

Relative location constraints group logic elements into discrete sets. You can define the location of any element within the set relative to other elements in the set, regardless of eventual placement in the overall design. For example, if RLOC constraints are applied to a group of eight flip-flops organized in a column, PPR maintains the columnar order and moves the entire group of flip-flops as a single unit. In contrast, absolute location (LOC) constraints constrain design elements to specific locations on the FPGA die with no relation to other design elements.

RLOC constraints allow you to place logic blocks relative to each other to increase speed, use die resources efficiently, and take advantage of the special carry logic built into the control logic blocks (CLBs) of the XC4000 devices. They provide an order and structure to related design elements without requiring you to specify their absolute placement on the FPGA die. They allow you to replace any existing hard macro with an equivalent that can be directly simulated.

The relationally placed macro (RPM) library, which replaces the hard macro library, uses RLOC constraints to define the order and

structure of the underlying design primitives. The RPM library offers the functionality and precision of the hard macro library with added flexibility. You can optimize RPMs and merge other logic within them. Because these macros are built upon standard schematic parts, they do not have to be translated before simulation.

In the Unified Libraries, you can use RLOC constraints with BUFT- and CLB-related primitives, that is, DFF, HMAP, FMAP, and CY4 primitives. You can also use them on non-primitive macro symbols. There are some restrictions on the use of RLOC constraints on BUFT symbols. See the section on the RLOC_ORIGIN attribute later in this chapter. However, you cannot use RLOC constraints with decoders, clocks, or I/O primitives. LOC constraints, on the other hand, can be used on all primitives: BUFTs, CLBs, IOBs, decoders, and clocks.

The libraries created before the release of the Unified Libraries do not include RLOC constraints on the primitive symbols below the macro symbols. To add RLOC constraints to the underlying macro primitives, make a copy of the library in your local directory and add the RLOC=R0C0 constraint to the underlying primitives. You can also attach RLOC constraints directly to non-macro primitives as you can for the Unified Libraries.

The following symbols (primitives) accept RLOCs:

FDCE
FDPE
FMAP
HMAP
RAM16X1
RAM32X1
ROM16X1
ROM32X1
BUFT

Syntax

The syntax of the RLOC constraint is the following:

RLOC = R*row*#C*column*#[.*extension*]

where the optional *.extension* can take all the values that are available with the current absolute LOC syntax: FFX, FFY, F, G, H, 1, and 2. The 1 and 2 values are available for BUFT primitives, and the rest are

available for primitives associated with CLBs. Only extensions for the XC4000 family designs are currently supported.

The row and column numbers can be any positive integer, including zero. Absolute die locations, in contrast, cannot have zero as a row or column number. Because row and column numbers in RLOC constraints define only the order and relationship between design elements and not their absolute die locations, their numbering can include zero. Even though you can use any positive integer in numbering rows and columns for RLOC constraints, it is recommended that you use small integers for clarity and ease of use.

It is not the absolute values of the row and column numbers that is important in RLOC specifications but their relative values or differences. For example, if design element A has an RLOC=R3C4 constraint and design element B has an RLOC=R6C7 constraint, the absolute values of the row numbers (3 and 6) are not important in themselves. However, the difference between them is important; in this case, 3 (6 -3) specifies that the location of design element B is three rows away from the location of design element A. To capture this information, a normalization process is used at some point in the design implementation. In the example just given, normalization would reduce the RLOC on design element A to R0C0, and the RLOC on design element B to R3C3.

In Xilinx programs, rows are numbered in increasing order from top to bottom, and columns are numbered in increasing order from left to right. RLOC constraints follow this numbering convention.

Figure 4-5a demonstrates the use of RLOC constraints. Four flip-flop primitives named A, B, C, and D are assigned RLOC constraints as shown. These RLOC constraints require each flip-flop to be placed in a different CLB in the same column and stacked in the order shown: A above B, C, and D. Within a CLB, however, they can be placed either in the FFX or FFY position.

If you wish to place more than one of these flip-flop primitives per CLB, you can specify the RLOCs as shown in Figure 4-5b. The arrangement in Figure 4-5b requires that A and B be placed in a single CLB and that C and D be placed in another CLB immediately below the AB CLB. However, within a CLB, the flip-flops can be placed in either of the two flip-flop positions, FFX or FFY.

To control the ordering of these flip-flop primitives specifically, you can use the extension field, as shown in Figure 4-5c. In this figure, the same four flip-flops are constrained to use specific resources in the CLBs. This specification always ensures that these elements are arranged exactly as shown: A must be placed in the FFX spot, B in the same CLB at the FFY spot, and so on.

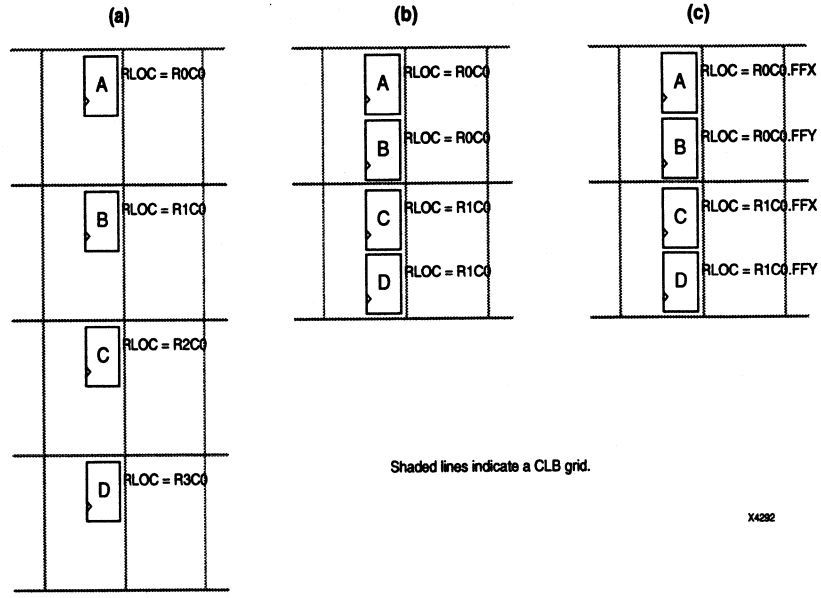


Figure 4-5 Different RLOC Specifications for Four Flip-flop Primitives

RLOC Sets

As noted previously, RLOC constraints give order and structure to related design elements. This section describes RLOC sets, which are groups of related design elements to which RLOC constraints have been applied. For example, the four flip-flops in Figure 4-5 are related by RLOC constraints and form a set. Elements in a set are related by RLOC constraints to other elements in the same set. Each member of a set must have an RLOC constraint, which relates it to other elements in the same set. You can create multiple sets, but a design element can belong to one set only.

Sets can be defined in several ways: explicitly through the use of a set parameter or implicitly through the structure of the design hierarchy.

There are four distinct types of rules associated with each set:

- Definition rules define the requirements for membership in a set.
- Linkage rules specify how elements can be linked to other elements to form a single set.
- Modification rules dictate how to specify parameters that modify RLOC values of all the members of the set.
- Naming rules specify the nomenclature of sets.

These rules are discussed in the sections that follow.

The following sections discuss three different set constraints: U_SET, H_SET, and HU_SET. Elements must be tagged with both the RLOC constraint and one of these set constraints to belong to a set.

U_SET

U_SET constraints enable you to group into a single set design elements with attached RLOC constraints that are distributed throughout the design hierarchy. The letter U in the name U_SET indicates that the set is user-defined. U_SET constraints allow you to group elements, even though they are not directly related by the design hierarchy. By attaching a U_SET constraint to design elements, you can explicitly define the members of a set. The design elements tagged with a U_SET constraint can exist anywhere in the design hierarchy; they can be primitive or non-primitive symbols. When attached to non-primitive symbols, the U_SET constraint propagates to all the primitive symbols with RLOC constraints that are below it in the hierarchy.

The syntax of the U_SET constraint is the following:

U_SET=*name*

where *name* is the user-specified identifier of the set. All design elements with RLOC constraints tagged with the same U_SET constraint name belong to the same set. Names therefore must be unique among all the sets in the design.

H_SET

In contrast to the U_SET constraint, which you explicitly define by tagging design elements, the H_SET (hierarchy set) is defined implicitly through the design hierarchy. The combination of the design hierarchy and the presence of RLOC constraints on elements defines a hierarchical set, or H_SET set. You do not use an HSET constraint to tag the design elements to indicate their set membership. The set is defined automatically by the design hierarchy. All design elements with RLOC constraints at a single node of the design hierarchy are considered to be in the same H_SET set unless they are tagged with another type of set constraint such as RLOC_ORIGIN or RLOC_RANGE. These constraints are discussed later in this chapter. If you explicitly tag any element with an RLOC_ORIGIN, RLOC_RANGE, U_SET, or HU_SET constraint, it is removed from an H_SET set. Most designs contain only H_SET constraints, since they are the underlying mechanism for relationally placed macros.

The design-flattening program, XNFMerge, recognizes the implicit H_SET set, derives its name, or identifier, attaches the H_SET constraint to the correct members of the set, and writes them to the output file.

The syntax of the H_SET constraint as generated by XNFMerge follows:

H_SET=*name*

Name is the identifier of the set and is unique among all the sets in the design. The base name for any H_SET is "hset," to which XNFMerge adds a hierarchy path prefix to obtain unique names for different H_SET sets in the XNFMerge output file.

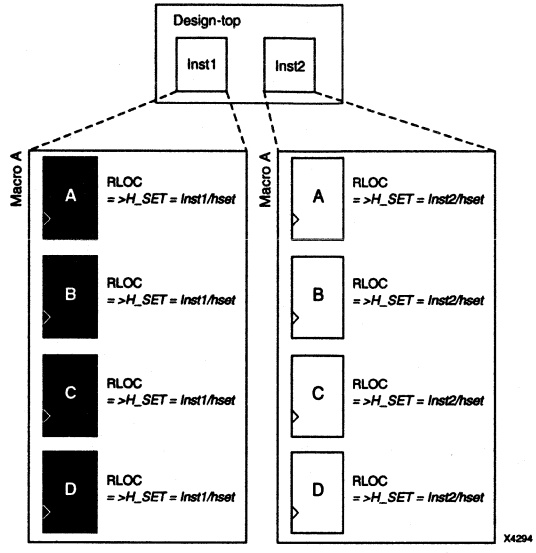


Figure 4-6 Macro A Instantiated Twice

Note: In Figure 4-6 and the other figures shown in this section, the italicized text prefixed by => is added by XNFMerge during the design flattening process. You add all other text.

Figure 4-6 demonstrates a typical use of the implicit H_SET (hierarchy set). The figure shows only the first "RLOC" portion of the constraint. In a real design, the RLOC constraint must be specified completely with RLOC=R*row*#C*col*#. In this example, macro A is originally designed with RLOC constraints on four flip-flops: A, B, C, and D. The macro is then instantiated twice in the design: Inst1 and Inst2. When the design is flattened, two different H_SET sets are recognized because two distinct levels of hierarchy contain elements with RLOC constraints. XNFMerge creates and attaches the appropriate H_SET constraint to the set members: H_SET=Inst1/hset for the macro instantiated in Inst1, and H_SET=Inst2/hset for the macro instantiated in Inst2. The design implementation programs place each of the two sets individually as a unit with relative ordering within each set specified by the RLOC constraints. However, the two sets are regarded as completely independent of each other.

The name of the H_SET set is derived from the symbol or node in the hierarchy that includes all the RLOC elements. In Figure 4-6, Inst1 is the node (instantiating macro) that includes the four flip-flop elements with RLOCs shown on the left of Figure 4-6. Therefore, the name of this H_SET set is the hierarchically qualified name of "Inst1" followed by "hset." The Inst1 symbol is considered the "start" of the H_SET, which gives a convenient handle to the entire H_SET and attaches constraints that modify the entire H_SET. Constraints that modify sets are discussed later in this chapter.

Figure 4-6 demonstrates the simplest use of a set that is defined and confined to a single level of hierarchy. Through linkage and modification, you can also create an H_SET set that is linked through two or more levels of hierarchy. Linkage allows you to link elements through the hierarchy into a single set. On the other hand, modification allows you to modify RLOC values of the members of a set through the hierarchy.

Set Linkage

The example in Figure 4-7 explains and illustrates the process of linking together elements through the design hierarchy. Again, the complete RLOC specification, RLOC=R*row*#C*col*#, is required for a real design.

Note: In this and other illustrations in this section, the sets are shaded differently to distinguish one set from another.

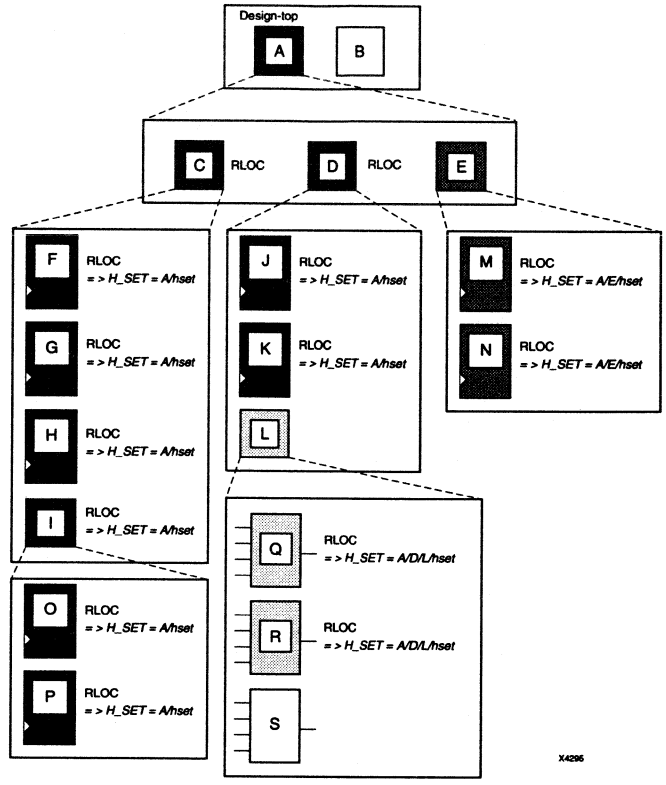


Figure 4-7 Three H_SET Sets

As noted previously, all design elements with RLOC constraints at a single node of the design hierarchy are considered to be in the same H_SET set unless they are assigned another type of set constraint, an RLOC_ORIGIN constraint, or an RLOC_RANGE constraint. In Figure 4-7, RLOC constraints have been added on primitives and non-primitives C, D, F, G, H, I, J, K, M, N, O, P, Q, and R. No RLOC constraints were placed on B, E, L, or S. Macros C and D have an RLOC constraint at node A, so all the primitives below C and D that have RLOCs are members of a single H_SET set. Furthermore, the name of this H_SET set is "A/hset" because it is at node A that the set starts. The start of an H_SET set is the lowest common ancestor of all the RLOC-tagged constraints that constitute the elements of that

H_SET set. Because element E does not have an RLOC constraint, it is not linked to the A/hset set. The RLOC-tagged elements M and N, which lie below element E, are therefore in their own H_SET set. The start of that H_SET set is A/E, giving it the name "A/E/hset."

Similarly, the Q and R primitives are in their own H_SET set because they are not linked through element L to any other design elements. The lowest common ancestor for their H_SET set is L, which gives it the name "A/D/L/hset." After the flattening, XNFMerge attaches H_SET=A/hset to the F, G, H, O, P, J, and K primitives; H_SET=A/D/L/hset to the Q and R primitives; and H_SET=A/E/hset to the M and N primitives.

Consider a situation in which a set is created at the top of the design. In Figure 4-7, there would be no lowest common ancestor if macro A also had an RLOC constraint, since A is at the top of the design and has no ancestor. In this case, the base name "hset" would have no hierarchically qualified prefix, and the name of the H_SET set would simply be "hset."

Set Modification

As noted earlier, the RLOC constraint assigns a primitive an RLOC value (the row and column numbers with the optional extensions), specifies its membership in a set, and links together elements at different levels of the hierarchy. In Figure 4-7, the RLOC constraint on macros C and D links together all the objects with RLOC constraints below them. An RLOC constraint is also used to modify the RLOC values of constraints below it in the hierarchy. In other words, RLOC values of elements affect the RLOC values of all other member elements of the same H_SET set that lie below the given element in the design hierarchy.

When the design is flattened, the row and column numbers of an RLOC constraint on an element are added to the row and column numbers of the RLOC constraints of the set members below it in the hierarchy. This feature gives you the ability to modify existing RLOC values in submodules and macros without changing the previously assigned RLOC values on the primitive symbols. This modification process also applies to the optional extension field. However, when using extensions for modifications, you must ensure that inconsistent extensions are not attached to the RLOC value of a design element that may conflict with RLOC extensions placed on underlying

elements. For example, if an element has an RLOC constraint with the FFX extension, all the underlying elements with RLOC constraints must either have the same extension, in this case FFX, or no extension at all; any underlying element with an RLOC constraint and an extension different from FFX, such as FFY or F, is flagged as an error. After resolving all the RLOC constraints, extensions that are not valid on primitives are removed from those primitives. For example, if XNFMerge generates an FFX extension to be applied on a primitive after propagating the RLOC constraints, it applies the extension if and only if the primitive is a flip-flop. If the primitive is an element other than a flip-flop, the extension is ignored. Only the extension is ignored in this case, not the entire RLOC constraint.

Figure 4-8 illustrates the process of adding RLOC values down the hierarchy. The row and column values between the parentheses show the addition function performed by XNFMerge. The italicized text prefixed by => is added by XNFMerge during the design flattening process and replaces the original RLOC constraint that you added.

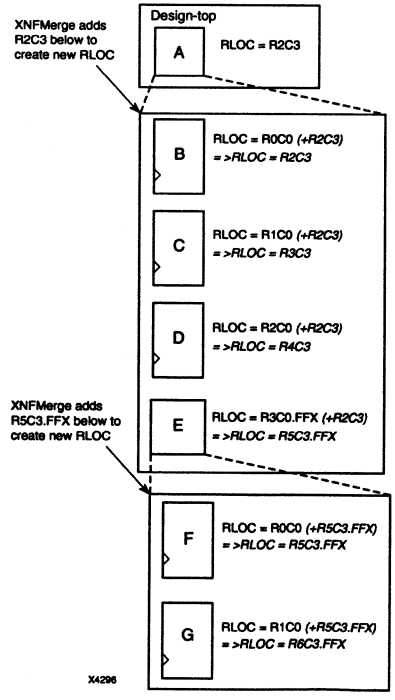


Figure 4-8 Adding RLOC Values Down the Hierarchy

The ability to modify RLOC values down the hierarchy is particularly valuable when instantiating the same macro more than once. Typically, macros are designed with RLOC constraints that are modified when the macro is instantiated. Figure 4-9 is a variation of the sample design in Figure 4-6. The RLOC constraint on Inst1 and Inst2 now link all the objects in one H_SET set. Because the RLOC=R0C0 modifier on the Inst1 macro does not affect the objects below it, XNFMerge only adds the H_SET tag to the objects and leaves the RLOC values as they are. However, the RLOC=R0R1 modifier on the Inst2 macro causes XNFMerge to change the RLOC values on objects below it, as well as to add the H_SET tag, as shown in the italicized text.

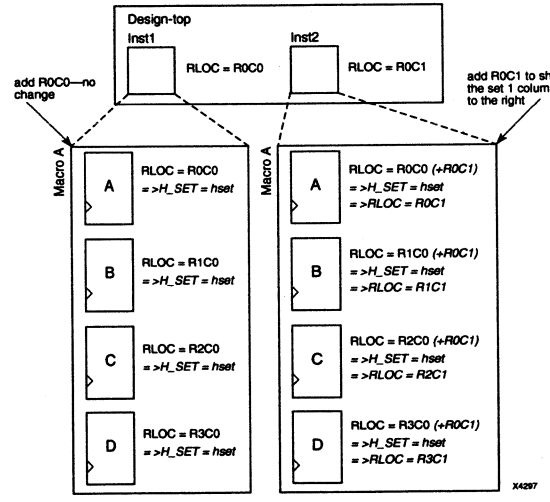


Figure 4-9 Modifying RLOC Values of Same Macro and Linking Together as One Set

HU_SET

The HU_SET constraint is a variation of the implicit H_SET (hierarchy set). Like H_SET, HU_SET is defined by the design hierarchy. However, you can use the HU_SET constraint to assign a user-defined name to the HU_SET.

The syntax of the HU_SET constraint is the following:

HU_SET=*name*

where *name* is the identifier of the set; it must be unique among all the sets in the design. You must define the base names to ensure unique hierarchically qualified names for the sets after XNFMerge flattens the design and attaches the hierarchical names as prefixes.

This user-defined name is the base name of the HU_SET set. Like the H_SET set, in which the base name of "hset" is prefixed by the hierarchical name of the lowest common ancestor of the set elements, the user-defined base name of an HU_SET set is prefixed by the hierarchical name of the lowest common ancestor of the set elements.

The HU_SET constraint defines the start of a new set: all design elements at the same node that have the same user-defined value for the HU_SET constraint are members of the same HU_SET set. Along with the HU_SET constraint, elements can also have an RLOC constraint. The presence of an RLOC constraint in an H_SET constraint links the element to all elements tagged with RLOCs above and below in the hierarchy. However, in the case of an HU_SET constraint, the presence of an RLOC constraint along with the HU_SET constraint on a design element does not automatically link the element to other elements with RLOC constraints at the same hierarchy level or above.

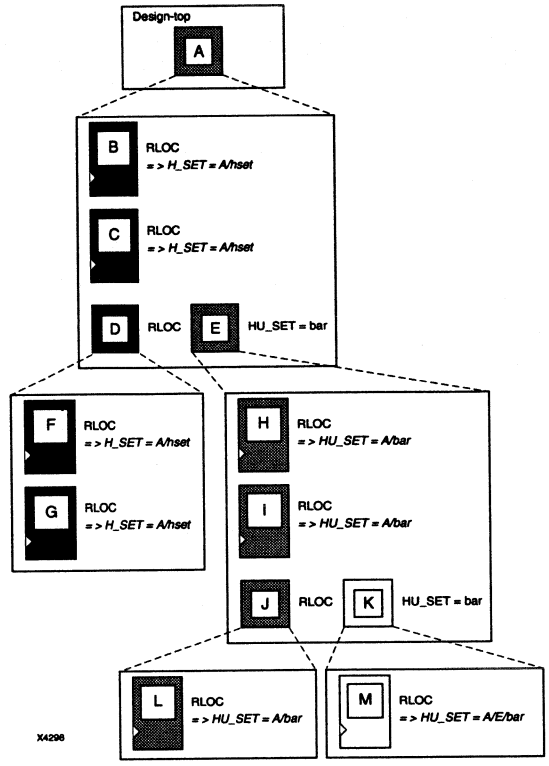


Figure 4-10 HU_SET Constraint Linking and Separating Elements from H_SET Sets

Figure 4-10 demonstrates how HU_SET constraints designate elements as set members, break links between elements tagged with RLOC constraints in the hierarchy to separate them from H_SET sets, and generate names as identifiers of these sets. The user-defined HU_SET constraint on E separates its underlying design elements, namely H, I, J, K, L, and M from the implicit H_SET=A/hset that contains primitive members B, C, F, and G. The HU_SET set that is defined at E includes H, I, and L (through the element J). XNFMerge hierarchically qualifies the name value "bar" on element E to be A/bar, since A is the lowest common ancestor for all the elements of the HU_SET set, and attaches it to the set member primitives H, I, and L. An HU_SET constraint on K starts another set that includes M, which

receives the HU_SET=A/E/bar constraint after processing by XNFMerge. In Figure 4-10, the same name field is used for the two HU_SET constraints, but because they are attached to symbols at different levels of the hierarchy, they define two different sets.

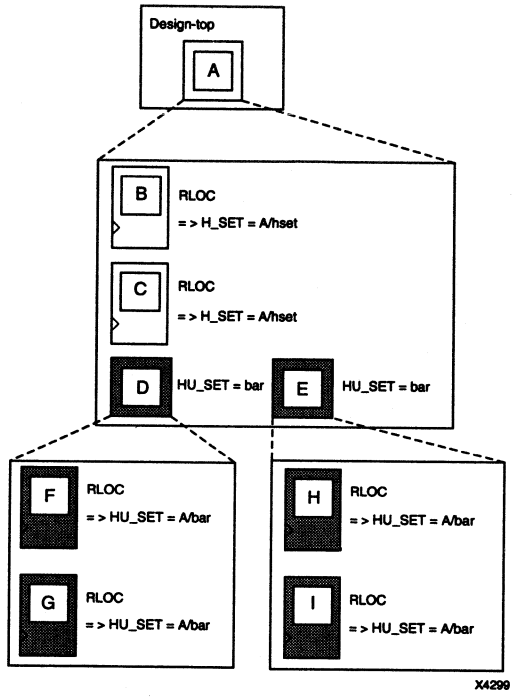


Figure 4-11 Linking Two HU_SET Sets

Figure 4-11 shows how HU_SET constraints link elements in the same node together by naming them with the same identifier. Because of the same name, "bar," on two elements, D and E, the elements tagged with RLOC constraints below D and E become part of the same HU_SET.

Set Modifiers

A modifier, as its name suggests, modifies the RLOC constraints associated with design elements. Since it modifies the RLOC constraints of all the members of a set, it must be applied in a way that propagates it to all the members of the set easily and intuitively.

For this reason, the RLOC modifiers of a set are placed at the start of that set. This section discusses the different modifiers that you can use to modify the RLOC set constraints.

RLOC

As discussed previously, the RLOC constraint associated with a design element modifies the values of other RLOC constraints below the element in the hierarchy of the set. Regardless of the set type, RLOC row, column, and extension values on an element always propagate down the hierarchy and are added at lower levels of the hierarchy to RLOC constraints on elements in the same set.

RLOC_ORIGIN

Specifying RLOC constraints to describe the spatial relationship of the set members to themselves allows the members of the set to float anywhere on the die as a unit. You can, however, fix the exact die location of the set members. The RLOC_ORIGIN constraint allows you to change the RLOC values into absolute LOC constraints that respect the structure of the set.

Following is the syntax of this constraint:

RLOC_ORIGIN=R*row#***C***column#*

where the row and column numbers are positive non-zero integer values. When an RLOC_ORIGIN constraint is applied to a set, the row and column values of the RLOC_ORIGIN are added to the individual RLOC values of the members of the set to obtain a final LOC constraint for each element in the set. Since the row and column numbers of an RLOC_ORIGIN constraint refer to actual die locations, its value must exclude zero.

Note: In the XACT 5.0 release, you must use the RLOC_ORIGIN constraint with sets that include BUFT symbols. Sets with BUFT symbols must be fixed to an exact die location.

The design flattening program, XNFMerge, translates the RLOC_ORIGIN constraint into LOC constraints. The row and column values of the RLOC_ORIGIN are added individually to the members of the set after all RLOC modifications have been made to their row and column values by addition through the hierarchy. The final values are then turned into LOC constraints on individual primitives.

When this constraint is used in conjunction with an implicit H_SET (hierarchy set), it must be placed on the element that is the start of the H_SET set, that is, on the lowest common ancestor of all the members of the set. If you apply an RLOC_ORIGIN constraint to an HU_SET constraint, place it on the element at the start of the HU_SET set, that is, on an element with the HU_SET constraint. However, since there could be several elements linked together with the HU_SET constraint at the same node, the RLOC_ORIGIN constraint can be applied to only one of these elements to prevent more than one RLOC_ORIGIN constraint from being applied to the HU_SET set. Similarly, when used with a U_SET constraint, the RLOC_ORIGIN constraint can be placed on only one element with the U_SET constraint. If you attach the RLOC_ORIGIN constraint to an element that has only an RLOC constraint, the membership of that element in any set is removed, and the element is considered the start of a new H_SET set with the specified RLOC_ORIGIN constraint attached to the newly created set.

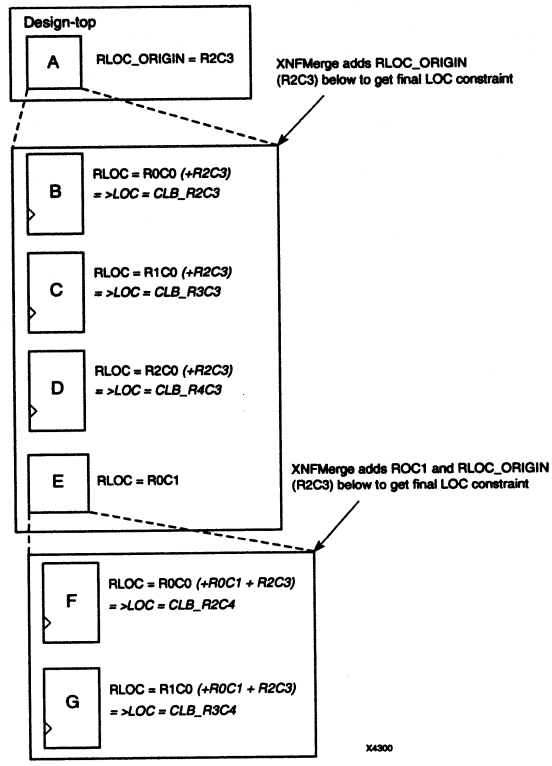


Figure 4-12 Using an RLOC_ORIGIN Constraint to Modify an H_SET Set

In Figure 4-12, the elements B, C, D, F, and G are members of an H_SET set with the name A/hset. This figure is the same as Figure 4-8 except for the presence of an RLOC_ORIGIN constraint at the start of the H_SET set (at A). The RLOC_ORIGIN values are added to the resultant RLOC values at each of the member elements to obtain the values that are then converted by XNFMerge to LOC constraints. For example, the RLOC value of F, given by adding the RLOC value at E (R0C1) and that at F (R0C0), is added to the RLOC_ORIGIN value (R2C3) to obtain the value of (R2C4), which is then converted to a LOC constraint, LOC = CLB_R2C4.

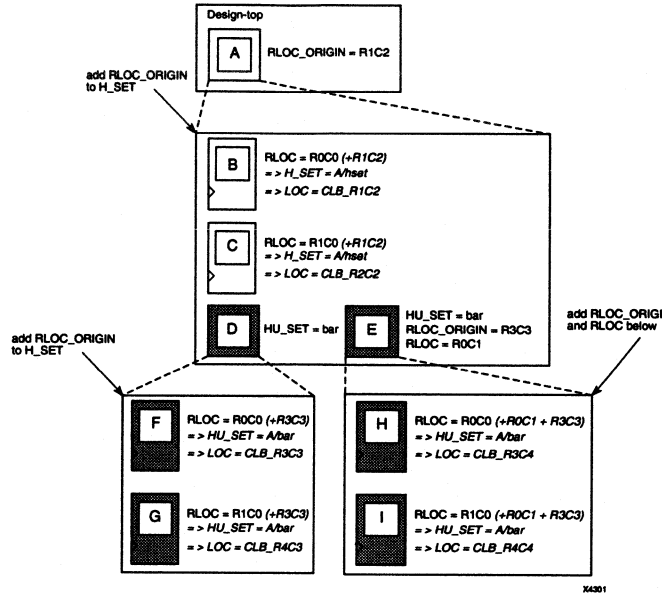


Figure 4-13 Using an RLOC_ORIGIN to Modify H_SET and HU_SET Sets

Figure 4-13 shows an example of an RLOC_ORIGIN constraint modifying an HU_SET constraint. The start of the HU_SET A/bar is given by element D or E. The RLOC_ORIGIN attached to E, therefore, applies to this HU_SET set. On the other hand, the RLOC_ORIGIN at A, which is the start of the H_SET set A/hset, applies to elements B and C, which are members of the H_SET set.

RLOC_RANGE

As noted in the previous discussion, you can fix the members of a set at exact die locations with the RLOC_ORIGIN constraint. In the XACT 5.0 release, you must use the RLOC_ORIGIN constraint with sets that include BUFT symbols. However, for sets that do not include BUFT symbols, you can limit the members of a set to a certain range on the die. In this case, the set could "float" as a unit within the range until a final placement. Since every member of the set must fit within the range, it is important that you specify a range that defines an area large enough to respect the spatial structure of the set.

The syntax of this constraint is the following:

RLOC_RANGE=R*row1*#C*col1*#:R*row2*#C*col2*#

where *row1*, *row2*, *col1*, and *col2* can be non-zero positive numbers, or the wildcard (*) character. This syntax allows for three kinds of range specifications:

- R*r1*C*c1*:R*r2*C*c2* — A rectangular region enclosed by rows r1, r2, and columns c1, c2
- R*C*c1*:R*C*c2* — A region enclosed by the columns c1 and c2 (any row number)
- R*r1*C*:R*r2*C*— A region enclosed by the rows r1 and r2 (any column number)

For the second and third kinds of specifications with wildcards, applying the wildcard asterisk differently on either side of the separator colon creates an error. For example, specifying R*C1:R2C* is an error since the wildcard asterisk is applied to rows on one side and to columns on the other side of the separator colon.

The values of the RLOC_RANGE constraint are not simply added to the RLOC values of the elements. In fact, the RLOC_RANGE constraint does not change the values of the RLOC constraints on underlying elements. It is an additional constraint that is attached automatically by XNFMerge to every member of a set. The RLOC_RANGE constraint is attached to design elements in exactly the same way as the RLOC_ORIGIN constraint. The values of the RLOC_RANGE constraint, like RLOC_ORIGIN values, must be non-zero positive numbers since they directly correspond to die locations.

USE_RLOC

Another important set modifier is the USE_RLOC constraint. It turns the RLOC constraints on and off for a specific element or section of a set.

The syntax of this constraint is:

USE_RLOC=*value*

where *value* is either True or False.

The application of the USE_RLOC constraint is strictly based on hierarchy. A USE_RLOC constraint attached to an element applies to

all its underlying elements that are members of the same set. If it is attached to a symbol that defines the start of a set, the constraint is applied to all the underlying member elements, which represent the entire set. However, if it is applied to an element below the start of the set (for example, E in Figure 4-14), only the members of the set (H and I) below the specified element are affected.You can also attach the USE_RLOC constraint directly to a primitive symbol so that it affects only that symbol.

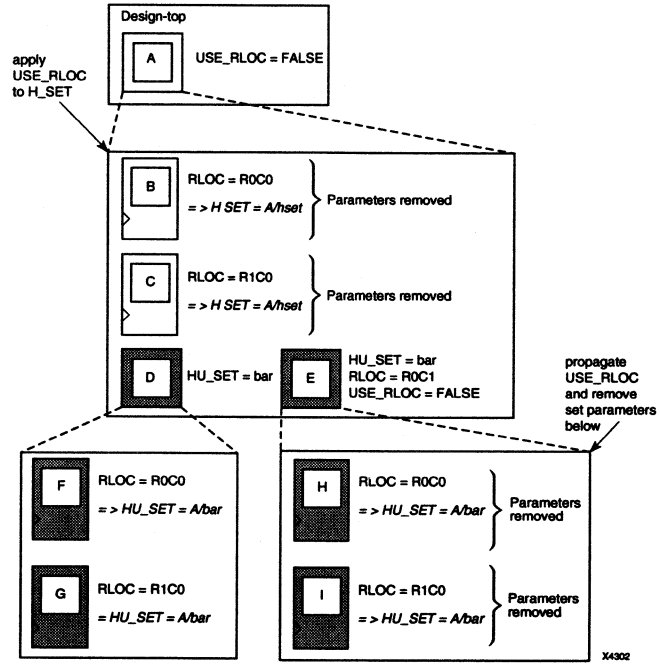


Figure 4-14 Using the USE_RLOC Constraint to Control RLOC Application on H_SET and HU_SET Sets

When the USE_RLOC=false constraint is applied, the RLOC and set constraints are removed from the affected symbols in the XNFMerge output file. This process is different than that followed for the RLOC_ORIGIN constraint. For RLOC_ORIGIN, XNFMerge generates and outputs a LOC constraint in addition to all the set and RLOC constraints in the output file. XNFMerge does not retain the

original constraints in the presence of a USE_RLOC=false constraint because these cannot be turned on again in later programs.

Figure 4-14 illustrates the use of the USE_RLOC constraint to mask an entire set as well as portions of a set.

Applying the USE_RLOC constraint on U_SET sets is a special case because of the lack of hierarchy in the U_SET set. Because the USE_RLOC constraint propagates strictly in a hierarchical manner, the members of a U_SET set that are in different parts of the design hierarchy must be tagged separately with USE_RLOC constraints; no single USE_RLOC constraint is propagated to all the members of the set that lie in different parts of the hierarchy. If you create a U_SET set through an instantiating macro, you can attach the USE_RLOC constraint to the instantiating macro to allow it to propagate hierarchically to all the members of the set. You can create this instantiating macro by placing a U_SET constraint on a macro and letting XNFMerge propagate that constraint to every symbol with an RLOC constraint below it in the hierarchy.

Figure 4-15 illustrates an example of the use of the USE_RLOC=false constraint. The USE_RLOC=false on primitive E removes it from the U_SET set, and USE_RLOC=false on element F propagates to primitive G and removes it from the U_SET set.

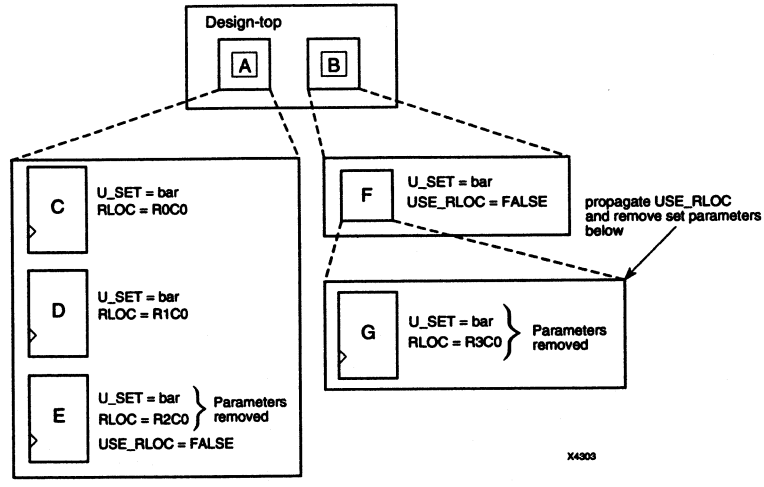


Figure 4-15 Using the USE_LOC Constraint to Control RLOC Application on U_Sets

While propagating the USE_RLOC constraint, XNFMerge ignores underlying USE_RLOC constraints if it encounters elements higher in the hierarchy that already have USE_RLOC constraints. For example, if XNFMerge encounters an underlying element with a USE_RLOC=true constraint during the propagation of a USE_RLOC=false constraint, it ignores the newly encountered True constraint.

Xilinx Macros

Xilinx-supplied flip-flop macros include an RLOC_R0C0 constraint on the underlying primitive, which allows you to attach an RLOC to the macro symbol. This symbol links the underlying primitive to the set that contains the macro symbol. Simply attach an appropriate RLOC constraint to the instantiation of the actual Xilinx flip-flop macro. XNFMerge adds the RLOC value that you specified to the underlying primitive so that it has the desired value.

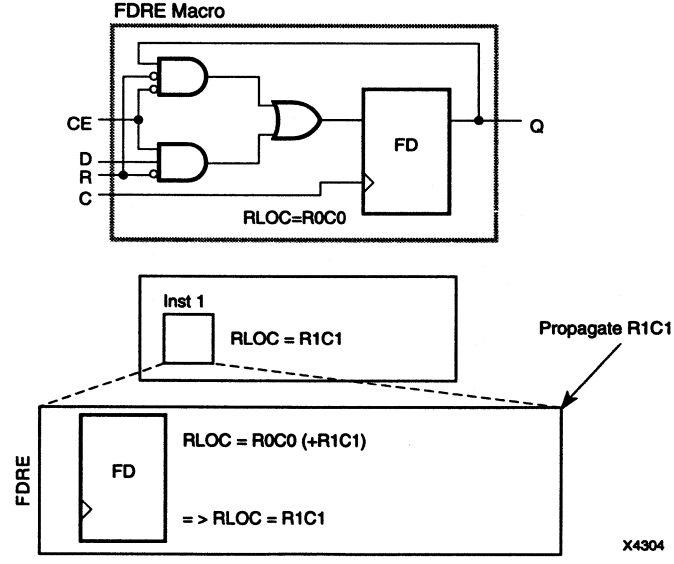


Figure 4-16 Typical Use of a Xilinx Macro

For example, in Figure 4-16, the RLOC = R1C1 constraint is attached to the instantiation (Inst1) of the FDRE macro. It is added to the R0C0 value of the RLOC constraint on the flip-flop within the macro to obtain the new RLOC values.

If you do not put an RLOC constraint on the flip-flop macro symbol, the underlying primitive symbol is the lone member of a set. XNFMerge removes RLOC constraints from a primitive that is the only member of a set or from a macro that has no RLOC objects below it.

LOC Propagation Through Design Flattening

XNFMerge continues to propagate LOC constraints down the design hierarchy. It adds this constraint to appropriate objects that are not members of a set. While RLOC constraint propagation is limited to sets, the LOC constraint is applied from its start point all the way down the hierarchy.

Summary

Table 4-15 summarizes the RLOC set types and the constraints that identify members of these sets.

Table 4-15 Summary of Set Types

Type	Definition	Naming	Linkage	Modification
Set	A set is a collection of elements to which relative location constraints are applied.			
U_SET= *name*	All elements with the same user-tagged U_SET constraint value are members of the same U_SET set.	The name of the set is the same as the user-defined name without any hierarchical qualification.	U_SET links elements to all other elements with the same value for the U_SET constraint.	U_SET is modified by applying RLOC_ORIGIN or RLOC_RANGE constraints on, at most, one of the U_SET constraint-tagged elements.

Type	Definition	Naming	Linkage	Modification
H_SET (implicit through hierarchy) is not available as a constraint that you can attach to symbols.	RLOC on the node. Any other constraint removes a node from the H_SET set.	The lowest common ancestor of the members defines the start of the set. The name is the hierarchically qualified name of the start followed by the base name, "hset."	H_SET links elements to other elements at the same node that do not have other constraints. It links down to all elements that have RLOC constraints and no other constraints. Similarly, it links to other elements up the hierarchy that have RLOC constraints but no other constraints.	H_SET is modified by applying RLOC_ORIGIN and RLOC_RANGE at the start of the set: the lowest common ancestor of all the elements of the set.
HU_SET=*name*	All elements with the same hierarchically qualified name are members of the same set.	The lowest common ancestor of the members is prefixed to the user-defined name to obtain the name of the set.	HU_SET links to other elements at the same node with the same HU_SET constraint value. It links to elements with RLOC constraints below.	The start of the set is made up of the elements on the same node that are tagged with the same HU_SET constraint value. An RLOC_ORIGIN or an RLOC_RANGE can be applied to, at most, one of these start elements of an HU_SET set.

Relationally Placed Macros (RPMs)

The Xilinx libraries contain three types of elements.

- Primitives are basic logical elements such as AND2 and OR2 gates.
- Soft macros are schematics made by combining primitives and sometimes other soft macros.
- Relationally placed macros (RPMs) are soft macros that contain relative location constraint (RLOC) information, carry logic symbols, and FMAP/HMAP symbols, where appropriate. RPMs are currently only available in the XC4000 library.

Designs created with RPMs can be functionally simulated.

The HM2RPM utility translates old custom hard macro files into RPM files. If you created your own hard macro files, you must run HM2RPM on each hard macro file and place the new XNF file in your current working directory or in a search directory specified for XNFMerge. For instructions on using the HM2RPM utility, see the "HM2RPM" chapter of the *XACT Reference Guide*.

RPMs can, but need not, include all the following elements:

- FMAPs, HMAPs, and CLB-grouping attributes to control mapping. FMAPs and HMAPs have pin-lock attributes, which allow better control over routing. FMAPs and HMAPs are described in the "Mapping Constraints" section of the "PPR Placement Constraints" section earlier in this chapter.
- Relative location (RLOC) constraints to provide placement structure. They allow positioning of elements relative to each other. They are discussed in the "Relative Location Constraints" section earlier in this chapter.
- Carry logic primitive symbols. Carry logic is discussed in the next section, "Carry Logic in XC4000 LCAs."

These elements allow you to access carry logic easily and to control mapping and block placement. Because RPMs are a super-set of ordinary macros, you can design them in the normal design entry environment. They can include any primitive logic. The macro logic is fully visible to you and can be easily back-annotated with timing information.

RPMs do not include routing capability. XACT-Performance specifications address timing issues more effectively.

Carry Logic in XC4000 LCAs

This section describes the use of carry logic in XC4000 CLBs and lists all the carry logic configuration mnemonics available. The XC4000 carry logic modes are shown in the following figure.

Figure 4-17 XC4000 Carry Logic Modes

The XC4000 CLB contains a feature called dedicated carry logic. This carry logic is independent of the function generators, although it shares some of the same input pins. Dedicated interconnect propagates carry signals through a column of CLBs. The carry logic in each CLB can implement approximately 40 different functions, which you can use to build faster and more efficient adders, subtracters, counters, comparators, and so forth. Figure 4-18 shows the carry logic in an XC4000 CLB.

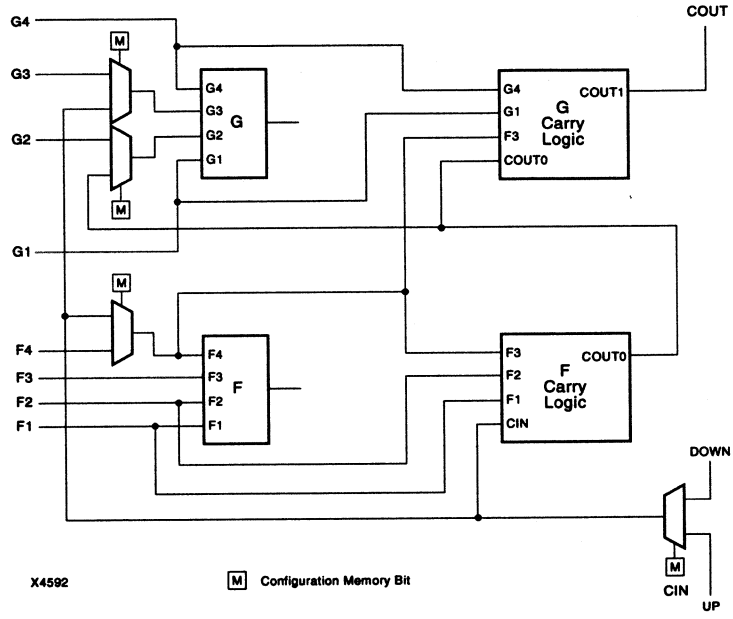


Figure 4-18 XC4000 CLB Carry Logic

Primitives and Symbols

The schematic capture libraries that Xilinx supports contain one generic carry logic primitive and several specific carry mode primitive symbols. The generic carry logic primitive represents the complete carry logic in a single CLB. The carry mode primitive symbols represent unique carry modes, such as ADD-FG-CI. To specify the particular mode that you wish, connect a carry mode symbol to the C0-C7 mode pins of the carry logic symbol. It is the pair of symbols that defines the specific kind of carry logic desired.

A carry logic symbol requires you to place either a LOC or an RLOC constraint on it. If a LOC constraint is used, it must be a single LOC= constraint; it cannot be an area or prohibit LOC constraint or use wildcards in its syntax.

Table 4-16 lists the carry mode names and symbols.

Table 4-16 Carry Modes

Carry Mode Name	Symbol
ADD-F-CI	cy4_01
ADD-FG-CI	cy4_02
ADD-G-F1	cy4_03
ADD-G-CI	cy4_04
ADD-G-F3	cy4_05
ADDSUB-F-CI	cy4_12
ADDSUB-FG-CI	cy4_13
ADDSUB-G-F1	cy4_14
ADDSUB-G-CI	cy4_15
ADDSUB-G-F3	cy4_16
FORCE-0	cy4_37
FORCE-1	cy4_38
FORCE-F1	cy4_39
FORCE-CI	cy4_40
FORCE-F3	cy4_41
EXAMINE-CI	cy4_42
DEC-F-CI	cy4_24
DEC-FG-CI	cy4_25
DEC-FG-0	cy4_26
DEC-G-0	cy4_27
DEC-G-F1	cy4_28
DEC-G-CI	cy4_29
DEC-G-F3-	cy4_30

Carry Mode Name	Symbol
INC-F-CI	cy4_17
INC-FG-CI	cy4_18
INC-FG-1	cy4_19
INC-G-1	cy4_20
INC-G-F1	cy4_21
INC-G-CI	cy4_22
INC-G-F3-	cy4_23
SUB-F-CI	cy4_06
SUB-FG-CI	cy4_07
SUB-G-1	cy4_08
SUB-G-F1	cy4_10
SUB-G-CI	cy4_09
SUB-G-F3	cy4_11
INCDEC-F-CI	cy4_31
INCDEC-FG-CI	cy4_32
INCDEC-FG-1	cy4_33
INCDEC-G-0	cy4_34
INCDEC-G-F1	cy4_35
INCDEC-G-CI	cy4_36

cy4 and cy4_n are not supported by XC7000.

Carry Logic Handling in XNFPrep

The XNFPrep program checks for legal connections between carry logic symbols and also performs simple trimming on some carry modes. CY4 symbols might be trimmed as follows:

- If neither the COUT0 pin nor the COUT pin is used, the CY4 symbol is removed from the design. However, if the signal on the CIN pin connects to other logic, XNFPrep converts the CY4 to the EXAMINE-CI mode. An EXAMINE-CI mode CY4 is trimmed only if there is no other load on the signal on the CIN pin.

- If the COUT0 pin is used but the COUT pin is not, XNFPrep attempts to convert the CY4 symbol to use a 1-bit equivalent mode. That is, if the mode was originally of the form -FG-CI, it converts it to the equivalent -F-CI mode, allowing signals to be removed from the CY4 A1 and B1 operand inputs, which may save routing resources.
- If the specified mode does not require any of the A0, B0, A1, B1, and/or ADD CY4 inputs, XNFPrep removes the signals from these pins, which may save routing resources.

Carry Mode Configuration Mnemonics

The first step in configuring a CLB for carry logic is to choose the appropriate carry mode configuration mnemonic. Each of the 42 possible configurations of the carry logic has been assigned a three-part mnemonic code, for example:

```
ADD-FG-CI
```

- The first field (ADD) describes the operation performed in the CLB function generators, in this case, a binary addition. By implication, the carry logic in this CLB calculates the carry for this addition.
- The second field (FG) indicates which of the two function generators is used in the specified operation, in this case, both F and G.
- The last field (CI) specifies the source of the carry-in signal to the CLB, in this case, the CIN pin itself.

Consider another example:

```
INCDEC-G-F1
```

This mnemonic describes a CLB in which the G function generator performs an increment/decrement function. The carry-in to this CLB is sourced by the F1 pin.

All available carry mode configuration mnemonics are listed in the next section, "Carry Logic Configurations."

To determine which carry mode primitive corresponds to which mnemonic, see Table 4-16.

Carry Logic Configurations

This section lists and describes all the available carry mode configuration mnemonics. The following information is given for each mnemonic:

- The name of the mode mnemonic
- A brief description of the CLB function
- The COUT0 and COUT1 equations performed by the carry logic
- Default equations for the F and G function generators
- Default assignments for the F4, G2, and G3 inputs

The default F and G functions and default F4, G2, and G3 inputs are based on the generic CLB function described. You can change these defaults as required, allowing for features such as parallel enable or synchronous reset. However, if these defaults are changed, the CLB may no longer function as the mnemonic describes.

The COUT0 and COUT1 equations are absolutely determined by the carry mode configuration mnemonic. The only way to change these carry logic outputs is by selecting a different mnemonic.

ADD-F-CI

The ADD-F-CI configuration performs a 1-bit addition of A+B in the F function generator, with the A and B inputs on the F1 and F2 pins. The carry signal enters on the CIN pin, propagates through the F carry logic, and exits on the COUT pin. This configuration can be used as the MSB of an adder, with the G function generator accessing the carry-out signal or calculating a twos-complement overflow.

F=(F1@F2)@F4

COUT0=(F1*F2) + CIN*(F1+F2)

G=

COUT1=COUT0

F4=CIN

G2=G2I (COUT0 for overflow, OFL=G2@G3, or for carry-out, CO=G2)

G3=G3I (CIN for overflow, OFL=G2@G3)

ADD-FG-CI

The ADD-FG-CI configuration performs a 2-bit addition of A+B in both the F and G function generators, with the lower-order A and B inputs on the F1 and F2 pins, and the higher-order A and B inputs on the G1 and G4 pins. The carry signal enters on the CIN pin, propagates through the F and G carry logic, and exits on the COUT pin. This configuration comprises the middle bits of an adder.

F=(F1@F2)@F4

COUT0=(F1*F2) + CIN*(F1+F2)

G=(G4@G1)@G2

COUT1=(G4*G1) + COUT0*(G4+G1)

F4=CIN

G2=COUT0

G3=G3I

ADD-G-F1

The ADD-G-F1 configuration performs a 1-bit addition of A+B in the G function generator, with the A and B inputs on the G1 and G4 pins. The carry signal enters on the F1 pin, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of an adder, where the carry-in signal is routed to F1. The F function generator is not used.

F=

COUT0=F1

G=(G4@G1)@G2

COUT1=(G4*G1) + COUT0*(G4+G1)

F4=F4I

G2=COUT0

G3=G3I

ADD-G-CI

The ADD-G-CI configuration performs a 1-bit addition of A+B in the G function generator, with the A and B inputs on the G1 and G4 pins. The carry signal enters on the CIN pin, propagates through the G carry logic, and exits on the COUT pin. This configuration is for the middle bit of an adder, where the F function generator is reserved for another purpose.

F=

COUT0=CIN

G=(G4@G1)@G2

COUT1=(G4*G1) + COUT0*(G4+G1)

F4=F4I

G2=COUT0

G3=G3I

ADD-G-F3-

The ADD-G-F3- configuration performs a 1-bit addition of A+B in the G function generator, with the A and B inputs on the G1 and G4 pins. The carry signal enters on the F3 pin, is inverted by the F carry logic, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of an adder, where the inverted carry-in signal is routed to F3. The F function generator is not used.

F=

COUT0=~F3

G=(G4@G1)@G2

COUT1=(G4*G1) + COUT0*(G4+G1)

F4=F4I

G2=COUT0

G3=G3I

SUB-F-CI

The SUB-F-CI configuration performs a 1-bit twos-complement subtraction of A-B in the F function generator, with the A input on F1 and the B input on F2. The carry signal enters on the CIN pin, propagates through the F carry logic, and exits on the COUT pin. This configuration can be used as the MSB of a subtracter, with the G function generator accessing the carry-out signal or calculating a twos-complement overflow.

F=(F1@F2)@~F4=~(F1@F2@F4)

COUT0=(F1*~F2) + CIN*(F1+~F2)

G=

COUT1=COUT0

F4=CIN

G2=G2I (COUT0 for overflow, OFL=G2@G3, or for carry-out, CO=G2)

G3=G3I (CIN for overflow, OFL=G2@G3)

SUB-FG-CI

The SUB-FG-CI configuration performs a 2-bit twos-complement subtraction of A-B in both the F and G function generators. For the lower bit, the A input is on F1 and the B input is on F2. For the upper bit, the A input is on G4 and the B input is on G1. The carry signal enters on the CIN pin, propagates through the F and G carry logic, and exits on the COUT pin. This configuration comprises the middle bits of a subtracter.

F=(F1@F2)@~F4=~(F1@F2@F4)

COUT0=(F1*~F2) + CIN*(F1+~F2)

G=(G4@G1)@~G2=~(G4@G1@G2)

COUT1=(G4*~G1) +COUT0*(G4+~G1)

F4=CIN

G2=COUT0

G3=G3I

SUB-G-1

The SUB-G-1 configuration performs a 1-bit twos-complement subtraction of A-B in the G function generator, with the A input on G4 and the B input on G1. The carry-in is tied High (no borrow). The carry signal propagates through the G carry logic and exits on the COUT pin. This configuration comprises the LSB of a subtracter with no carry-in. The F function generator is not used.

F=

COUT0=1

G=(G4@G1)

COUT1=(G4+~G1)

F4=F4I

G2=G2I

G3=G3I

SUB-G-F1

The SUB-G-F1 configuration performs a 1-bit twos-complement subtraction of A-B in the G function generator, with the A input on G4 and the B input on G1. The carry signal enters on the F1 pin, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of a subtracter, where the carry-in signal is routed to F1. The F function generator is not used.

F=

COUT0=F1

G=(G4@G1)@~G2=~(G4@G1@G2)

COUT1=(G4*~G1) + COUT0*(G4+~G1)

F4=F4I

G2=COUT0

G3=G3I

SUB-G-CI

The SUB-G-CI configuration performs a 1-bit twos-complement subtraction of A-B in the G function generator, with the A input on G4 and the B input on G1. The carry signal enters on the CIN pin, propagates through the G carry logic, and exits on the COUT pin. This configuration is for the middle bit of a subtracter, where the F function generator is reserved for another purpose.

F=

COUT0=CIN

G=(G4@G1)@~G2=~(G4@G1@G2)

COUT1=(G4*~G1) + COUT0*(G4+~G1)

F4=F4I

G2=COUT0

G3=G3I

SUB-G-F3-

The SUB-G-F3- configuration performs a 1-bit twos-complement subtraction of A-B in the G function generator, with the A input on G4 and the B input on G1. The carry signal enters on the F3 pin, is inverted by the F carry logic, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of a subtracter, where the inverted carry-in signal is routed to F3. The F function generator is not used.

F=

COUT0=~F3

G=(G4@G1)@~G2=~(G4@G1@G2)

COUT1=(G4*~G1) + COUT0*(G4+~G1)

F4=F4I

G2=COUT0

G3=G3I

ADDSUB-F-CI

The ADDSUB-F-C1 configuration performs a 1-bit twos-complement add/subtract of A±B in the F function generator, with the A input on F1 and the B input on F2. The carry signal enters on the CIN pin, propagates through the F carry logic, and exits on the COUT pin. The F3 input indicates add (F3=1) or subtract (F3=0). This configuration can be used as the MSB of an adder/subtracter, with the G function generator accessing the carry-out signal or calculating a twos-complement overflow.

F=(F1@F2)@F4@~F3=~(F1@F2@F4@F3)

COUT0=F3*((F1*F2) + CIN*(F1+F2)) + ~F3*((F1*~F2) + CIN*(F1+~F2))

G=

COUT1=COUT0

F4=CIN

G2=G2I (COUT0 for overflow, OFL=G2@G3, or for carry-out, CO=G2)

G3=G3I (CIN for overflow, OFL=G2@G3)

ADDSUB-FG-CI

The ADDSUB-FG-CI configuration performs a 2-bit twos-complement add/subtract of A±B in both the F and G function generators. For the lower bit, the A input is on F1 and the B input is on F2. For the upper bit, the A input is on G4 and the B input is on G1. The carry signal enters on the CIN pin, propagates through the F and G carry logic, and exits on the COUT pin. The F3 and G3 inputs indicate add (F3=G3=1) or subtract (F3=G3=0): the add/subtract control signal must be routed to both the F3 and G3 pins. This configuration comprises the middle bits of an adder/subtracter.

F=(F1@F2)@F4@~F3=~(F1@F2@F4@F3)

COUT0=F3*((F1*F2) + CIN*(F1+F2)) + ~F3*((F1*~F2) + CIN*(F1+~F2))

G=(G4@G1)@G2@~G3=~(G4@G1@G2@G3)

COUT1=F3*((G4*G1)+COUT0*(G4+G1))+~F3*((G4*~G1)+COUT0*(G4+~G1))

F4=CIN

G2=COUT0

G3=G3I

ADDSUB-G-F1

The ADDSUB-G-F1 configuration performs a 1-bit twos-complement add/subtract of A±B in the G function generator, with the A input on G4 and the B input on G1. The carry signal enters on the F1 pin, propagates through the G carry logic, and exits on the COUT pin. The F3 and G3 inputs indicate add (F3=G3=1) or subtract (F3=G3=0): the add/subtract control signal must be routed to both the F3 and G3 pins. This configuration comprises the LSB of an adder/subtracter, where the carry-in signal is routed to F1. The F function generator is not used.

F=

COUT0=F1

G=(G4@G1)@G2@~G3=~(G4@G1@G2@G3)

COUT1=F3*((G4*G1)+COUT0*(G4+G1))+~F3*((G4*~G1)+COUT0*(G4+~G1))

F4=F4I

G2=COUT0

G3=G3I

ADDSUB-G-CI

The ADDSUB-G-CI configuration performs a 1-bit twos-complement add/subtract of A±B in the G function generator, with the A input on G4 and the B input on G1. The carry signal enters on the CIN pin, propagates through the G carry logic, and exits on the COUT pin. The F3 and G3 inputs indicate add (F3=G3=1) or subtract (F3=G3=0): the add/subtract control signal must be routed to both the F3 and G3 pins. This configuration is for the middle bit of an adder/subtracter, where the F function generator is reserved for another purpose.

F=

COUT0=CIN

G=(G4@G1)@G2@~G3=~(G4@G1@G2@G3)

COUT1=F3*((G4*G1)+COUT0*(G4+G1))+~F3*((G4*~G1)+COUT0*(G4+~G1))

F4=F4I

G2=COUT0

G3=G3I

ADDSUB-G-F3-

The ADDSUB-G-F3 configuration performs a 1-bit twos-complement add/subtract of A±B in the G function generator, with the A input on G4 and the B input on G1. The carry signal enters on the F3 pin, is inverted by the F carry logic, propagates through the G carry logic, and exits on the COUT pin. Because the F3 input also indicates add (F3=1) or subtract (F3=0), the carry-in is always null (0 for add, 1 for subtract). This configuration comprises the LSB of an adder/subtracter with no carry-in. The F function generator is not used.

F=

COUT0=~F3

G=(G4@G1)

COUT1=F3*G4*G1 + ~F3(G4+~G1)

F4=F4I

G2=COUT0

G3=G3I

INC-F-CI

The INC-F-CI configuration performs a 1-bit increment in the F function generator, with the input on the F1 pin. The carry signal enters on the CIN pin, propagates through the F carry logic, and exits on the COUT pin. The G function generator can be used to output the terminal count of a counter.

F=(F1@F4)

COUT0=CIN*F1

G=

COUT1=COUT0

F4=CIN

G2=G2I (COUT0 for terminal count, TC=G2)

G3=G31

INC-FG-CI

The INC-FG-CI configuration performs a 2-bit increment in both the F and G function generators, with the lower-order input on the F1 pin and the higher-order input on the G4 pin. The carry signal enters on the CIN pin, propagates through the F and G carry logic, and exits on the COUT pin. This configuration comprises the middle bits of an incrementer.

F=(F1@F4)

COUT0=CIN*F1

G=(G4@G2)

COUT1=COUT0*G4

F4=CIN

G2=COUT0

G3=G3I

INC-G-1

The INC-G-1 configuration performs a 1-bit increment in the G function generator, with the input on the G4 pin. The carry-in is tied High. The carry signal propagates through the G carry logic and exits on the COUT pin. This configuration comprises the LSB of an incrementer that is always enabled. The F function generator is not used. This configuration is identical to DEC-G-0, since the LSB of an incrementer is identical to the LSB of a decrementer.

F=

COUT0=0

G=~(G4)

COUT1=G4

F4=F4I

G2=G2I

G3=G3I

INC-G-F1

The INC-G-F1 configuration performs a 1-bit increment in the G function generator, with the input on the G4 pin. The carry signal enters on the F1 pin, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of an incrementer where F1 is an active-High enable. The F function generator is not used.

F=

COUT0=F1

G=(G4@G2)

COUT1=COUT0*G4

F4=F4I

G2=COUT0

G3=G3I

INC-G-CI

The INC-G-CI configuration performs a 1-bit increment in the G function generator, with the input on the G4 pin. The carry signal enters on the CIN pin, propagates through the G carry logic, and exits on the COUT pin. This configuration is for the middle bit of an incrementer where the F function generator is reserved for another purpose.

F=

COUT0=CIN

G=(G4@G2)

COUT1=COUT0*G4

F4=F4I

G2=COUT0

G3=G3I

INC-G-F3-

The INC-G-F3- configuration performs a 1-bit increment in the G function generator, with the input on the G4 pin. The carry signal enters on the F3 pin, is inverted in the F carry logic, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of an incrementer where F3 is an active-Low enable. The F function generator is not used.

F=

COUT0=~F3

G=(G4@G2)

COUT1=COUT0*G4=~F3*G4

F4=F4I

G2=COUT0

G3=G3I

INC-FG-1

The INC-FG-1 configuration performs a 2-bit increment in both the F and G function generator, with the lower-order A input on the F1 pin and the higher-order A input on the G4 pin. The carry-in is tied High. The carry signal propagates through the F and G carry logic and exits on the COUT pin. This configuration comprises the two least significant bits of an incrementer that is always enabled.

F=~(F1)

COUT0=F1

G=G2@G4

COUT1=COUT0*G4

F4=F4I or CIN

G2=COUT0

G3=G3I or CIN

DEC-F-CI

The DEC-F-CI configuration performs a 1-bit decrement in the F function generator, with the input on the F1 pin. The carry signal enters on the CIN pin, propagates through the F carry logic, and exits on the COUT pin. The G function generator can be used to output the terminal count of a counter.

F=~(F1@F4)

COUT0=F1+CIN*~F1

G=

COUT1=COUT0

F4=CIN

G2=G2I (COUT0 for terminal count, TC=G2)

G3=G31

DEC-FG-CI

The DEC-FG-CI configuration performs a 2-bit decrement in both the F and G function generators, with the lower-order input on the F1 pin and the higher-order input on the G4 pin. The carry signal enters on the CIN pin, propagates through the F and G carry logic, and exits on the COUT pin. This configuration comprises the middle bits of a decrementer.

F=~(F1@F4)

COUT0=F1+CIN*~F1

G=~(G4@G2)

COUT1=G4+COUT0*~G4

F4=CIN

G2=COUT0

G3=G3I

DEC-G-0

The DEC-G-0 configuration performs a 1-bit decrement in the G function generator, with the input on the G4 pin. The carry-in is tied High (no borrow). The carry signal propagates through the G carry logic and exits on the COUT pin. This configuration comprises the LSB of a decrementer that is always enabled. The F function generator is not used. This configuration is identical to INC-G-1, since the LSB of an incrementer is identical to the LSB of a decrementer.

F=

COUT0=0

G=~(G4)

COUT1=G4

F4=F4I

G2=G2I

G3=G3I

DEC-G-F1

The DEC-G-F1 configuration performs a 1-bit decrement in the G function generator, with the input on the G4 pin. The carry signal enters on the F1 pin, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of a decrementer where F1 is an active-Low enable. The F function generator is not used.

F=

COUT0=F1

G=~(G4@G2)

COUT1=COUT0 + G4

F4=F4I

G2=COUT0

G3=G3I

DEC-G-CI

The DEC-G-CI configuration performs a 1-bit decrement in the G function generator, with the input on the G4 pin. The carry signal enters on the CIN pin, propagates through the G carry logic, and exits on the COUT pin. This configuration is for the middle bit of a decrementer, where the F function generator is reserved for another purpose.

F=

COUT0=CIN

G=~(G4@G2)

COUT1=G4+COUT0*~G4

F4=F4I

G2=COUT0

G3=G3I

DEC-G-F3-

The DEC-G-F3- configuration performs a 1-bit decrement in the G function generator, with the input on the G4 pin. The carry signal enters on the F3 pin, is inverted in the F carry logic, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of a decrementer, where F3 is an active-High enable. The F function generator is not used.

F=

COUT0=~F3

G=~(G4@G2)

COUT1=COUT0 + G4

F4=F4I

G2=COUT0

G3=G3I

DEC-FG-0

The DEC-FG-0 configuration performs a 2-bit decrement in both the F and G function generator, with the lower-order input on the F1 pin and the higher order input on the G4 pin. The carry-in is tied Low. The carry signal propagates through the F and G carry logic and exits on the COUT pin. This configuration comprises the two least significant bits of a decrementer that is always enabled.

F=~(F1)

COUT0=F1

G=~(G4@G2)

COUT=COUT1=(COUT0*~G4) + G4

F4=F4I

G2=COUT0

G3=G3I

INCDEC-F-CI

The INCDEC-F-CI configuration performs a 1-bit increment/decrement in the F function generator, with the input on the F1 pin. The carry signal enters on the CIN pin, propagates through the F carry logic, and exits on the COUT pin. The F3 input indicates increment (F3=1) or decrement (F3=0). The G function generator can be used to output the terminal count of a counter.

F=(F1@F4)@~F3

COUT0=~F3*(F1+ CIN) + F3*F1*CIN

G=

COUT1=COUT0

F4=CIN

G2=G2I (COUT0 for terminal count, TC=G2)

G3=G31

INCDEC-FG-CI

The INCDEC-FG-CI configuration performs a 2-bit increment/decrement in both the F and G function generators, with the lower-order input on the F1 pin and the higher-order input on the G4 pin. The carry signal enters on the CIN pin, propagates through the F and G carry logic, and exits on the COUT pin. The F3 and G3 inputs indicate increment (F3=G3=1) or decrement (F3=G3=0): the increment/decrement control signal must be routed to both the F3 and G3 pins. This configuration comprises the middle bits of an incrementer/decrementer.

F=(F1@F4)@~F3

COUT0=~F3*(F1+ CIN) + F3*F1*CIN

G=(G4@G2)@~G3

COUT1=~F3*(G4+ COUT0) + F3*G4*COUT0

F4=CIN

G2=COUT0

G3=G3I

INCDEC-G-0

The INCDEC-G-0 configuration performs a 1-bit increment/decrement in the G function generator, with the input on the G4 pin. The carry-in is tied High. The carry signal propagates through the G carry logic and exits on the COUT pin. This configuration comprises the LSB of an incrementer/decrementer that is always enabled. The F function generator is not used. F3 is not required for increment/decrement control, since the LSB of an incrementer is identical to the LSB of a decrementer; this configuration is identical to INC-G-1 and DEC-G-0.

F=

COUT0=0

G=~(G4)

COUT1=G4

F4=F4I

G2=G2I

G3=G3I

INCDEC-G-F1

The INCDEC-G-F1 configuration performs a 1-bit increment/decrement in the G function generator, with the input on the G4 pin. The carry signal enters on the F1 pin, propagates through the G carry logic, and exits on the COUT pin. This configuration comprises the LSB of an incrementer/decrementer where the carry-in signal is routed to F1. The carry-in is active-High for an increment operation and active-Low for a decrement operation. The F function generator is not used. The F3 and G3 inputs indicate increment (F3=G3=1) or decrement (F3=G3=0): the increment/decrement control signal must be routed to both the F3 and G3 pins.

F=

COUT0=F1

G=(G4@G2)@~G3

COUT1=F3*(G4*COUT0) + ~F3*(G4+COUT0)

F4=F4I

G2=COUT0

G3=G3I

INCDEC-G-CI

The INCDEC-G-CI configuration performs a 1-bit increment/decrement in the G function generator, with the input on the G4 pin. The carry signal enters on the CIN pin, propagates through the G carry logic, and exits on the COUT pin. The F3 and G3 inputs indicate increment (F3=G3=1) or decrement (F3=G3=0): the increment/decrement control signal must be routed to both the F3 and G3 pins. This configuration is for the middle bit of an incrementer/decrementer, where the F function generator is reserved for another purpose, although the F3 pin is used by the carry logic.

F=

COUT0=CIN

G=(G4@G2)@~G3

COUT1=~F3*(G4+ COUT0) + F3*G4*COUT0

F4=F4I

G2=COUT0

G3=G3I

INCDEC-FG-1

The INCDEC-FG-1 configuration performs a 2-bit increment/decrement in both the F and G function generator, with the lower-order input on the F1 pin and the higher-order input on the G4 pin. The F3 and G3 inputs indicate increment (F3=G3=1) or decrement (F3=G3=0): the increment/decrement control signal must be routed to both the F3 and G3 pins. The carry-in is always active (High in increment mode and Low in decrement mode). The carry signal propagates through the F and G carry logic and exits on the COUT pin. This configuration comprises the two least significant bits of an incrementer/decrementer that is always enabled.

F=~(F1)

COUT0=F1

G=(G2@G4)@~G3

COUT=COUT1=~F3*((COUT0*~G4)+G4) + F3*(G4*COUT0)

F4=F4I

G2=COUT0

G3=G3I

FORCE-0

The FORCE-0 configuration forces the carry-out signal on the COUT pin to be 0.

COUT0=0

COUT1=0

FORCE-1

The FORCE-1 configuration forces the carry-out signal on the COUT pin to be 1.

COUT0=1

COUT1=1

FORCE-F1

The FORCE-F1 configuration forces the signal on the F1 pin to pass through to the COUT pin.

COUT0=F1

COUT1=COUT0=F1

FORCE-CI

The FORCE-CI configuration forces the signal on the CIN pin to pass through to the COUT pin.

COUT0=CIN

COUT1=COUT0=CIN

FORCE-F3-

The FORCE-F3- configuration forces the signal on the F3 pin to pass inverted to the COUT pin.

COUT0=~F3

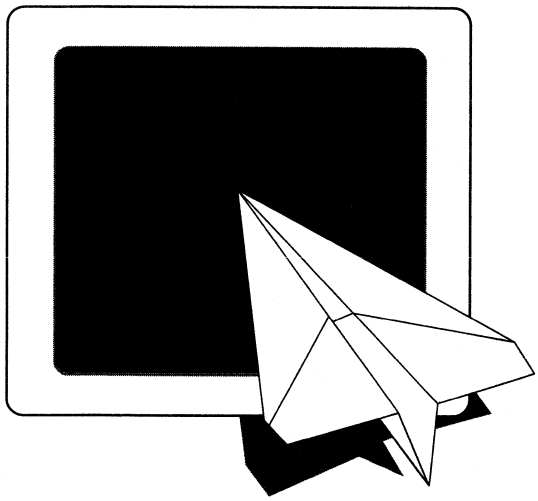
COUT1=COUT0=~F3

EXAMINE-CI

The EXAMINE-CI configuration allows the carry signal on the CIN pin to be used in the F or G function generators. This configuration forces the signal on the CIN pin to pass through to the COUT pin and is equivalent to the FORCE-CI configuration. EXAMINE-CI is provided for CLBs in which the carry logic is unused but the CIN signal is required.

COUT0=CIN

COUT1=COUT0=CIN



XACT Libraries Guide

Index

XACT Libraries Guide — April, 1994 (0401098 01)

Printed in U.S.A.

Index

G

H

I

K

L

M

N

O

P