A+PLUS Reference Guide
Version 5.0
September 1987 P25-02217-00

Altera Corporation
3525 Monroe Street
Santa Clara, CA 95051
(408) 984-2800
TELEX: 888496

# Read This First...

A+PLUS software documentation consists of two manuals:

- ***A+PLUS User Guide***
- ***A+PLUS Reference Guide***

The ***A+PLUS User Guide*** provides installation and operation information for the Altera Programmable Logic User System (A+PLUS™). It is divided into three major parts: **Introduction to A+PLUS**, **Design Entry**, and **Simulation**. If you are a first-time user, you are advised to read Sections 1, 2, and 3 in the **Introduction to A+PLUS** and then—depending on your choice of design entry method—the appropriate section in **Design Entry**. Some of the design entry packages are optional; therefore, they may not be part of your version of the manual. Each of the design entry sections contains a sample session(s) that guides you through the complete process of programming an EPLD with A+PLUS. **Simulation** contains the optional *Functional Simulator*. Following Sections 1 through 3, you will find an *Index* covering the information in the ***A+PLUS User Guide***, ***A+PLUS Reference Guide***, and the optional packages *State Machine Entry* and *Functional Simulator*.

The ***A+PLUS Reference Guide*** contains detailed information on Altera primitives, the Altera Design File format, APLUS and ADP Menus, Utilization Reports, and A+PLUS messages. Also included are several *Appendixes*, a *Glossary*, and an *Index* covering the information in the ***A+PLUS User Guide***, ***A+PLUS Reference Guide***, *State Machine Entry*, and *Functional Simulator*.

At the back of each manual, you will find a Customer Comment Form, a Problem Report Card, and a Warranty Card.

☞ A+PLUS hardware and EPLD programming are described in the ***LogicMap II*** manual.

The ***A+PLUS User Guide*** contains the following sections:

## I. Introduction to A+PLUS

*Functional Description* Gives an overview of A+PLUS that helps you to determine which design entry method is most appropriate for your particular application, and provides detailed information on the Altera Design Processor.

*Installation* Lists system components and provides specific instructions for installing A+PLUS software.

*Getting Started* Gives a brief introduction to the menus and menu functions used in running A+PLUS software.

## II. Design Entry

*Boolean Equation Entry* Describes how to enter and process a design exclusively with Boolean equations.

*State Machine Entry* (Option) Describes how to enter and process a design with state machines.

*Schematic Capture With FutureNet* (Option) Describes how to enter a schematic design with FutureNet's Schematic Designer and process it with A+PLUS.

☞ Altera's schematic capture package, LogiCaps, is described in the ***LogiCaps*** manual.

## III. Simulation

*Functional Simulator* (Option) Explains how to simulate a design with Altera's Functional Simulator.

*Index*

The ***A+PLUS Reference Guide*** contains the following sections:

*Altera Primitive Library* Describes each Altera primitive in detail, including block diagrams.

*Altera Design File Format* Explains the format of the Altera Design File.

*A+PLUS and ADP Reference* Provides detailed descriptions of the APLUS Menu; the user-controllable and automatic functions of the ADP; and DOS command line processing techniques.

*Utilization Report* Explains the Utilization Report and gives sample reports for various Altera EPLDs.

*A+PLUS Messages* Lists all error, information, and warning messages generated by A+PLUS and the ADP, their causes, and suggestions for corrective action.

| | |
|---|---|
| *Appendix A* | *Part-Specific Information* |
| *Appendix B* | *Altera Design File-to-Logic Equation File Translation* |
| *Appendix C* | *BNF Rules* |
| *Appendix D* | *JEDEC Standard File Format* |
| *Appendix E* | *Electronic Design Support Service* |
| *Appendix F* | *Altera Primitive Library (Foldouts)* |

*Glossary*

*Index*

# Manual Updates

Altera documentation is updated with Change Pages, Section Reprints, and a **READ.ME** file.

**Change Pages** are issued for minor changes to the manual. New information is identified with vertical change bars in the margins next to the changed text. In addition, the date of issue is printed at the bottom of each page.

**Section Reprints** are issued if a section requires a substantial number of changes. The date of issue is indicated at the bottom of each page.

A **READ.ME File** is provided on the A+PLUS **INSTALL** diskette. This file contains information about recent changes to the A+PLUS software that are not yet reflected in the manual.

# Printing Conventions

The following notational conventions are used throughout this manual:

| | |
|---|---|
| **Times Bold** | – A+PLUS commands, prompts, and messages<br>– All types of user input<br>– Primitive names<br>– Key names (enclosed in < >'s) |
| Times Light | – Most screen and file output |
| ***Helvetica Bold Italics*** | – References to Altera manual titles |
| *Helvetica Light Italics* | – References to sections within Altera manuals |
| ☞ | – Indicates information that requires special attention |

# Contents

## Appendixes

# Illustrations

# Tables

SECTION 1

# Altera Primitive Library

The Altera Primitive Library contains the primitives available for designing circuits in Altera EPLDs. It also includes a title block symbol that provides header information when used with a schematic design entry interface.

This section provides a detailed description of the functions and use of each primitive in the Altera Primitive Library, including schematic diagrams. Primitives are assigned mnemonic names for easy reference. (Refer also to *Appendix F* for foldout pages showing all primitives and a list of which primitives are available for each EPLD.)

For information on the high-level design tools in Altera's Standard and TTL MacroFunction libraries, refer to the manuals for the optional ***LogiCaps*** and ***MacroFunctions*** packages.

# Introduction

Altera primitives consist of the following types:

- Input Primitives
- Logic Primitives
- Equation (EQN) Primitives
- I/O Primitives
- Bus I/O Primitives

The following pages list the primitives of each category in alphabetical order. Each primitive description is presented on a separate page and consists of a block diagram that shows pin assignments, followed by detailed information as outlined here:

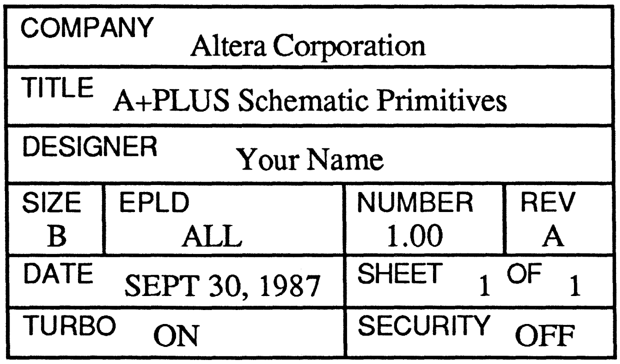
| | |
|---|---|
| Name: | Gives the name and definition of the primitive. (Logic primitives have two names: one with the number of inputs appended for the schematic symbol and one for the Altera Design File.) |
| ADF Syntax: | Shows the appearance of the primitive statement in the Altera Design File format. (Refer also to *Altera Design File Format*). |
| Description: | Defines the input and output signal(s) for the primitive. |
| EPLDs: | Indicates which EPLDs support the primitive. |
| Note: | If necessary, further information is given in notes. |

☞ The block diagrams for Altera input, equation, logic and I/O primitives are shown as they appear on screen when the SN (Symbol Numbers) command is used with the LogiCaps schematic capture package. Some of these primitives may appear in a slightly different form if you are using the FutureNet DASH Schematic Designer.

## Title Block

The Title Block allows you to store information about your circuit design. Each schematic diagram should be documented with an Altera Title Block. Ordinarily, you load and fill out the title block before starting the schematic drawing, then arrange the other symbols within the drawing. The appropriate schematic capture converter module uses the information from the title block to process the schematic. Figure 1-1 shows an Altera Title Block.

For instructions on how to enter the title block into a schematic design, see the individual sections on schematic design entry in the ***A+PLUS User Guide***.



| | | | |
|---|---|---|---|
| COMPANY Altera Corporation | | | |
| TITLE A+PLUS Schematic Primitives | | | |
| DESIGNER Your Name | | | |
| SIZE B | EPLD ALL | NUMBER 1.00 | REV A |
| DATE SEPT 30, 1987 | | SHEET 1 OF 1 | |
| TURBO ON | | SECURITY OFF | |

**Figure 1-1. Altera Title Block**

The data fields in the title block are as follows:

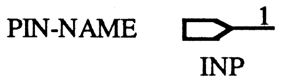
| | |
|---|---|
| COMPANY: | Company name |
| TITLE: | Title of the drawing |

| | |
|---|---|
| DESIGNER: | Your name |
| SIZE: | Drawing size (**A**, **B**, **C**, **D**, or **E**) |
| EPLD: | Target device number, i.e., one of the following: **'EP310'** \| **'EP310D'** \| **'EP320'** \| **'EP320D'** \| **'EP600'** \| **'EP600D'** \| **'EP600J'** \| **'EP610'** \| **'EP610D'** \| **'EP610J'** \| **'EP900'** \|**'EP900D'** \| **'EP900J'** \| **'EP910'** \| **'EP910D'** \| **'EP910J'** \| **'EP1210'** \| **'EP1210D'** \| **'EP1210J'** \| **'EPB1400'** \| **'EPB1400D'** \| **'EPB1400J'** \| **'EP1800'** \| **'EP1800J'** \| **'EP1800G'** \| **'AUTO'** \| **'MACRO'** |
| ☞ | **MACRO** is used only with the **ADLIB** option (available with LogiCaps). |
| NUMBER: | Drawing number |
| REV: | Revision identification |
| DATE: | Date of schematic entry |
| SHEET: | Page identification |
| TURBO: | **OFF** turns the Turbo-Bit option off (low power). **ON** (the default) turns the Turbo-Bit option on (high speed). |
| ☞ | The BUSTER (EPB1400) and EP310 parts do not support the Turbo-Bit option. LogicMap will ignore any entry made in this field. |
| SECURITY: | **ON** turns on the Security Bit, which prevents the device from being interrogated or inadvertently reprogrammed. **OFF** turns the Security Bit option off. |

## Input Primitives

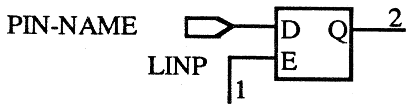
The following pages provide detailed descriptions of the Altera input primitives.

# INP



| | |
|---|---|
| Name: | **INP** (Primary Input) |
| ADF Syntax: | **Out** = **INP(In)** |
| Description: | Outputs:<br>**Out(1)** = output to logic array<br><br>Inputs:<br>**In** = pin input |
| EPLDs: | All |

## LINP



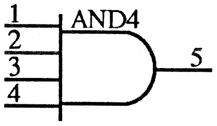
| | |
|---|---|
| Name: | **LINP** (Latched Primary Input) |
| ADF Syntax: | **Out** = **LINP(In,Ile)** |
| Description: | Outputs:<br>**Out(2)** = latched output to logic array<br><br>Inputs:<br>**In** = EPLD pin input<br>**Ile(1)** = input latch enable (clock input) |
| EPLDs: | EP1210 |

## Logic Primitives

The following pages provide detailed descriptions of the Altera logic primitives. The schematic symbol name for these primitives differs from the Altera Design File primitive name: the symbol name has a suffix indicating the number of inputs (e.g., **AND6**). Note that for schematic capture entry, the multiple input primitives are available in 2, 3, 4, 6, 8, and 12 input versions.

In addition, the **AND** gates are available as "bubble version" **BNOR** gates; **NAND** gates have **BOR**-gate equivalents; **OR** gates have **BNAND**-gate equivalents; and **NOR** gates have **BAND**-gate equivalents.

## AND (BNOR)



| | |
|---|---|
| Schematic Name: | **AND2, AND3, AND4, AND6, AND8, AND12** |
| ADF Name: | **AND(In1, ..., In12)** |
| ADF Syntax: | **Out = AND(In1, In2, ..., In12)** |

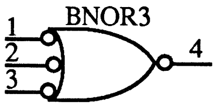
Description:

Outputs:
**Out** = logical AND of inputs

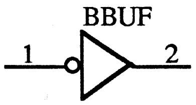
Inputs:
**In1, In2, ..., In12** = 2 to 12 inputs

EPLDs: All

Also available: **BNOR** gates ("bubble **NOR** gates"), which are **AND**-gate equivalents. For example:

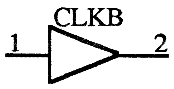


## BBUF



| | |
|---|---|
| Name: | **BBUF** (Bubble Buffer) |
| ADF Syntax: | **Out** = **BBUF(In1)** |
| Description: | Outputs: **Out** = inversion of input |
| | Inputs: **In1** = 1 input only |
| EPLDs: | All |

## CLKB

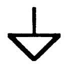


| | |
|---|---|
| Name: | **CLKB** (Asynchronous Clock Buffer) |
| ADF Syntax: | **Out** = **CLKB(In)** |
| Description: | Outputs:<br>**Out** = output to register clock<br>Inputs:<br>**In** = 1 input only |
| EPLDs: | EP600, EP610, EP900, EP910, EPB1400, EP1800 |
| Notes: | 1. **CLKB** may be used to request asynchronous clocking of a register when the clock is driven directly from a pin. |

2. **CLKB** is *optional* when the input to a register clock is driven by logic. When **CLKBs** are fed by logic primitives, the resulting logic must be reducible to one product term, except in the EPB1400 ("BUSTER"), which permits two product terms for the Clock (**Clk**), Output Latch Enable (**Ole**), Output Enable (**Oe**), Read Enable (**Re**), and Write Enable (**We**) inputs.

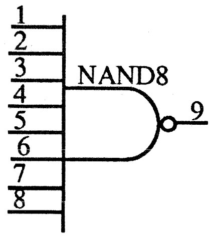
3. In the EP600, EP610, EP900, EP910, and EP1800, the asynchronous clock and Output Enable (**Oe**) inputs share a single product term. When an asynchronous clock is used for the EP600, EP610, EP900, EP910, and the *local* macrocells of the EP1800, **Oe** must be permanently enabled (VCC). For EP1800 *global* and *all* EPB1400 macrocells, **Oe** may be set to **GND** or **VCC** when asynchronous clocking is used.

## GND

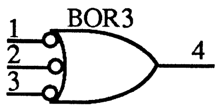


| | |
|---|---|
| Name: | **GND** |
| ADF Syntax: | **GND** |
| Description: | The only purpose of this symbol is to assign a node to ground. |
| EPLDs: | All |
| Notes: | 1. Node name automatically assigned **GND**. |
| | 2. Does not exist in the P-CAD schematic editor (PC-CAPS). You must use the node name **LO** or **GND**. In the P-CAD Simulator (PC-LOGS) you may use only the name **LO**. |

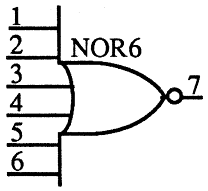
## NAND (BOR)



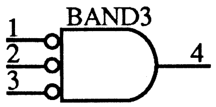
|  |  |
|---|---|
| Schematic Name: | **NAND2, NAND3, NAND4, NAND6, NAND8, NAND12** |
| ADF Name: | **NAND(In1, ..., In12)** |
| ADF Syntax: | **Out = NAND(In1, In2, ..., In12)** |
| Description: | Outputs: **Out** = logical NAND of inputs<br>Inputs: **In1, In2, ..., In12** = 2 to 12 inputs |
| EPLDs: | All |
| Also available: | **BOR** gates ("bubble **OR** gates"), which are **NAND**-gate equivalents. For example: |



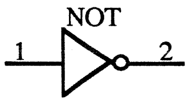
## NOR (BAND)



| | |
|---|---|
| Schematic Name: | **NOR2, NOR3, NOR4, NOR6, NOR8, NOR12** |
| ADF Name: | **NOR(In1, ..., In12)** |
| ADF Syntax: | **Out = NOR(In1, In2, ..., In12)** |
| Description: | Outputs:<br>**Out** = logical NOR of inputs<br><br>Inputs:<br>**In1, In2, ..., In12** = 2 to 12 inputs |
| EPLDs: | All |
| Also available: | **BAND** gates ("bubble **AND** gates"), which are **NOR**-gate equivalents. For example: |

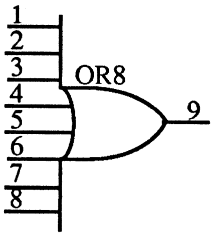


## NOT

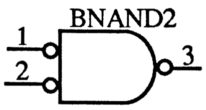


| | |
|---|---|
| Name: | **NOT** (Logical Inversion) |
| ADF Syntax: | **Out** = **NOT(In1)** |
| Description: | Outputs:<br>**Out** = inversion of input<br><br>Inputs:<br>**In1** = 1 input only |
| EPLDs: | All |

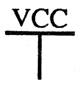
## OR (BNAND)



| | |
|---|---|
| Schematic Name: | **OR2, OR3, OR4, OR6, OR8, OR12** |
| ADF Name: | **OR(In1, ..., In12)** |
| ADF Syntax: | **Out = OR(In1, In2, ..., In12)** |
| Description: | Outputs:<br>**Out** = logical OR of inputs<br><br>Inputs:<br>**In1, In2, ..., In12** = 2 to 12 inputs |
| EPLDs: | All |
| Also available: | **BNAND** gates ("bubble **NAND** gates"), which are **OR**-gate equivalents. For example: |

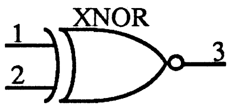


## VCC



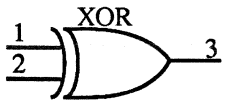
| | |
|---|---|
| Name: | **VCC** |
| ADF Syntax: | **VCC** |
| Description: | The only purpose of this symbol is to assign a node to VCC. |
| EPLDs: | All |
| Notes: | 1. Node name automatically assigned **VCC**. |
| | 2. Does not exist in the P-CAD schematic editor (PC-CAPS). You must name the node **VCC** or **HI**. (In the P-CAD Simulator [PC-LOGS] you may use only the name **HI**.) |

## XNOR



| | |
|---|---|
| Name: | **XNOR** (Logical Exclusive NOR) |
| ADF Syntax: | **Out = XNOR(In1, In2)** |
| Description: | Outputs:<br>**Out** = exclusive NOR of inputs<br><br>Inputs:<br>**In1, In2** = 2 inputs only |
| EPLDs: | All |
| Note: | Converted into equivalent logic function using two **NOT**, two **AND**, and one **NOR** primitive by the Translator. |

## XOR



|  |  |
|---|---|
| Name: | **XOR** (Logical Exclusive OR) |
| ADF Syntax: | **Out = XOR(In1, In2)** |
| Description: | Outputs:<br>**Out** = exclusive OR of inputs<br><br>Inputs:<br>**In1, In2** = 2 inputs only |
| EPLDs: | All |
| Note: | Converted by the Translator into an equivalent logic function using two **NOT**, two **AND**, and one **OR** primitive. See Figure 1-2. |

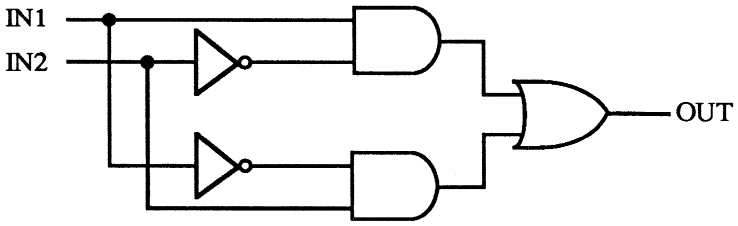
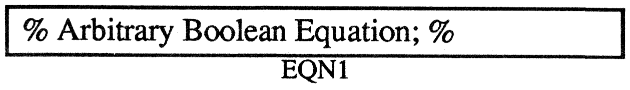


**Figure 1-2. XOR Equivalent Circuit**

## Equation (EQN) Primitives

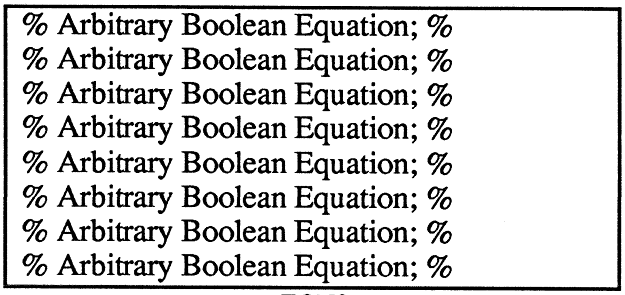
The following pages provide detailed descriptions of the Altera equation primitives, which are used for entering Boolean equations with schematic capture programs. These two primitives, labeled EQN1 and EQN8, can accommodate one and eight arbitrary Boolean expressions, respectively. (Refer to the documentation for individual schematic capture programs for how to create EQN primitives with any number of equations.)

## EQN1



| | |
|---|---|
| Schematic Name: | **EQN1** |
| ADF Name: | **EQN** |
| ADF Syntax: | **Out** = arbitrary Boolean expression; (must be terminated with a semicolon) |
| Description: | This equation box is intended to accommodate one arbitrary Boolean expression. |
| EPLDs: | All |

## EQN8


% Arbitrary Boolean Equation; %
% Arbitrary Boolean Equation; %
% Arbitrary Boolean Equation; %
% Arbitrary Boolean Equation; %
% Arbitrary Boolean Equation; %
% Arbitrary Boolean Equation; %
% Arbitrary Boolean Equation; %
% Arbitrary Boolean Equation; %

EQN8

| | |
|---|---|
| Schematic Name: | **EQN8** |
| ADF Name: | **EQN** |
| ADF Syntax: | **Out** = arbitrary Boolean expression;<br>(must be terminated with a semicolon) |
| Description: | This equation box is intended to accommodate eight arbitrary Boolean expressions. |
| EPLDs: | All |

# I/O Primitives

The following pages provide detailed descriptions of the Altera I/O primitives.

The following rules apply to I/O primitives:

1. If the Output Enable (**Oe**) is unconnected, it defaults to **VCC** (output enabled).

2. The default for an unconnected Clear (**C**) is **GND** (clear disabled).

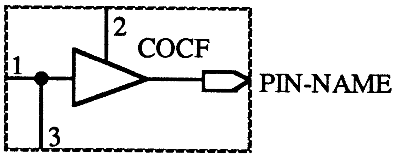
3. The default for an unconnected Preset (**P**) is **GND** (preset disabled).

4. The Clock (**Clk**) must be assigned under all circumstances.

5. The letters that comprise the mnemonic names of I/O primitives have the following meanings:

| | | |
|---|---|---|
| **C** | = | Combinatorial |
| **F** | = | Feedback |
| **I** | = | I/O |
| **J** | = | JK-type flipflop |
| **L** | = | Latched |
| **N** | = | No |
| **O** | = | Output |
| **R** | = | (Registered) D-type flipflop |
| **S** | = | SR-type flipflop |
| **T** | = | T-type flipflop |

## COCF



| | |
|---|---|
| Name: | **COCF** (Combinatorial Output, Combinatorial Feedback) |
| ADF Syntax: | **Out,Fbk** = **COCF(In,Oe)** |
| Description: | Outputs: |

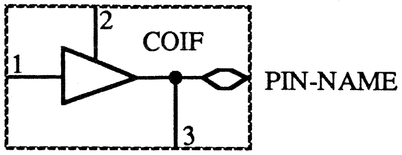
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(3)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Oe(2)** | = | 3-state buffer output enable input |

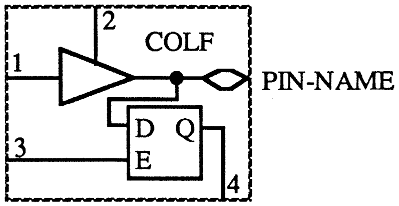
EPLDs: EP310, EPB1400, EP1800

## COIF



| | |
|---|---|
| Name: | **COIF** (Combinatorial Output, I/O Feedback) |
| ADF Syntax: | **Out,Fbk** = **COIF(In,Oe)** |
| Description: | Outputs: |
| | **Out** = output to EPLD pin |
| | **Fbk(3)** = feedback to logic array |
| | Inputs: |
| | **In(1)** = input from logic array |
| | **Oe(2)** = 3-state buffer output enable input |
| EPLDs: | All |

## COLF



| | | | |
|---|---|---|---|
| Name: | **COLF** (Combinatorial Output, Latched Feedback) | | |
| ADF Syntax: | **Out,Fbk** | = | **COLF(In,Oe,Ile)** |

Description: Outputs:

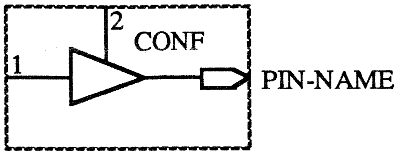
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(4)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Oe(2)** | = | 3-state buffer output enable input |
| **Ile(3)** | = | latch clock input |

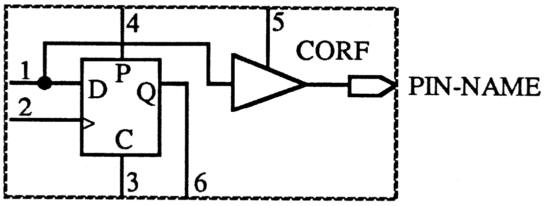
EPLDs: EP1210

## CONF



| | |
|---|---|
| Name: | **CONF** (Combinatorial Output, No Feedback) |
| ADF Syntax: | **Out** = **CONF(In,Oe)** |
| Description: | Outputs:<br>**Out** = output to EPLD pin<br><br>Inputs:<br>**In(1)** = input from logic array<br>**Oe(2)** = 3-state buffer output enable input |
| EPLDs: | All |

## CORF



| | |
|---|---|
| Name: | **CORF** (Combinatorial Output, Registered Feedback) |
| ADF Syntax: | **Out,Fbk = CORF(In,Clk,C,P,Oe)** |

Description: Outputs:

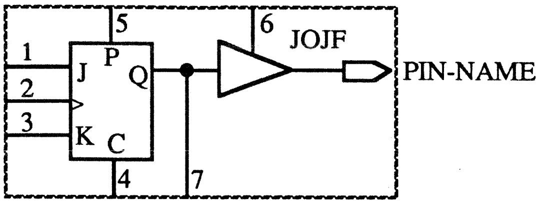
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **C(3)** | = | clear input |
| **P(4)** | = | preset input (not used in EP1210 ) |
| **Oe(5)** | = | 3-state buffer output enable input |

EPLDs: EP310, EP1210

## JOJF



| | |
|---|---|
| Name: | **JOJF** (JK Output, JK Feedback) |
| ADF Syntax: | **Out,Fbk = JOJF(JIn,Clk,KIn,C,P,Oe)** |

Description:

Outputs:

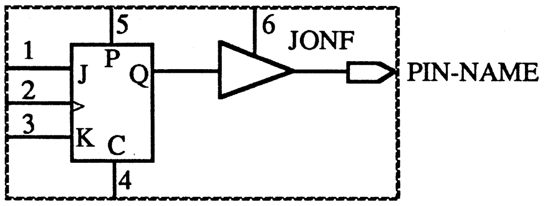
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(7)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **JIn(1)** | = | input from logic array |
| **Clk(2)** | = | register clock |
| **KIn(3)** | = | input from logic array |
| **C(4)** | = | clear input |
| **P(5)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |
| **Oe(6)** | = | 3-state buffer output enable input |

EPLDs: EP600, EP610, EP900, EP910, EPB1400, EP1800

# JONF



| | |
|---|---|
| Name: | **JONF** (JK Output, NO Feedback) |
| ADF Syntax: | **Out = JONF(JIn,Clk,KIn,C,P,Oe)** |

Description:

Outputs:

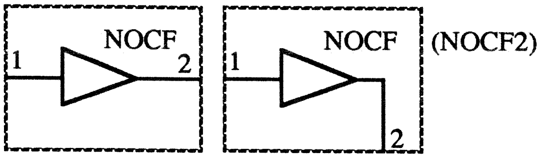
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |

Inputs:

| | | |
|---|---|---|
| **JIn(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **KIn(3)** | = | input from logic array |
| **C(4)** | = | clear input |
| **P(5)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |
| **Oe(6)** | = | 3-state buffer output enable input |

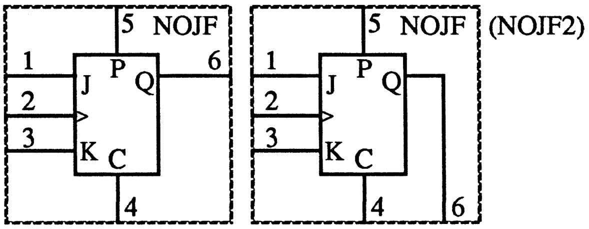
EPLDs: EP600, EP610, EP900, EP910, EPB1400, EP1800

## NOCF



| | |
|---|---|
| Name: | **NOCF** (No Output, Combinatorial Feedback) |
| ADF Syntax: | **Fbk = NOCF(In)** |
| Description: | Outputs:<br>**Fbk(2)** = feedback to logic array<br><br>Inputs:<br>**In(1)** = input from logic array |
| EPLDs: | EP310, EP1210, EPB1400, EP1800 |
| Note: | 1. In the EP1210, **NOCF** can only be used on the four macrocells without I/O pins (buried macrocells). If combinatorial feedback is needed on a macrocell with an I/O pin, the **NOCF** is promoted to a **COIF** primitive.<br><br>2. If used in the EP320, EP600, EP610, EP900, or EP910, the **NOCF** is promoted to a **COIF**. |

## NOJF



|  |  |
|---|---|
| Name: | **NOJF** (No Output, JK Feedback) |
| ADF Syntax: | **Fbk = NOJF(JIn,Clk,KIn,C,P)** |
| Description: | Outputs: |

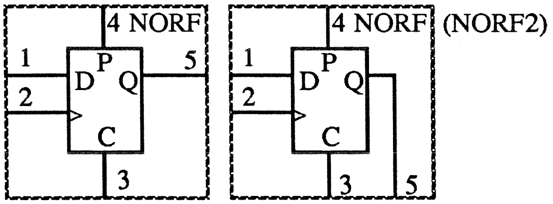
| | | |
|---|---|---|
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **JIn(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **KIn(3)** | = | input from logic array |
| **C(4)** | = | clear input |
| **P(5)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |

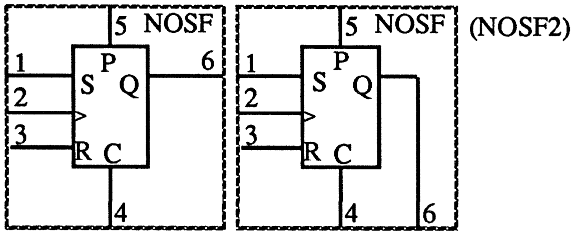
EPLDs: EP600, EP610, EP900, EP910, EPB1400, EP1800

## NORF



|  |  |  |  |
|---|---|---|---|
| Name: | **NORF** (No Output, Registered Feedback) | | |
| ADF Syntax: | **Fbk** | = | **NORF(In,Clk,C,P)** |
| Description: | Outputs: | | |
| | **Fbk(5)** | = | feedback to logic array |
| | Inputs: | | |
| | **In(1)** | = | input from logic array |
| | **Clk(2)** | = | register clock input |
| | **C(3)** | = | clear input |
| | **P(4)** | = | preset input (not used in EP600, EP610, EP900, EP910, EP1210, EPB1400, EP1800) |
| EPLDs: | All | | |

## NOSF



| | |
|---|---|
| Name: | **NOSF** (No Output, SR Feedback) |
| ADF Syntax: | **Fbk = NOSF(SIn,Clk,RIn,C,P)** |

Description:

Outputs:

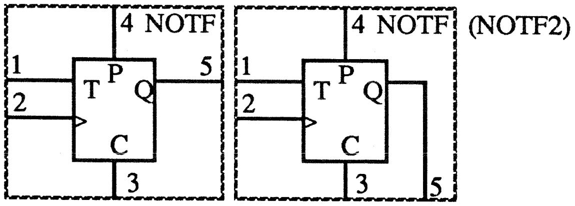
| | | |
|---|---|---|
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **SIn(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **RIn(3)** | = | input from logic array |
| **C(4)** | = | clear input |
| **P(5)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |

EPLDs: EP600, EP610, EP900, EP910, EPB1400, EP1800

## NOTF



| | |
|---|---|
| Name: | **NOTF** (No Output, T Feedback) |
| ADF Syntax: | **Fbk = NOTF(In,Clk,C,P)** |

Description: Outputs:

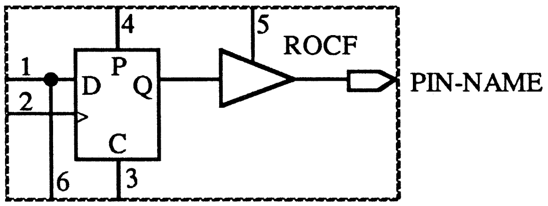
| | | |
|---|---|---|
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **C(4)** | = | clear input |
| **P(5)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |

EPLDs: EP600, EP610, EP900, EP910, EPB1400, EP1800

## ROCF



| | |
|---|---|
| Name: | **ROCF** (Registered Q Output, Combinatorial Feedback) |
| ADF Syntax: | **Out,Fbk = ROCF(In,Clk,C,P,Oe)** |
| Description: | Outputs: |

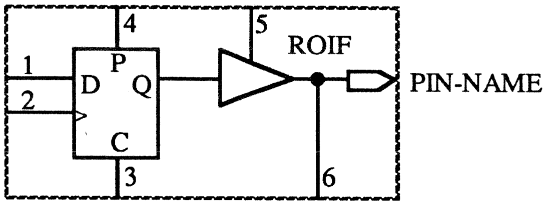
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock |
| **C(3)** | = | clear input |
| **P(4)** | = | preset input |
| **Oe(5)** | = | 3-state buffer output enable input |

EPLDs: EP310

## ROIF



| | |
|---|---|
| Name: | **ROIF** (Registered Q Output, I/O Feedback) |
| ADF Syntax: | **Out,Fbk = ROIF(In,Clk,C,P,Oe)** |

Description:

Outputs:

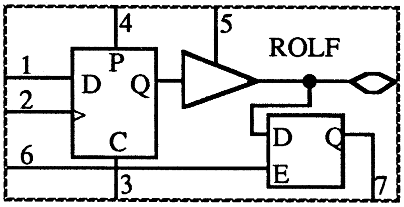
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock |
| **C(3)** | = | clear input |
| **P(4)** | = | preset input (not used in EP600, EP610, EP900, EP910, EP1210, EPB1400, EP1800 ) |
| **Oe(5)** | = | 3-state buffer output enable input |

EPLDs: EP310, EP600, EP610, EP900, EP910, EP1210, EPB1400, EP1800

## ROLF



| | |
|---|---|
| Name: | **ROLF** (Registered Q Output, Latched Feedback) |
| ADF Syntax: | **Out,Fbk = ROLF(In,Clk,C,P,Oe,Ile)** |

Description:

Outputs:

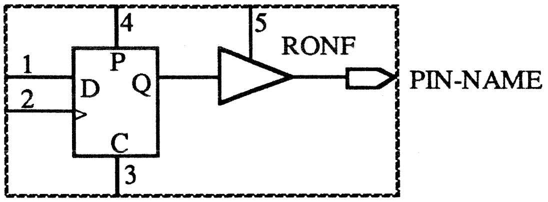
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(7)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **C(3)** | = | clear input |
| **P(4)** | = | preset input (not used in EP1210 ) |
| **Oe(5)** | = | 3-state buffer output enable input |
| **Ile(6)** | = | latch clock input |

EPLDs: EP1210

## RONF



| | |
|---|---|
| Name: | **RONF** (Registered Q Output, No Feedback) |
| ADF Syntax: | **Out = RONF(In,Clk,C,P,Oe)** |

Description:

Outputs:

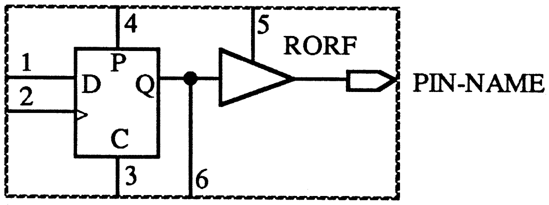
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **C(3)** | = | clear input |
| **P(4)** | = | preset input (not used in EP320, EP600, EP610, EP900, EP910, EP1210, EPB1400, EP1800 ) |
| **Oe(5)** | = | 3-state buffer output enable input |

EPLDs: All

## RORF



| | |
|---|---|
| Name: | **RORF** (Registered Q Output, Registered Feedback) |
| ADF Syntax: | **Out,Fbk = RORF(In,Clk,C,P,Oe)** |

Description:

Outputs:

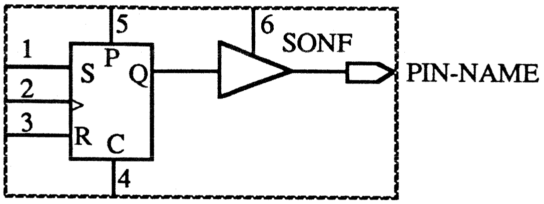
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **C(3)** | = | clear input |
| **P(4)** | = | preset input (not used in EP320, EP600, EP610, EP900, EP910, EP1210, EPB1400, EP1800 ) |
| **Oe(5)** | = | 3-state buffer output enable input |

EPLDs: All

## SONF



| | |
|---|---|
| Name: | **SONF** (SR Output, No Feedback) |
| ADF Syntax: | **Out = SONF(SIn,Clk,RIn,C,P,Oe)** |

Description:

Outputs:

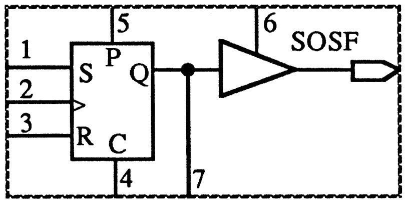
| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |

Inputs:

| | | |
|---|---|---|
| **SIn(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **RIn(3)** | = | input from logic array |
| **C(4)** | = | clear input |
| **P(5)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |
| **Oe(6)** | = | 3-state buffer output enable input |

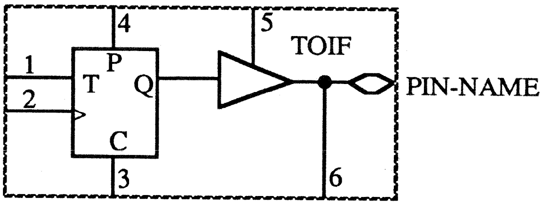
EPLDs: EP600, EP610, EP900, EP910, EPB1400, EP1800

## SOSF



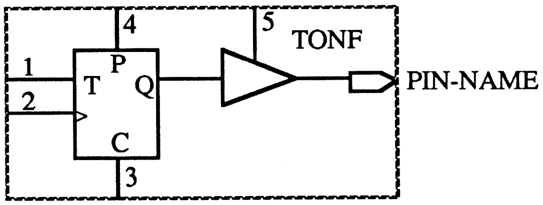
| | | | |
|---|---|---|---|
| Name: | **SOSF** (SR Output, SR Feedback) | | |
| ADF Syntax: | **Out,Fbk = SOSF(SIn,Clk,RIn,C,P,Oe)** | | |
| Description: | Outputs: | | |
| | **Out** | = | output to EPLD pin |
| | **Fbk(7)** | = | feedback to logic array |
| | Inputs: | | |
| | **SIn(1)** | = | input from logic array |
| | **Clk(2)** | = | register clock input |
| | **RIn(3)** | = | input from logic array |
| | **C(4)** | = | clear input |
| | **P(5)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |
| | **Oe(6)** | = | 3-state buffer output enable input |
| EPLDs: | EP600, EP610, EP900, EP910, EPB1400, EP1800 | | |

## TOIF



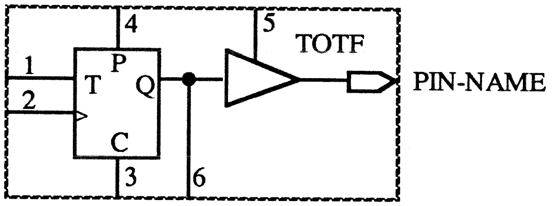
| | |
|---|---|
| Name: | **TOIF** (T Output, I/O Feedback) |
| ADF Syntax: | **Out,Fbk = TOIF(In,Clk,C,P,Oe)** |
| Description: | Outputs:<br>**Out** = output to EPLD pin<br>**Fbk(6)** = feedback to logic array<br><br>Inputs:<br>**In(1)** = input from logic array<br>**Clk(2)** = register clock input<br>**C(3)** = clear input<br>**P(4)** = preset input (not used in EP600, EP610, EP900, EP910,EPB1400, EP1800)<br>**Oe(5)** = 3-state buffer output enable input |
| EPLDs: | EP600, EP610, EP900, EP910, EPB1400, EP1800 |

## TONF



| | |
|---|---|
| Name: | **TONF** (T Output, No Feedback) |
| ADF Syntax: | **Out** = **TONF(In,Clk,C,P,Oe)** |
| Description: | Outputs: |
| | **Out** = output to EPLD pin |
| | Inputs: |
| | **In(1)** = input from logic array |
| | **Clk(2)** = register clock input |
| | **C(3)** = clear input |
| | **P(4)** = preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |
| | **Oe(5)** = 3-state buffer output enable input |
| EPLDs: | EP600, EP610, EP900, EP910, EPB1400, EP1800 |

## TOTF



| | |
|---|---|
| Name: | **TOTF** (T Output, T Feedback) |
| ADF Syntax: | **Out,Fbk** = **TOTF(In,Clk,C,P,Oe)** |

Description:

Outputs:

| | | |
|---|---|---|
| **Out** | = | output to EPLD pin |
| **Fbk(6)** | = | feedback to logic array |

Inputs:

| | | |
|---|---|---|
| **In(1)** | = | input from logic array |
| **Clk(2)** | = | register clock input |
| **C(3)** | = | clear input |
| **P(4)** | = | preset input (not used in EP600, EP610, EP900, EP910, EPB1400, EP1800) |
| **Oe(5)** | = | 3-state buffer output enable input |

EPLDs: EP600, EP610, EP900, EP910, EPB1400, EP1800

## Bus I/O Primitives

The following pages provide detailed descriptions of the Bus I/O primitives, which are available for use with the EPB1400 ("BUSTER").

The following rules apply to Bus I/O primitives:

1. Each BUSTER (EPB1400) part may use 0 or 1 **BUSX** primitive; 0, 1, or 2 **LBUSO** primitives; and 0, 1, or 2 primitives selected from the following group: **LBUSI**, **LINP8**, **RBUSI** and **RINP8**.

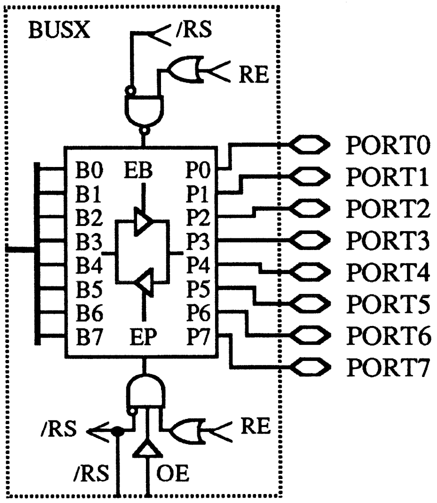
2. The Clock (**Clk**) input to the **LBUSO** primitive is optional. If it is connected, it must be connected directly to a pin or **VCC** (latch enabled). When the **Clk** input is connected, the value of an unconnected Output Latch Enable (**Ole**) input will default to **VCC**. If the **Clk** is left unconnected, logic must be connected to the **Ole**, which will have full control of the latch. If either input is left unconnected, you must retain the comma that delimits the field.

3. The Read Strobe (**Rs**) input to the **BUSX** primitive is optional. This signal is predefined as an active low input, therefore, it is not necessary to invert the signal to implement active low logic. If the **Rs** is connected, it must be connected directly to the dedicated pin (/**CRS**) or to **GND**. When the **Rs** input is connected, the value of an unconnected Output Enable (**Oe**) input will default to **VCC**, allowing the bus port to drive the internal bus. If the **Rs** is left unconnected, logic must be connected to the **Oe**, which will have full control of the bus transceiver. If either input is left unconnected, you must retain the comma that delimits the field.

4. The Write Strobe (**Ws**) input to the **LBUSI**, **LINP8**, **RBUSI**, and **RINP8** primitives is optional. This signal is predefined as an active low input, therefore, it is not necessary to invert the signal to implement active low logic. If the **Ws** is connected, it must be connected directly to the dedicated pin (/**CWS**) or **GND**. When the **Ws** input is connected, the value of an unconnected Write Enable (**We**) input will default to **VCC** (latch/register enabled). If the **Ws** is left unconnected, logic must be connected to the **We**, which will have full control of the latch/register. If either input is left unconnected, you must retain the comma that delimits the field.

5. The logic feeding the Write Enable (**We**), Read Enable (**Re**), Output Latch Enable (**Ole**), and Output Enable (**Oe**) inputs may contain up to two product terms.

## BUSX



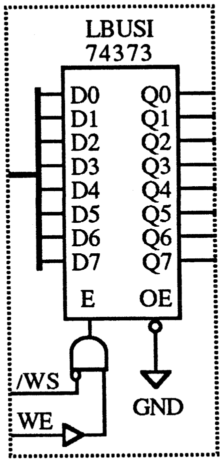
| | |
|---|---|
| Name: | **BUSX** (Bus Transceiver) |
| ADF Syntax: | **Out0, Out1, Out2, Out3, Out4, Out5, Out6, Out7 = BUSX(Ibus, Rs, Oe)** |
| Description: | Outputs:<br>**Out(0-7)** = EPLD port pin outputs (PORT0–7)<br><br>Inputs:<br>**Ibus** = (internal bus) output from logic array<br>**Rs** = read strobe input (active low)<br>**Oe** = 3-state buffer output enable input |
| EPLDs: | EPB1400 (BUSTER) |
| Notes: | 1. The **OR** gates in this primitive are fed by the Read Enable (**Re**) output of up to two **LBUSO** primitives. |

2. The Read Strobe (**Rs**) input to the **BUSX** primitive is optional. This signal is predefined as an active low input, therefore, it is not necessary to invert the signal to implement active low logic. If the **Rs** is connected, it must be connected directly to the dedicated pin (/**CRS**) or to **GND**. When the **Rs** input is connected, the value of an unconnected Output Enable (**Oe**) input will default to **VCC**, allowing the bus port to drive the internal bus. If the **Rs** is left unconnected, logic must be connected to the **Oe**, which will have full control of the bus transceiver. If either input is left unconnected, you must retain the comma that delimits the field.

3. The Read Strobe (**Rs**) output of this primitive is the **Rs** used by an **LBUSO** primitive.

4. The Read Enable (**Re**) input of this primitive is the supplied by an **LBUSO** primitive.

## LBUSI

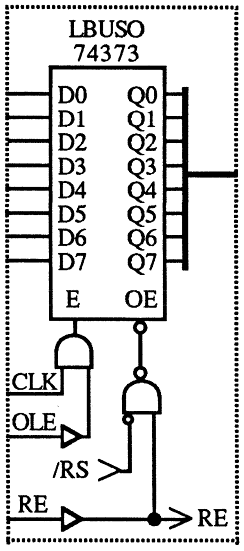


| | |
|---|---|
| Name: | **LBUSI** (Latched Bus Input to Logic) |
| ADF Syntax: | **Out0, Out1, Out2, Out3, Out4, Out5, Out6, Out7 = LBUSI(Ibus, Ws, We)** |
| Description: | Outputs:<br>**Out(0-7)** = output to logic array (Q0-Q7)<br><br>Inputs:<br>**Ibus** = internal bus<br>**Ws** = write strobe input (active low)<br>**We** = write enable input |
| EPLDs: | EPB1400 (BUSTER) |
| Notes: | 1. The Write Strobe (**Ws**) input to this primitive is optional. This signal is predefined as an active low input, therefore, it is not necessary to invert the signal to implement active low logic. If the **Ws** is connected, it must be connected directly to the dedicated pin (/**CWS**) or **GND**. When the **Ws** input is connected, the value of an unconnected Write Enable (**We**) input will default to **VCC** (latch |

enabled). If the **Ws** is left unconnected, logic must be connected to the **We**, which will have full control of the latch. If either input is left unconnected, you must retain the comma that delimits the field.

2. The logic feeding the Write Enable (**We**) input may contain up to 2 product terms.

## LBUSO



| | |
|---|---|
| Name: | **LBUSO** (Latched Bus Output from Logic) |
| ADF Syntax: | **Ibus = LBUSO(In0, In1, In2, In3, In4, In5, In6, In7, Clk, Ole, Re)** |

Description: Outputs:

| | | |
|---|---|---|
| **Ibus** | = | Internal bus |

Inputs:

| | | |
|---|---|---|
| **In(0-7)** | = | EPLD pin inputs or macrocell feedbacks (D0-D7) |
| **Clk** | = | clock input |
| **Ole** | = | output latch enable |
| **Re** | = | read enable |

EPLDs: EPB1400 (BUSTER)

Notes: 1. The Clock (**Clk**) input to the **LBUSO** primitive is optional. If it is connected, it must be connected directly to a pin or **VCC** (latch enabled). When the

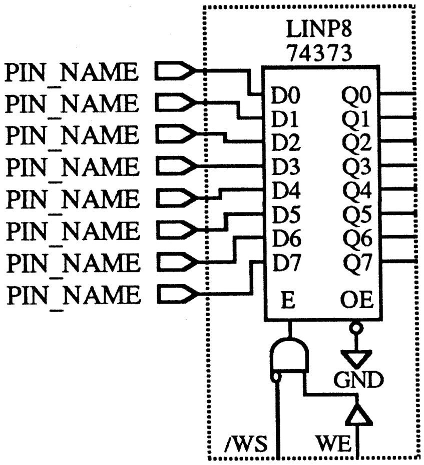
**Clk** input is connected, the value of an unconnected Output Latch Enable (**Ole**) input will default to **VCC**. If the **Clk** is left unconnected, logic must be connected to the **Ole**, which will have full control of the latch. If either input is left unconnected, you must retain the comma that delimits the field.

2. The logic feeding the Output Latch Enable (**Ole**) and Read Enable (**Re**) inputs may contain up to 2 product terms.

3. The Read Strobe (**Rs**) input to this primitive is supplied by the **BUSX** primitive.

4. The Read Enable (**Re**) output of this primitive is the **Re** used by a **BUSX** primitive.

## LINP8



| | |
|---|---|
| Name: | **LINP8** (8-Bit Latched Pin Input to Logic) |
| ADF Syntax: | **Out1, Out2, Out3, Out4, Out5, Out6, Out7 = LINP8(In0, In1, In2, In3, In4, In5, In6, In7, Ws, We)** |

Description:

Outputs:

| | | |
|---|---|---|
| **Out(0-7)** | = | outputs to logic array (Q0-Q7) |

Inputs:

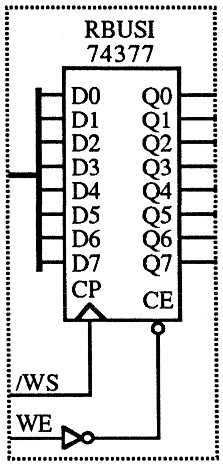
| | | |
|---|---|---|
| **In(0–7)** | = | EPLD pin inputs |
| **Ws** | = | write strobe input (active low) |
| **We** | = | write enable input |

EPLDs: EPB1400 (BUSTER)

Notes: 1. The Write Strobe (**Ws**) input to this primitive is optional. This signal is predefined as an active low input, therefore, it is not necessary to invert the signal to implement active low logic. If the **Ws** is connected, it must be connected directly to the dedicated pin (/**CWS**) or **GND**. When the **Ws** input is connected, the value of an unconnected Write Enable (**We**) input will default to **VCC** (latch enabled). If the **Ws** is left unconnected, logic must be connected to the **We**, which will have full control of the latch. If either input is left unconnected, you must retain the comma that delimits the field.

2. The logic feeding the Write Enable (**We**) input may contain up to 2 product terms.

## RBUSI

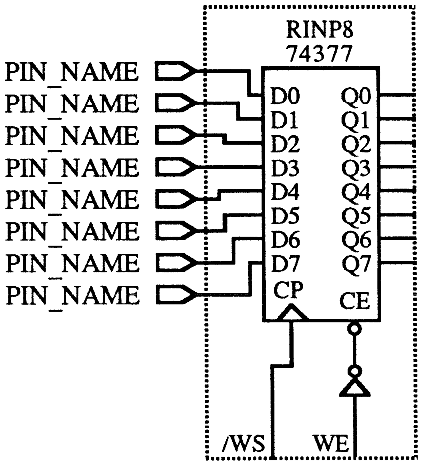


| | |
|---|---|
| Name: | **RBUSI** (Registered Bus Input to Logic) |
| ADF Syntax: | **Out0, Out1, Out2, Out3, Out4, Out5, Out6, Out7 = RBUSI(Ibus, Ws, We)** |
| Description: | Outputs:<br>**Out(0-7)** = output to logic array (Q0-Q7)<br><br>Inputs:<br>**Ibus** = internal bus<br>**Ws** = write strobe input (active low)<br>**We** = write enable input |
| EPLDs: | EPB1400 ("BUSTER") |
| Notes: | 1. The Write Strobe (**Ws**) input to this primitive is optional. This signal is predefined as an active low input, therefore, it is not necessary to invert the signal to implement active low logic. If the **Ws** is connected, it must be connected directly to the dedicated pin (**/CWS**) or **GND**. When the **Ws** input is connected, the value of an unconnected Write Enable (**We**) input will default to **VCC** (register |

enabled). If the **Ws** is left unconnected, logic must be connected to the **We**, which will have full control of the register. If either input is left unconnected, you must retain the comma that delimits the field.

2. The logic feeding the Write Enable (**We**) input may contain up to 2 product terms.

## RINP8



| | |
|---|---|
| Name: | **RINP8** (8-Bit Registered Pin Input to Logic) |
| ADF Syntax: | **Out0, Out1, Out2, Out3, Out4, Out5, Out6, Out7 = RINP8(In0, In1, In2, In3, In4, In5, In6, In7, Ws, We)** |

Description:

Outputs:

| | | |
|---|---|---|
| **Out(0-7)** | = | outputs to logic array (Q0-Q7) |

Inputs:

| | | |
|---|---|---|
| **In(0–7)** | = | EPLD pin inputs |
| **Ws** | = | write strobe input (active low) |
| **We** | = | write enable input |

EPLDs: EPB1400 (BUSTER)

Notes:

1. The Write Strobe (**Ws**) input to this primitive is optional. This signal is predefined as an active low input, therefore, it is not necessary to invert the signal to implement active low logic. If the **Ws** is connected, it must be connected directly to the dedicated pin (/**CWS**) or **GND**. When the **Ws** input

is connected, the value of an unconnected Write Enable (**We**) input will default to **VCC** (register enabled). If the **Ws** is left unconnected, logic must be connected to the **We**, which will have full control of the register. If either input is left unconnected, you must retain the comma that delimits the field.

2. The logic feeding the Write Enable (**We**) input may contain up to 2 product terms.

SECTION 2

# Altera Design File Format

This section describes the functions and syntax of the Altera Design File (ADF). It includes a detailed description of the syntax of each section of the Altera Design File and a sample ADF.

## Functions

The Altera Design File has several functions:

- It is the common design entry format for A+PLUS software.
- It is the output of the State Machine and FutureNet Pinlist Converter programs.
- It is the output of the LogiCaps schematic capture software.
- It is the input to the Altera Design Processor.
- It is the entry format of the Boolean Equation and Netlist design entry methods.

## Syntax

This description of the Altera Design File follows the Backus-Naur Form (BNF) conventions. The ADF syntax for each individual section is described in the paragraphs below. A full BNF description of the ADF syntax is also given in Figure 2-1. (See *Appendix C* for a description of BNF rules.)

The ADF has the following syntax:

<adf netlist> ::= [<header>] <declarations> <network> [<equations>] 'END$'

### Keywords

All ADF keywords—**OPTIONS:**, **PART:**, **INPUTS:**, **OUTPUTS:**, **NETWORK:**, and **EQUATIONS:**—must be the first word in the line in which they appear. (Do not enter comments, tabs, or blank spaces before keywords.)

### White Space and Comments

White space is defined as blank spaces, tabs, carriage returns, and line feeds. White space may be inserted between syntax elements to enhance readability, except in the following cases:

- Within any name.
- Between **<prefix>** and **<expression>** or **<expression>** and **<postfix>** in a Boolean expression. (Refer to *Equations Section.*)
- Before keywords, as described in the preceding section

Comments must be enclosed by percent symbols (%). They may contain any printable character except the delimiter %. They may be inserted wherever white space is allowed, *except* within the Header Section of a file.

The comment string syntax is defined as follows:

<comment string> ::= '%' {<comment char>} '%'

where

<comment char> ::= <tab> | <EOL> | <space> | <hex 21> | ... | <hex 24> |
    <hex 26> | <hex 27> | ... | <hex 7E>
        *(i.e., all printable characters except the comment delimiter "%")*

where

<tab> ::= <hex 09>
<EOL> ::= <hex 0D> <hex 0A>
<space> ::= <hex 20>

## Header Section

The optional Header Section of the Altera Design File serves to identify a design entered by the user. This section has several important uses and restrictions:

- It is included in the header of the Utilization Report, the Logic Equation File, and the JEDEC File.
- Any printable character except the asterisk (*) is allowed.
- If it is present, it must be the first section in the ADF.
- It is terminated by either the **OPTIONS:** or **PART:** keywords, which A+PLUS recognizes as the start of the Declarations Section.

Information in the Header Section must appear in the following order, with one item on each line. The maximum character countfor each field is indicated in parentheses.

|  |  |  |
|---|---|---|
| 1. | Designer | (48) |
| 2. | Company | (60) |
| 3. | Date | (24) |
| 4. | Number | (24) |
| 5. | Revision | (24) |
| 6. | EPLD | (10) |
| 7. | Comment | (512) |
| 8. | Any other information (may be more than one line) |  |

The Header Section has the following syntax:

```
<header> ::= <header char>:0:48 <EOL>
        <header char>:0:60 <EOL>
        <header char>:0:24 <EOL>
        <header char>:0:24 <EOL>
        <header char>:0:24 <EOL>
        <header char>:0:10 <EOL>
        <header char>:0:512 <EOL>
        {{<header char>} <EOL> {{<header char>} <EOL>}
```

where

```
<header char> ::= <tab> | <space> | <hex 21> | ... | <hex 29> | <hex 2B> |
        <hex 2C> | ... | <hex 7E>
```

*(i.e., any printable ASCII character except an asterisk (*))*

where

```
<tab> ::= <hex 09>
<EOL> ::= <hex 0D> <hex 0A>
<space> ::= <hex 20>
```

## Declarations Section

The Declarations Section consists of four (sub)sections:

- Options Section—contains the Turbo-Bit and Security Bit options, which allow you to set the Turbo-Bit and Security Bit to **ON** or **OFF**. (If you do not specify **ON** or **OFF**, the Turbo-Bit defaults to **ON** and the Security Bit defaults to **OFF**.) (The BUSTER (EPB1400) and EP310 parts do not support the Turbo-Bit option. LogicMap will ignore any Turbo-Bit information entered for these EPLDs.)
- Part Section—contains a target part number, **AUTO** (for automatic part selection), or **MACRO** if you are creating a MacroFunction with **ADLIB** (available with LogiCaps).
- Inputs Section—contains input signal designations, including optional pin assignments.
- Outputs Section—contains output signal designations, including optional pin assignments.

The Declarations Section has the following syntax:

<declarations> ::= [<options>] <part> <inputs> <outputs>

where

<options> ::= 'OPTIONS:' <opt spec> {[ ',' ] <opt spec>} <EOL>
<part> ::= 'PART:' <part name> <EOL>
<inputs> ::= 'INPUTS:' <i/o list> <EOL>
<outputs> ::= 'OUTPUTS:' <i/o list> <EOL>

where

<opt spec> ::= <opt name> [ '=' <opt value>]
<part name> ::= 'EP310' | 'EP310D' | 'EP320' | 'EP320D' | 'EP600' | 'EP600D' | 'EP600J' | 'EP610' | 'EP610D' | 'EP610J' | 'EP900' |'EP900D' | 'EP900J' | 'EP910' | 'EP910D' | 'EP910J' | 'EP1210' | 'EP1210D' | 'EP1210J' | 'EPB1400' | 'EPB1400D' | 'EPB1400J' | 'EP1800' | 'EP1800J' | 'EP1800G' | 'AUTO' | 'MACRO'
<i/o list> ::= <i/o name> {',' <i/o name>}

where

<opt name> ::= 'TURBO' | 'SECURITY'
<opt value> ::= 'ON' | 'OFF'
<i/o name> ::= <pin name> [ '@' <pin number>]

where

<pin name> ::= <pin char>:1:8
<pin number> ::= [<letter prefix>] <digit> [<digit>]

where

<pin char> ::= <any printable ASCII character except comma, %, @, =, (, )>
<letter prefix> ::= 'A' | 'B' | ... | 'H' | 'J' | 'K' | 'L'
*(letter prefix allowed only in EP1800G pin numbers)*
<digit> ::= '0' | '1' | ... | '9'

☞

1. To use the automatic part selection feature, simply enter **AUTO** instead of a specific part name after the **PART:** keyword. The ADP will automatically choose the part most likely to fit your design. (Automatic part selection will not be successful if you specify pin assignments, or if there are too many inputs, outputs, or macrocells in the design.)

2. If you use an asterisk (*) in an input or output pin name, it will be converted into a tilde (~) in the JEDEC file generated by the ADP. You must ensure that this conversion will not create duplicate pin names (for example, if you use both * and ~ when naming pins).

3. The ADP creates internal node names that contain periods (.). User-assigned pin names that contain periods may occasionally conflict with these node names and cause unpredictable results.

4. The Functional Simulator ignores all user-defined pin names that contain periods (periods are allowed only for referencing internal nodes of I/O primitives).

## Network Section

The Network Section specifies the input, output, and I/O architecture to be programmed. Primitive statements control the pin connections to internal logic and I/O macrocell configurations.

The Network Section has the following syntax:

```
<network> ::= 'NETWORK:' <network list>
```

where

```
<network list> ::= <primitive equation> | <MacroFunction statement>
        {<primitive equation> | <MacroFunction statement>}
```

where

```
<primitive equation> ::= <parameters> '=' <primitive> '(' <parameters> ')'
<MacroFunction statement> ::= <MacroFunction> '(' <parameters> ')'
```

where

```
<parameters> ::= <pin name> | <name> {',' <name>}
<primitive> ::= <any Altera Primitive name>
              (e.g., CONF, INP, NORF)
<MacroFunction> ::= <any user-defined or Altera MacroFunction name>
              (e.g., UNICNT, 74393)
```

where

```
<pin name> ::= <pin char>:1:8
<name> ::= <name char>:1:8
```

where

```
<pin char> ::= <any printable ASCII character except comma, %, @, =, (, )>
<name char> ::= <digit> | <alphabet letter>
```

where

```
<digit> ::= '0' | '1' | ... | '9'
<alphabet letter> ::= 'A' | 'B' | ... | 'Z' | 'a' | 'b' | ... | 'z'
```

Refer to *Altera Primitive Library* for primitive names and syntax. MacroFunction names and syntax are given in the optional ***LogiCaps*** and ***TTL MacroFunctions*** packages.

## Equations Section

The optional Equations Section supports the direct entry of logical equations. It consists of the keyword **EQUATIONS:**, followed by zero or more logical equations. Each equation has exactly one node name on the left, an equals sign (=), a Boolean expression, a semicolon (;), and an End-of-Line (EOL) sequence.

The Equations Section has the following syntax:

<equations> ::= 'EQUATIONS:' {<logic equation>}

where

<logic equation> ::= <LHS> '=' <expression> ';' <EOL>

where

<LHS> ::= <prefix><name> | <name> | <name><postfix>
*(no white space is allowed between <prefix><name> and <name><postfix>)*
<expression> ::= <name> | <prefix><expression> | <expression><postfix> | <expression> <infix> <expression> | '(' <expression> ')'
*(no white space is allowed between <prefix><expression> and <expression><postfix>)*

where

<prefix> ::= '/' | '!'
<infix> ::= '+' | '*' | '#' | '&'
<postfix> ::= '''
<name> ::= <name char>:1:8

where

<name char> ::= <digit> <alphabet letters>

where

```
<digit> ::= '0' | '1' | ... | '9'
<alphabet letters> ::= 'A' | 'B' | ... | 'Z' | 'a' | 'b' | ... | 'z'
```

## End Statement

This statement terminates the ADF. It has the following syntax:

```
'END$' <EOL>
```

```
<adf netlist> ::= [<header>] <declarations> <network> [<equations>] 'END$'

<header> ::= <header char>:0:48 <EOL>
        <header char>:0:60 <EOL>
        <header char>:0:24 <EOL>
        <header char>:0:24 <EOL>
        <header char>:0:24 <EOL>
        <header char>:0:10 <EOL>
        <header char>:0:512 <EOL>
        {{<header char>} <EOL> {{<header char>} <EOL>}
<header char> ::= <tab> | <space> | <hex 21> | ... | <hex 29> | <hex 2B> |
        <hex 2C> | ... | <hex 7E>
            (i.e., any printable ASCII character except an asterisk (*))
<tab> ::= <hex 09>
<EOL> ::= <hex 0D> <hex 0A>
<space> ::= <hex 20>

<declarations> ::= [<options>] <part> <inputs> <outputs>
<options> ::= 'OPTIONS:' <opt spec> {[ ',' ] <opt spec>} <EOL>
<part> ::= 'PART:' <part name> <EOL>
<inputs> ::= 'INPUTS:' <i/o list> <EOL>
<outputs> ::= 'OUTPUTS:' <i/o list> <EOL>
<opt spec> ::= <opt name> [ '=' <opt value>]
<opt name> ::= 'TURBO' | 'SECURITY'
<opt value> ::= 'ON' | 'OFF'
<part name> ::= 'EP310' | 'EP310D' | 'EP320' | 'EP320D' | 'EP600' | 'EP600D'
        | 'EP600J' | 'EP610' | 'EP610D' | 'EP610J' | 'EP900' |'EP900D' |
        'EP900J' | 'EP910' | 'EP910D' | 'EP910J' | 'EP1210' | 'EP1210D' |
        'EP1210J' | 'EPB1400' | 'EPB1400D' | 'EPB1400J' | 'EP1800' |
        'EP1800J' | 'EP1800G' | 'AUTO' | 'MACRO'
<i/o list> ::= <i/o name> {',' <i/o name>}
<i/o name> ::= <pin name> [ '@' <pin number>]
<pin name> ::= <pin char>:1:8
<pin char> ::= <any printable ASCII character except comma, %, @, =, (, )>
<pin number> ::= [<letter prefix>] <digit> [<digit>]
<letter prefix> ::= 'A' | 'B' | ... | 'H' | 'J' | 'K' | 'L'
            (letter prefix allowed only in EP1800G pin numbers)
<digit> ::= '0' | '1' | ... | '9'
```

**Figure 2-1. BNF Syntax for Altera Design File (Part 1 of 2)**

```
<network> ::= 'NETWORK:' <network list>
<network list> ::= <primitive equation> | <MacroFunction statement>
        {<primitive equation> | <MacroFunction statement>}
<primitive equation> ::= <parameters> '=' <primitive> '(' <parameters> ')'
<MacroFunction statement> ::= <MacroFunction> '(' <parameters> ')'
<parameters> ::= <pin name> | <name> {',' <name>}
<name> ::= <name char>:1:8
<name char> ::= <digit> | <alphabet letter>
<primitive> ::= <any Altera Primitive name>
              (e.g., CONF, INP, NORF)
<MacroFunction> ::= <any Altera or user-defined MacroFunction name>
              (e.g., UNICNT, 74393)

<equations> ::= 'EQUATIONS:' {<logic equation>}
<logic equation> ::= <LHS> '=' <expression> ';' <EOL>
<LHS> ::= <prefix><name> | <name> | <name><postfix>
              (no white space is allowed between <prefix><name> and
              <name><postfix>)
<expression> ::= <name> | <prefix><expression> | <expression><postfix> |
        <expression> <infix> <expression> | '(' <expression> ')'
              (no white space is allowed between <prefix><expression> and
              <expression><postfix>)
<prefix> ::= '/' | '!'
<infix> ::= '+' | '*' | '#' | '&'
<postfix> ::= '''

'END$'
```

**Figure 2-1. BNF Syntax for Altera Design File (Part 2 of 2)**

## Netlist Files

Using a standard text editor, you can type a netlist corresponding to your design into the ADF format. You may also adapt the netlist output of another design interface to the ADF format. (For guidelines on ADF entry, refer also to *Boolean Equation Entry* in the ***A+PLUS User Guide***.)

Figure 2-2 shows a sample ADF. Using only netlist format, this ADF describes the same 4-bit counter design shown in Figure BE-6 in *Boolean Equation Entry* .

```
DESIGNER NAME
COMPANY NAME
9/30/87
1
A
EP600                                   Turbo-Bit and Security Bit status
4 BIT COUNTER                           declared

OPTIONS: TURBO = ON, SECURITY = OFF

PART: EP600                             Active Low input

INPUTS: CLOCK@2, !HOLD@11

OUTPUTS: Q1, Q2, Q3, Q4                 Active High outputs

NETWORK:

CLOCK = INP(CLOCK)                      Asynchronous clock
nHOLD = INP(!HOLD)

Q1,Q1 = TOTF(D1t, CLOCKa,,,)
Q2,Q2 = TOTF(D2t, CLOCKa,,,)
Q3,Q3 = TOTF(D3t, CLOCKa,,,)
Q4,Q4 =TOTF(D4t, CLOCKa,,,)

CLOCKa = AND(D1t,CLOCK)

D1t = NOT(nHOLD)
D2t = AND(Q1,Q1)
D3t = AND(Q2,Q1)
D4t = AND(Q3,Q2,Q1)

END$
```

**Figure 2-2. Sample ADF**

SECTION 3

# A+PLUS and ADP Reference

This section contains:

- A detailed description of the functions available on the APLUS Menu
- A detailed description of the functions available on the ADP Menu
- A summary of the automatic functions of the Altera Design Processor (ADP)
- Instructions for running or invoking A+PLUS programs from the DOS command line (i.e., *stand-alone* mode)

For detailed information on using A+PLUS, see the individual design entry method sections of the ***A+PLUS User Guide***.

## APLUS Menu Functions

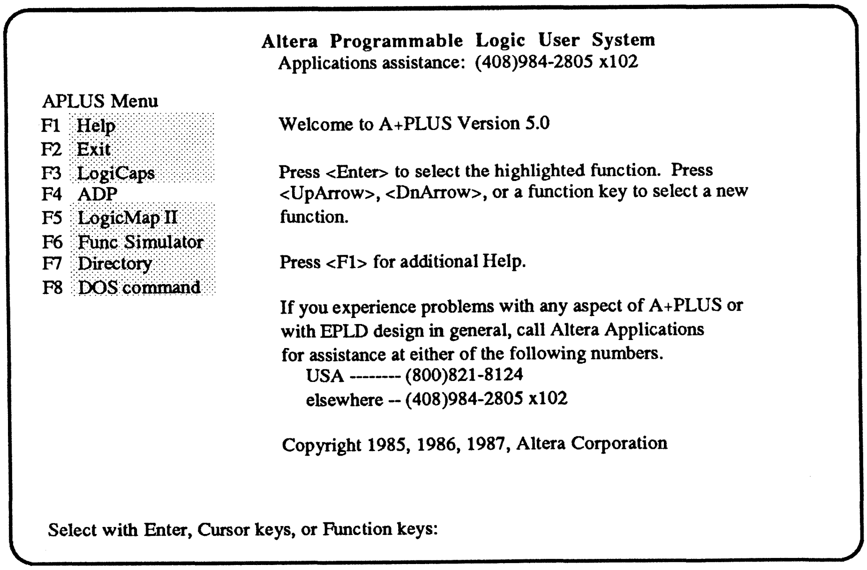
The APLUS Menu provides access to the following:

- Help
- Text editor of your choice
- Altera Design Processor (ADP)
- LogicMap II
- Functional Simulator (an optional package)
- Current directory
- Any DOS command

To invoke the APLUS Menu, type at the DOS prompt:

**APLUS <Enter>**

The APLUS Menu is displayed, as shown in Figure 3-1.



```
                         Altera Programmable Logic User System
                           Applications assistance: (408)984-2805 x102

APLUS Menu
F1  Help                   Welcome to A+PLUS Version 5.0
F2  Exit
F3  LogiCaps               Press <Enter> to select the highlighted function. Press
F4  ADP                    <UpArrow>, <DnArrow>, or a function key to select a new
F5  LogicMap II            function.
F6  Func Simulator
F7  Directory              Press <F1> for additional Help.
F8  DOS command
                           If you experience problems with any aspect of A+PLUS or
                           with EPLD design in general, call Altera Applications
                           for assistance at either of the following numbers.
                              USA -------- (800)821-8124
                              elsewhere -- (408)984-2805 x102

                           Copyright 1985, 1986, 1987, Altera Corporation


Select with Enter, Cursor keys, or Function keys:
```

**Figure 3-1. APLUS Menu**

The current (default) menu selection, **<F4> (ADP)**, is highlighted. The numbers displayed on the left of the menu selections show the function keys used to execute each function. You may also select functions with the <↓> and <↑> cursor keys and **<Enter>**.

The APLUS Menu functions are:

**<F1>** **Help** Displays helpful information on APLUS Menu functions. After pressing **<F1>**, you may use the function or <↓> and <↑> keys to display a help message for each menu function. Press **<Enter>** to quit the **Help** function.

**<F2>** **Exit** Terminates A+PLUS and returns you to DOS. (Use the **DOS Command** (**<F8>**) function to execute DOS commands during the current A+PLUS session.)

**<F3>** **<LogiCaps>** Invokes the text editor or schematic capture program of your choice. LogiCaps is the default, but you can override the default by entering another editor (e.g., WordStar) during installation.

**<F4>** **ADP** Invokes the menu for the Altera Design Processor (ADP). The functions controlled by the ADP Menu allow you to process your design file into a JEDEC file used for programming an Altera EPLD. (See *ADP Menu Functions* below.)

**<F5>** **LogicMap II** Invokes the LogicMap II program, which allows you to program an Altera EPLD with A+PLUS hardware. If you attempt to use this function when the Logic Programmer card is not installed, the following message is displayed:

```
            Programmer self test failed

   Device must NOT be in socket for this test to pass!

      Enter:
         C  to continue without programming card
         T  to run diagnostics again
         Q  to return to operating system
```

(See the ***LogicMap II*** manual for detailed information on device programming.)

<F6> **Func Simulator** Invokes Altera's Functional Simulator (an optional package). Refer to the *Functional Simulator* section of the ***A+PLUS User Guide***.

<F7> **Directory** Allows you to enter a search pattern. You may enter a drive or path name with asterisk (* – for any sequence) and question mark (? – for any single character) characters for pattern matching. A+PLUS then displays a list of DOS matching the pattern.

<F8> **DOS command** Temporarily returns you to DOS so that you can enter a DOS command. After entering the DOS command, type **EXIT** to return to APLUS. (Note: Do not re-invoke A+PLUS while in this temporary DOS environment—a duplicate copy of A+PLUS will be read into memory.)

## ADP Menu Functions

Several important Altera Design Processor features are controlled by the functions associated with the ADP Menu. This menu allows you to specify the conditions used for compiling your design file.

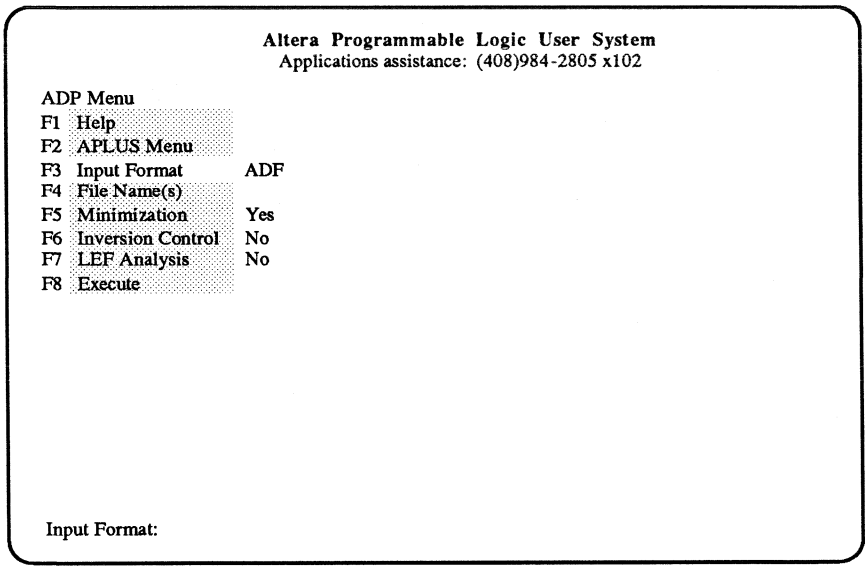
The ADP Menu allows you to:

- Request on-line help
- Return to the APLUS Menu
- Specify input file format
- Specify input file name(s)
- Request logic minimization
- Control logic inversion
- Request LEF Analysis
- Execute the ADP

To invoke the ADP Menu, press **<F4>** while in the APLUS Menu. The screen clears and the ADP Menu is displayed, as shown in Figure 3-2.

The numbers displayed on the left of the menu selections show the function keys used to execute each function. The current (default) menu selection, **<F3> (Input Format)**, is highlighted. This function is the first in the default sequence for entering processing parameters, but you may move around the menu and enter and change processing options in any order. (Use the Function keys **<F1>** through **<F8>** or the **<↑>**, **<↓>**, and **<Enter>** keys.) You can accept the default processing settings, which are shown on the right of the function names, by simply pressing **<Enter>**.

As each parameter is entered, it appears on the bottom line of the screen and may be edited with the **<→>** and **<←>** cursor keys, as well as the **<Del>**, **<Backspace>**, **<Home>**, and **<End>** keys.


**Altera Programmable Logic User System**
Applications assistance: (408)984-2805 x102

```
ADP Menu
F1  Help
F2  APLUS Menu
F3  Input Format       ADF
F4  File Name(s)
F5  Minimization       Yes
F6  Inversion Control  No
F7  LEF Analysis       No
F8  Execute

Input Format:
```

**Figure 3-2. ADP Menu**

The ADP Menu functions are:

<F1> **Help** Displays helpful information on ADP Menu functions. After pressing <F1>, you may use the function or <↓> and <↑> keys to display a help message for each menu function. Press <Enter> to quit the **Help** function.

<F2> **APLUS Menu** Terminates the current ADP session and returns you to the APLUS Menu.

<F3> **Input Format** Prompts you for the format of your input file:

| | |
|---|---|
| **A** or **<Enter>** – | for an Altera Design File with the extension **.ADF** (the default) |
| **C** – | for a component list with the extension **.CMP**, as output by P-CAD's PC-CAPS |
| **P** – | for a pinlist with the extension **.PIN**, as output by FutureNet's DASH Schematic Design Editor |
| **S** – | for a State Machine File with the extension **.SMF** |

<F4> **File Name(s)** Prompts you for the name(s) of file(s) containing your design. You need not enter the filename extension.

If you type **?**, you are prompted with **File/Directory search pattern:**. If you press **<Enter>**, a list of files in the current directory with the extension specified in **Input Format <F3>** is displayed. If you enter a path or drive name, a list of the files in the specified drive and/or directory is displayed. The most recently updated file is then provided as a default.

☞ You may enter up to 9 input filenames, separated by blanks or commas. If any filename has a different extension from the one entered at the **<F3>** prompt, you must type both the filename *and* extension. (All files will be automatically converted into ADFs and processed into a single JEDEC file.)

<F5> **Minimization** Allows you to request a reduction of the Boolean logic of the design. If you enter **Y** (Yes [the default]), Boolean minimization is performed on the design. If you enter **N** (No), minimization is not performed and the ADP Menu selection automatically moves to the **LEF Analysis** (<F7>) prompt.

<F6> **Inversion Control** Allows you to request control over inversion of individual Boolean equations during the minimization process. If you enter **Y** (Yes), you are asked on an equation-by-equation basis whether you wish to perform a De Morgan's inversion. This feature is useful mainly for unusual EP310 and EP1210 designs. For other EPLDs, you should enter **N** (No [the default]). The ADP then quickly inverts each equation to determine whether the inverted form reduces further than the non-inverted form. (In either case, the actual output of the pin remains the same, and the equation is retained in the form that contains the fewest product terms. If both forms contain the same number of product terms, the non-inverted form is retained.)

Every time the ADP asks whether you wish to perform an inversion on a particular equation, information on the equation's current status is displayed, including the product-term count after the first pass through the reduction algorithm and the symbol on the left-hand side of the equation. If this symbol is complemented, the equation has already been inverted by the software, and an inversion will cause the left-hand side of the equation to be uncomplemented.

<F7> **LEF Analysis** Allows you to request the ADP to analyze the intermediate Logic Equation File (LEF). If you enter **Y** (Yes), this module converts the intermediate LEF output by the Expander (or Minimizer) into a format similar to that of the ADF. This file appears in the current directory with the extension **.LEF**, and allows you to view the design after it has been translated, expanded, and, if requested at the **<F5>** prompt, minimized.

The LEF records the name of the input ADF and the ADP options you selected for processing to the header section of the file. It also documents any primitives that were promoted to different primitives, and, if necessary, inserts default values for the Turbo-Bit and Security Bit. The **.LEF** file enables you to check the pin assignments and the product term count, and to determine whether design modifications are needed to assist in the fitting process. After examining the LEF, you may wish to edit your input file and resubmit it to the ADP. (For an example of ADF-to-LEF translation, see *Appendix B*.)

For large designs, a non-minimized LEF may provide convenient breakpoints for partitioning. (More opportunities for partitioning can be found in the non-minimized representation of a design.) Note, however that there may not be enough memory to complete design processing (i.e., you will probably obtain an LEF but not a JEDEC file).

If you enter **N** (No [the default]) at the **LEF Analysis** prompt, the ADP indicates that you have answered the prompts for **<F3>** through **<F7>**, by asking:

**Do you wish to run under the above conditions [Y/N]?**

Enter **Y** or press **<F8>** (**Execute**) to execute the ADP. If you enter **N**, the ADP Menu returns to the **Input Format** function (**<F3>**), allowing you to change any of the previously entered parameters.

**<F8>** **Execute** Immediately processes the design according to the current menu options. (You are prompted for a filename if you have not entered one previously.)

This function also allows you to repeat processing during design iterations. If you need to edit a design, use the **Editor** (**<F3>**) function on the A+PLUS Menu to edit your input file. When you exit from your editor (e.g., LogiCaps) you are automatically returned to the APLUS Menu. Then, press **<F4>** to return to the ADP Menu and press **<F8>** to repeat design processing. The previous menu settings, including the name of the most recently processed file, will remain in effect until they are explicitly changed.

When the ADP has finished processing the design, you are asked:

**Would you like to implement another design [Y/N]?**

If you enter **Y**, a new loop through the ADP Menu is initiated. If you enter **N**, you are returned to the APLUS Menu.

## Automatic ADP Functions

Many functions of the Altera Design Processor are built into the A+PLUS software and are not user-controllable. The ADP *automatically* performs these tasks during normal design processing (i.e., they cannot be altered with the APLUS or ADP Menu functions). When used with the menu options described above, the ADP is a powerful tool for compiling and fitting an EPLD design.

The ADP's automatic functions are:

- Displays information, error, and warning messages, which are prefixed with *****INFO-**, *****ERR-**, and *****WARN-**, respectively. Many A+PLUS information messages simply indicate that a particular ADP module has completed execution without error. (For detailed information, see *A+PLUS Messages*.)

- Logs all information, error, and warning messages to a file called **ADP.LOG** in the current directory. This file is overwritten each time the ADP is invoked.

- Expands Boolean equations in the ADF into sum-of-products form, checks for evidence of combinatorial feedback, and removes redundant factors from product terms.

- Checks the ADF for MacroFunction statements, which are expanded and replaced with primitive statements. The ADP then converts the ADF into a file with the extension **.SDF**, and continues design processing. (No action is taken if MacroFunction statements are not present.)

- Translates the **.ADF** or **.SDF** (from the previous step) into internal logic equations.

- Selects an appropriate EPLD for the design if automatic part selection is requested in the input design file.

- Generates a Utilization Report (**.RPT** file) that documents macrocell and pin assignments, pin names, and buried registers, as well as any unused resources in the target EPLD.

- Generates a JEDEC file with the extension **.JED** for programming an Altera EPLD.

## Running A+PLUS Programs from DOS

A+PLUS software allows you to invoke and/or run the Altera Design Processor, Functional Simulator, FutureNet Pin List Converter (FCV), LogiCaps schematic capture package, LogicMap II program, and State Machine Converter (SMV) directly from DOS (i.e., in stand-alone mode). The following pages describe how to invoke and/or run these programs from the DOS command line.

Stand-alone execution of the ADP, SMV, or FCV increases the available memory for larger designs by approximately 70 Kbytes.

Stand-alone execution of the SMV and FCV allows you to create a design in several stages. For example, you may wish to combine design data from input files with different formats. By using DOS commands, you can run these converters without running the ADP, and examine the resulting ADFs before further processing is initiated. After State Machine and/or FutureNet Pin List files have been converted into ADFs, they may be combined for processing by the ADP.

☞

1. In the following descriptions, command line options are shown in square brackets ([ ]); required items are shown without brackets.

2. On-screen messages are the same for Command Line mode and Menu mode.

## Altera Design Processor

To run the ADP from the DOS command line, type:

```
ADP [-mdl] <filename> <Enter>
```

where the command line options have the following meaning:

| | |
|---|---|
| **-m** | perform minimization |
| **-d** | manually control De Morgan's inversion |
| **-l** | run LEF Analyzer |

You may use any combination of these options, e.g., **-ml**. The filename extension **.ADF** is assumed.

## Functional Simulator

To run the Functional Simulator in *batch mode* from the DOS command line, type:

```
FSIM <filename> <Enter>
```

☞ Do not type the **.JED** filename extension.

To run the Functional Simulator in *interactive mode* from the DOS command line, type:

```
FSIM <Enter>
```

## FutureNet Pin List Converter

To run the FutureNet Pin List Converter from the DOS command line, type:

```
FCV <filename> <Enter>
```

The filename extension **.PIN** is assumed.

## LogiCaps

To invoke LogiCaps from the DOS command line, type:

**LOGICAPS [filename] [@macroname] [options] <Enter>**

where the command line options have the following meaning:

| | |
|---|---|
| **filename** | (optional) The name of a design file to be automatically loaded. |
| **macroname** | (optional) The name of a macro recorded and written to a file with the extension **.MAC**. It is used instead of **INIT.MAC**. |
| **options** | Additional configuration specifications that you may enter when invoking LogiCaps. The available options are: |

**-1** Use COM1 for the mouse (the default).

**-2** Use COM2 for the mouse.

**-l <symbol library>** Selects a different symbol library. The default is **ALTERA.SYM**.

**-ml** Increase the memory for **LINE** objects. This option allows you to reallocate memory to increase the memory capacity for lines.

**-ms** Increase the memory for **SYMBOL** objects. This option allows you to reallocate memory to increase the memory capacity for symbols.

**-mt** Increase the memory for **TEXT** objects. This option allows you to reallocate memory to increase the memory capacity for text entries.

**-x** Set new file configuration data. This option lets you ignore configuration specifications in **LOGICAPS.CFG** and prompts you for new configuration information.

For a screen display of this information, type at the DOS prompt:

```
LOGICAPS ? <Enter>
```

## LogicMap II

To invoke LogicMap II from the DOS command line, type:

```
LOGICMAP <Enter>
```

## State Machine Converter

To run the State Machine Converter from the DOS command line, type:

```
SMV <filename> <Enter>
```

The filename extension **.SMF** is assumed.

## SECTION 4

# Utilization Report

This section describes the Utilization Report and provides several sample **.RPT** files.

The Utilization Report is generated by the Fitter module of the Altera Design Processor. This report documents how an EPLD's resources (i.e., pin and macrocell assignments, input and output pin names, and buried registers) were used by a design, as well as any unused resources. The Utilization Report is automatically stored and listed in the directory containing the input design file as **filename.RPT**.

☞ If you have specified all pin assignments and the design fits, you need not refer to the Utilization Report. However, if you did not specify all pin assignments, you should to refer to this report to determine which pin assignments were selected by the Fitter.

When fitting a design, the Fitter module first tries to fit any pin assignments requested in the design file. If the Fitter is unable to accommodate a pin assignment, it displays a message asking whether the request can be ignored. If you enter **Y** (Yes), all pin requests are ignored and the Fitter continues; if you enter **N** (No), the Fitter terminates.

For additional information, refer to *Appendix A*, which explains macrocell groups for each EPLD. This information will assist you in choosing pin assignments that will enable the Fitter to find a satisfactory fit for your design.

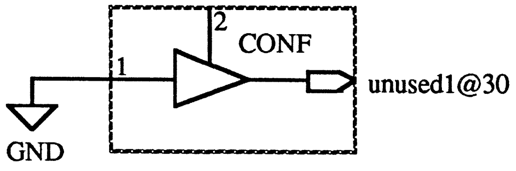
## Header Information

The first line of a Utilization Report contains a statement indicating whether a design has been implemented successfully.

This line is followed by the ADF header information, including the input filename(s) and ADP options requested for design processing, and a representation of the target EPLD that shows both used and unused pins.

- **GND** indicates unused input pins and unused macrocell group pins (except for the actual **GND** pins). Pins marked **GND** must be connected to ground; they should never be left floating.

  You may retain unused pins for future use by attaching a **CONF** primitive to each unused pin as shown in the example below. This allows you to avoid the external **GND** requirement by grounding the pin within the the EPLD. The pin is thus reserved and may be reprogrammed for future use.



- **RESERVED** indicates a buried register. (Note that in EP1210 applications, **RESERVED** may also refer to unused macrocells in a group in which one or more of that group's macrocells are being used.) **RESERVED** pins must be left floating.

- Pins marked **N.C.** have no internal connection within the EPLD.

## Utilization Information

The header information section is followed by the utilization information, which is divided into **OUTPUTS, BURIED REGISTERS, INPUTS, LATCHES, UNUSED RESOURCES,** and **PART UTILIZATION** subsections.

### Outputs, Buried Registers, Inputs, Latches, and Unused Resources Subsections

These subsections contain utilization information which is listed under the headings described below. (Note: the **LATCHES** subsection is generated only for BUSTER (EPB1400) Utilization Reports.)

| | |
|---|---|
| **Name** | Shows the signal name assigned by the user. (Note: if dual I/O feedback has been used, the signal name is shown in both the **INPUTS** and **OUTPUTS** subsections.) |
| **Pin** | Shows the pin to which the signal name was assigned. A pin number followed by an exclamation point (!) indicates that the Fitter's pin assignment differs from the pin assignment requested in your original design file. |
| **Resource** | Shows the name of the primitive. A primitive name followed by an exclamation point (!) indicates that the original primitive used in your design file has been promoted to the primitive shown. (In JK and SR primitives, the resource will always be the primitive originally requested in your design, even if the Logic Equation File indicates that it has been minimized to a D- or T-type register.) |
| **MCell** | Shows the macrocell number to which the signal name was assigned. |

| | |
|---|---|
| **Ref #** | Shows the hardware latch number (**L#**) and bit number to which the signal name was assigned. |
| **PTerms** | Shows the number of product terms used and the number of product terms available. (In JK and SR primitives, this is the total number of product terms for both equations.) |

The following columns appear in the Utilization Reports for EPLDs that use grouped local feedback or grouped synchronous clocks, Output Enable (**Oe**), Preset (**P**), and Clear (**C**) inputs:

| | |
|---|---|
| **FdBck Group** | For EPLDs supporting local feedback, this column indicates the local feedback group that can feed and be fed by the signal. A feedback group number followed by the letter **G** indicates that this signal may globally feed any macrocell in the EPLD. |
| **Sync Clock** | For EPLDs supporting synchronous clock groups, this column indicates the input signal that is used as this macrocell's synchronous clock. |
| **Oe** | For EPLDs supporting grouped Output Enable (**Oe**), this column indicates the name of the signal that feeds this macrocell's **Oe** input. |
| **Clear** | For EPLDs supporting grouped Clear (**C**) control, this column indicates the name of the signal that feeds this macrocell's **C** input. |
| **Preset** | For EPLDs supporting grouped Preset (**P**) control, this column indicates the name of the signal that feeds this macrocell's **P** input. |

## Part Utilization Subsection

The **PART UTILIZATION** subsection shows the percentage of pins, macrocells, and product terms used in the design.

## Macrocell Interconnection Cross Reference

The **Macrocell Interconnection Cross Reference** indicates the connectivity of the design by showing the inputs, macrocells, or latches that feed or are fed by a particular signal. Each row in the cross-reference lists:

- An input or output signal name.
- The signal resource (i.e., a primitive).
- The pin number (@# or @##) or macrocell number (@M# or @M##) to which the signal is attached.
- A list of symbols indicating which macrocells (M# or M##) and latches (L#) it feeds. A signal is considered to feed a macrocell or latch if it feeds a D, T, SR, or JK flipflop; or C, Oe, Output Latch Enable (Ole), P, Read Enable (Re), Read Strobe (Rs), Write Strobe (Ws), Write Enable (We), or asynchronous clock input.
- The macrocell (M# or M##) or pin number (# or ##) associated with the pin or macrocell, respectively. If this number is enclosed in parentheses (), it indicates that the pin or macrocell is buried.

The following symbols are used to show the interconnection cross-reference:

| | |
|---|---|
| * | Indicates that the signal feeds one or more product terms in the corresponding macrocell or latch. |
| • | Indicates that the signal is not an input to the corresponding macrocell or latch. |

If the EPLD supports local feedback, the following symbols may also appear:

| | |
|---|---|
| ? | Indicates that the local signal feeds the corresponding macrocell or latch, but is in a different local feedback group and cannot reach it (i.e., the design does not fit). |
| x | Indicates that the local signal is in a different local feedback group from the latch or macrocell. |

## Sample Utilization Reports

Tables 4-1, 4-2, 4-3, and 4-4 show sample Utilization Reports for the EP310, EP600, EPB1400 (BUSTER), and EP1800J. Note that the EP310 report is for the **BEVDIS** sample design described in *Boolean Equation Entry* in the ***A+PLUS User Guide***.

## Table 4-1. Sample Utilization Report for EP310 (Part 1 of 3)

```
ALTERA Design Processor Utilization Report                        bevdis.rpt
@(#) FIT Version 5.0     9/30/87 19:42:24 39.16
 ***** Design implemented successfully

Your Name
Your Company
9/30/87
1.00
B
EP310
Beverage Dispenser Controller

Input files : bevdis.adf
ADP Options: Minimization = Yes,  Inversion Control = No,  LEF Analysis = Yes

OPTIONS: TURBO = ON, SECURITY = OFF

               EP310
              - - - - -
     CLOCK -|1       20|- Vcc
   CUPFULL -|2       19|- Gnd
  COINDROP -|3       18|- Gnd
       Gnd -|4       17|- Gnd
     RESET -|5       16|- Gnd
       Gnd -|6       15|- Gnd
    ENABLE -|7       14|- DROPCUP
       Gnd -|8       13|- STROBE
       Gnd -|9       12|- POURDRNK
       GND -|10      11|- Gnd
              - - - - -
```

**Table 4-1. Sample Utilization Report for EP310 (Part 2 of 3)**

```
                                                                  bevdis.rpt
**OUTPUTS**

      Name   Pin  Resource  MCell  PTerms  | Sync Clock |   Clear    |  Preset

   DROPCUP   14       RORF      6    1/ 8  | CLOCK      | NEWCYCLE   | RESET
  POURDRNK   12       RORF      8    2/ 8  | CLOCK      | NEWCYCLE   | RESET
    STROBE   13       CONF      7    2/ 8  |     -      |     -      |    -


**INPUTS**

      Name   Pin  Resource  MCell  PTerms  | Sync Clock |   Clear    |  Preset

     CLOCK    1        CKR      -      -   |     -      |     -      |    -
  COINDROP    3        INP      -      -   |     -      |     -      |    -
   CUPFULL    2        INP      -      -   |     -      |     -      |    -
    ENABLE    7        INP      -      -   |     -      |     -      |    -
     RESET    5        INP      -      -   |     -      |     -      |    -


**UNUSED RESOURCES**

      Name   Pin  Resource  MCell  PTerms  | Sync Clock |   Clear    |  Preset

         -    4      INPUT      -      -   |     -      |     -      |    -
         -    6      INPUT      -      -   |     -      |     -      |    -
         -    8      INPUT      -      -   |     -      |     -      |    -
         -    9      INPUT      -      -   |     -      |     -      |    -
         -   11      INPUT      -      -   |     -      |     -      |    -
         -   15      MCELL      5      8   | CLOCK      | NEWCYCLE   | RESET
         -   16      MCELL      4      8   | CLOCK      | NEWCYCLE   | RESET
         -   17      MCELL      3      8   | CLOCK      | NEWCYCLE   | RESET
         -   18      MCELL      2      8   | CLOCK      | NEWCYCLE   | RESET
         -   19      MCELL      1      8   | CLOCK      | NEWCYCLE   | RESET


**PART UTILIZATION**

3/ 8  MacroCells (37%)
4/ 9  Input Pins (44%)
      PTerms Used 20%
```

## Table 4-1. Sample Utilization Report for EP310 (Part 3 of 3)

```
Macrocell Interconnection Cross Reference                    bevdis.rpt

FEEDBACKS:
                              M M M
                              6 7 8
DROPCUP .. RORF @M6 -> * * * @14
STROBE ... CONF @M7 -> . . . @13
POURDRNK . RORF @M8 -> * * * @12


INPUTS:

CUPFULL .. INP  @2  -> . . *
COINDROP . INP  @3  -> * . .
RESET .... INP  @5  -> * . *
ENABLE ... INP  @7  -> * . *
                       D S P
                       R T O
                       O R U
                       P O R
                       C B D
                       U E R
                       P   N
                           K
```

## Table 4-2. Sample Utilization Report for EP600 (Part 1 of 3)

```
ALTERA Design Processor Utilization Report                          16bitpii.rpt
@(#) FIT Version 5.0      9/30/87 19:42:24 39.16
 ***** Design implemented successfully

DON FARIA
ALTERA
SEPTEMBER 1987
1.00
A
EP600
16BIT SHFT/CNT
Input files : 16bitpii.adf
ADP Options: Minimization = Yes,  Inversion Control = No,  LEF Analysis = Yes

OPTIONS: TURBO = ON, SECURITY = OFF
 ***** Externally connect signal "CLOCK" to pins 1 and 13.

               EP600
              - - - - -
     CLOCK -|1       24|- Vcc
      UPDN -|2       23|- DATA
       Q16 -|3       22|- Q1
       Q15 -|4       21|- Q2
       Q14 -|5       20|- Q3
       Q13 -|6       19|- Q4
       Q12 -|7       18|- Q5
       Q11 -|8       17|- Q6
       Q10 -|9       16|- Q7
        Q9 -|10      15|- Q8
   SHFTCNT -|11      14|- LTRT
       GND -|12      13|- CLOCK
              - - - - -
```

**Table 4-2. Sample Utilization Report for EP600 (Part 2 of 3)**

```
                                                          16bitpii.rpt
**OUTPUTS**

     Name  Pin  Resource  MCell  PTerms  | Sync Clock

       Q1   22      TOIF      1    4/ 8  | .01126050
       Q2   21      TOIF      2    6/ 8  | .01126050
       Q3   20      TOIF      3    6/ 8  | .01126050
       Q4   19      TOIF      4    6/ 8  | .01126050
       Q5   18      TOIF      5    6/ 8  | .01126050
       Q6   17      TOIF      6    6/ 8  | .01126050
       Q7   16      TOIF      7    6/ 8  | .01126050
       Q8   15      TOIF      8    6/ 8  | .01126050
       Q9   10      TOIF     16    6/ 8  | .01126050
      Q10    9      TOIF     15    6/ 8  | .01126050
      Q11    8      TOIF     14    6/ 8  | .01126050
      Q12    7      TOIF     13    6/ 8  | .01126050
      Q13    6      TOIF     12    6/ 8  | .01126050
      Q14    5      TOIF     11    6/ 8  | .01126050
      Q15    4      TOIF     10    6/ 8  | .01126050
      Q16    3      TOIF      9    6/ 8  | .01126050


**INPUTS**

     Name  Pin  Resource  MCell  PTerms  | Sync Clock

    CLOCK    1       CKR      -       -  |     -
     DATA   23       INP      -       -  |     -
     LTRT   14       INP      -       -  |     -
  SHFTCNT   11       INP      -       -  |     -
     UPDN    2       INP      -       -  |     -


**PART UTILIZATION**

16/16  MacroCells (100%)
 4/ 4  Input Pins (100%)
       PTerms Used 73%
```

## Table 4-2. Sample Utilization Report for EP600 (Part 3 of 3)

```
Macrocell Interconnection Cross Reference                           16bitpii.rpt

FEEDBACKS:                                                M M M M M M M
                              M M M M M M M M           M 1 1 1 1 1 1 1
                              1 2 3 4 5 6 7 8           9 0 1 2 3 4 5 6
Q1 ....... TOIF @M1 ->        * * * * * * * *           * * * * * * * * @22
Q2 ....... TOIF @M2 ->        * * * * * * * *           * * * * * * * * @21
Q3 ....... TOIF @M3 ->        . * * * * * * *           * * * * * * * * @20
Q4 ....... TOIF @M4 ->        . . * * * * * *           * * * * * * * * @19
Q5 ....... TOIF @M5 ->        . . . * * * * *           * * * * * * * * @18
Q6 ....... TOIF @M6 ->        . . . . * * * *           * * * * * * * * @17
Q7 ....... TOIF @M7 ->        . . . . . * * *           * * * * * * * * @16
Q8 ....... TOIF @M8 ->        . . . . . . * *           * * * * * * * * @15

Q16 ...... TOIF @M9 ->        . . . . . . . .           * * . . . . . . @3
Q15 ...... TOIF @M10->        . . . . . . . .           * * * . . . . . @4
Q14 ...... TOIF @M11->        . . . . . . . .           * * * * . . . . @5
Q13 ...... TOIF @M12->        . . . . . . . .           * * * * * . . . @6
Q12 ...... TOIF @M13->        . . . . . . . .           * * * * * * . . @7
Q11 ...... TOIF @M14->        . . . . . . . .           * * * * * * * . @8
Q10 ...... TOIF @M15->        . . . . . . . .           * * * * * * * * @9
Q9 ....... TOIF @M16->        . . . . . . . *           * * * * * * * * @10


INPUTS:

UPDN ..... INP  @2   ->       . * * * * * * *           * * * * * * * *
SHFTCNT .. INP  @11  ->       * * * * * * * *           * * * * * * * *
LTRT ..... INP  @14  ->       * * * * * * * *           * * * * * * * *
DATA ..... INP  @23  ->       * . . . . . . .           * . . . . . . .
                              Q Q Q Q Q Q Q Q           Q Q Q Q Q Q Q Q
                              1 2 3 4 5 6 7 8           1 1 1 1 1 1 1 9
                                                        6 5 4 3 2 1 0
```

## Table 4-3. Sample Utilization Report for EPB1400 (Part 1 of 6)

```
ALTERA Design Processor Utilization Report                          datacom.rpt
@(#) FIT Version 5.0     9/30/87 19:42:24 39.16
 ***** Design implemented successfully

CLIFF TONG
ALTERA CORPORATION
SEPTEMBER 1987
1.00
A
EPB1400
DATACOM TRANSMITTER
LogiCaps Schematic Capture Ver 1.6
Input files : datacom.sdf
ADP Options: Minimization = Yes,  Inversion Control = No,  LEF Analysis = Yes

OPTIONS: TURBO = ON, SECURITY = OFF
 ***** Externally connect signal "SYSCLK" to pins 7 and 14.

               EPB1400
               - - - - -
       Gnd  -|1       40|- RESERVED
       Gnd  -|2       39|- RESERVED
       Gnd  -|3       38|- RESERVED
       Gnd  -|4       37|- RESERVED
       Gnd  -|5       36|- /RD
       Gnd  -|6       35|- PORT7
    SYSCLK  -|7       34|- PORT6
    /RESET  -|8       33|- PORT5
       /CE  -|9       32|- PORT4
       Vcc  -|10      31|- GND
       GND  -|11      30|- GND
        A1  -|12      29|- PORT3
        A0  -|13      28|- PORT2
    SYSCLK  -|14      27|- PORT1
       Gnd  -|15      26|- PORT0
   RELDINT  -|16      25|- /WR
    SEROUT  -|17      24|- RESERVED
  RESERVED  -|18      23|- RESERVED
  RESERVED  -|19      22|- RESERVED
  RESERVED  -|20      21|- RESERVED
               - - - - -
```

## Table 4-3. Sample Utilization Report for EPB1400 (Part 2 of 6)

```
                                                                  datacom.rpt
**OUTPUTS**
                                                    FdBck
      Name   Pin  Resource  MCell  PTerms   |  Group  | Sync Clock

   RELDINT    16      JONF      2    2/ 8   |    1G   | .0102067
    SEROUT    17      RORF      3    4/ 8   |    1G   |     -


**BURIED REGISTERS**
                                                    FdBck
      Name   Pin  Resource  MCell  PTerms   |  Group  | Sync Clock

    COUNT0 ( 4)     NORF     18    2/ 8   |    2G   | .0102067
    COUNT1 ( 3)     NORF     17    3/ 8   |    2G   | .0102067
    COUNT2 ( 2)     NORF     16    4/ 8   |    2G   | .0102067
    COUNT3 ( 1)     NORF     15    5/ 8   |    2G   | .0102067
    COUNT4 (40)     NORF     14    2/ 8   |    2G   |     -
    COUNT5 (39)     NORF     13    3/ 8   |    2G   |     -
    COUNT6 (38)     NORF     12    4/ 8   |    2G   |     -
    COUNT7 (37)     NORF     11    5/ 8   |    2G   |     -
    SHIFT0 (24)     NORF     10    1/ 8   |    1G   |     -
    SHIFT1 (23)     NORF      9    4/ 8   |    1G   |     -
    SHIFT2 (22)     NORF      8    4/ 8   |    1G   |     -
    SHIFT3 (21)     NORF      7    4/ 8   |    1G   |     -
    SHIFT4 (20)     NORF      6    4/ 8   |    1G   |     -
    SHIFT5 (19)     NORF      5    4/ 8   |    1G   |     -
    SHIFT6 (18)     NORF      4    4/ 8   |    1G   |     -
    STREG0 (15)     NOTF      1    1/ 8   |    1G   |     -
    STREG1 ( 6)     NOTF     20    1/ 8   |    2G   |     -
    STREG2 ( 5)     NOTF     19    1/ 8   |    2G   |     -


**INPUTS**
                                                    FdBck
      Name   Pin  Resource  MCell  PTerms   |  Group  | Sync Clock

        A0    13       INP      -       -   |    -    |     -
        A1    12       INP      -       -   |    -    |     -
       /CE     9       INP      -       -   |    -    |     -
       /RD    36        RS      -       -   |    -    |     -
    /RESET     8       INP      -       -   |    -    |     -
    SYSCLK     7       CKX      -       -   |    -    |     -
       /WR    25        WS      -       -   |    -    |     -
```

## Table 4-3. Sample Utilization Report for EPB1400 (Part 3 of 6)

```
**Latches**                                                            datacom.rpt
                                                        FdBck
     Name   Pin   Resource  MCell    Ref#        |  Group  |  Sync Clock

   SHIFT7 (17)     LBUSO       3 L#5 (7)     |    1    | .0102067
   SHIFT6 (18)     LBUSO       4 L#5 (6)     |    1    | .0102067
   SHIFT5 (19)     LBUSO       5 L#5 (5)     |    1    | .0102067
   SHIFT4 (20)     LBUSO       6 L#5 (4)     |    1    | .0102067
   SHIFT3 (21)     LBUSO       7 L#5 (3)     |    1    | .0102067
   SHIFT2 (22)     LBUSO       8 L#5 (2)     |    1    | .0102067
   SHIFT1 (23)     LBUSO       9 L#5 (1)     |    1    | .0102067
   SHIFT0 (24)     LBUSO      10 L#5 (0)     |    1    | .0102067
  SERIAL7 (17)     RBUSI       - L#3 (7)     |    1    |     -
  SERIAL6 (18)     RBUSI       - L#3 (6)     |    1    |     -
  SERIAL5 (19)     RBUSI       - L#3 (5)     |    1    |     -
  SERIAL4 (20)     RBUSI       - L#3 (4)     |    1    |     -
  SERIAL3 (21)     RBUSI       - L#3 (3)     |    1    |     -
  SERIAL2 (22)     RBUSI       - L#3 (2)     |    1    |     -
  SERIAL1 (23)     RBUSI       - L#3 (1)     |    1    |     -
  SERIAL0 (24)     RBUSI       - L#3 (0)     |    1    |     -
   SCALE7 (37)     RBUSI       - L#1 (7)     |    2    |     -
   SCALE6 (38)     RBUSI       - L#1 (6)     |    2    |     -
   SCALE5 (39)     RBUSI       - L#1 (5)     |    2    |     -
   SCALE4 (40)     RBUSI       - L#1 (4)     |    2    |     -
   SCALE3  (1)     RBUSI       - L#1 (3)     |    2    |     -
   SCALE2  (2)     RBUSI       - L#1 (2)     |    2    |     -
   SCALE1  (3)     RBUSI       - L#1 (1)     |    2    |     -
   SCALE0  (4)     RBUSI       - L#1 (0)     |    2    |     -
    PORT7   35      BUSX       - L#2 (7)     |    2    |     -
    PORT6   34      BUSX       - L#2 (6)     |    2    |     -
    PORT5   33      BUSX       - L#2 (5)     |    2    |     -
    PORT4   32      BUSX       - L#2 (4)     |    2    |     -
    PORT3   29      BUSX       - L#2 (3)     |    2    |     -
    PORT2   28      BUSX       - L#2 (2)     |    2    |     -
    PORT1   27      BUSX       - L#2 (1)     |    2    |     -
    PORT0   26      BUSX       - L#2 (0)     |    2    |     -
   COUNT7 (37)     LBUSO      11 L#4 (7)     |    2    | .0102067
   COUNT6 (38)     LBUSO      12 L#4 (6)     |    2    | .0102067
   COUNT5 (39)     LBUSO      13 L#4 (5)     |    2    | .0102067
   COUNT4 (40)     LBUSO      14 L#4 (4)     |    2    | .0102067
   COUNT3  (1)     LBUSO      15 L#4 (3)     |    2    | .0102067
   COUNT2  (2)     LBUSO      16 L#4 (2)     |    2    | .0102067
   COUNT1  (3)     LBUSO      17 L#4 (1)     |    2    | .0102067
   COUNT0  (4)     LBUSO      18 L#4 (0)     |    2    | .0102067
```

## Table 4-3. Sample Utilization Report for EPB1400 (Part 4 of 6)

```
                                                                        datacom.rpt

**UNUSED RESOURCES**

                                                        FdBck
      Name  Pin  Resource  MCell  PTerms  | Group | Sync Clock

        -    5     INPUT     -       -    |   -   | .0102067
        -    6     INPUT     -       -    |   -   | .0102067
        -   15     INPUT     -       -    |   -   | .0102067


**PART UTILIZATION**

20/20  MacroCells (100%)
 6/ 9  Input Pins (66%)
       PTerms Used 38%
```

## Table 4-3. Sample Utilization Report for EPB1400 (Part 5 of 6)

```
Macrocell Interconnection Cross Reference                                datacom.rpt

FEEDBACKS:                                  M  M M M M M M M M M M L L  L  L  L
                        M M M M M M M M M 1  1 1 1 1 1 1 1 1 1 2 # #  #  #  #
                        1 2 3 4 5 6 7 8 9 0  1 2 3 4 5 6 7 8 9 0 1 2  3  4  5
STREG0 ... NOTF @M1 ->  . * * * * * * * * *  . . . . . . . . * * . .  .  .  . (15)
RELDINT .. JONF @M2 ->  . * . . . . . . . .  . . . . . . . . . . . .  .  .  . @16
SEROUT ... RORF @M3 ->  . . . . . . . . . .  . . . . . . . . . . . .  .  .  * @17
SHIFT6 ... NORF @M4 ->  . . * . . . . . . .  . . . . . . . . . . . .  .  .  * (18)
SHIFT5 ... NORF @M5 ->  . . . * . . . . . .  . . . . . . . . . . . .  .  .  * (19)
SHIFT4 ... NORF @M6 ->  . . . . * . . . . .  . . . . . . . . . . . .  .  .  * (20)
SHIFT3 ... NORF @M7 ->  . . . . . * . . . .  . . . . . . . . . . . .  .  .  * (21)
SHIFT2 ... NORF @M8 ->  . . . . . . * . . .  . . . . . . . . . . . .  .  .  * (22)
SHIFT1 ... NORF @M9 ->  . . . . . . . * . .  . . . . . . . . . . . .  .  .  * (23)
SHIFT0 ... NORF @M10->  . . . . . . . . * .  . . . . . . . . . . . .  .  .  * (24)

COUNT7 ... NORF @M11->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (37)
COUNT6 ... NORF @M12->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (38)
COUNT5 ... NORF @M13->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (39)
COUNT4 ... NORF @M14->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (40)
COUNT3 ... NORF @M15->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (1)
COUNT2 ... NORF @M16->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (2)
COUNT1 ... NORF @M17->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (3)
COUNT0 ... NORF @M18->  * . * * * * * * * *  * * * * * * * * * * . .  .  *  . (4)
STREG2 ... NOTF @M19->  . * * * * * * * * *  . . . . . . . . . . . .  .  .  . (5)
STREG1 ... NOTF @M20->  . * * * * * * * * *  . . . . . . . . * . . .  .  .  . (6)
                        S R S S S S S S S S  C C C C C C C C S S
                        T E E H H H H H H H  O O O O O O O O T T
                        R L R I I I I I I I  U U U U U U U U R R
                        E D O F F F F F F F  N N N N N N N N E E
                        G I U T T T T T T T  T T T T T T T T G G
                        0 N T 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0 2 1
                          T
```

## Table 4-3. Sample Utilization Report for EPB1400 (Part 6 of 6)

```
Macrocell Interconnection Cross Reference                          datacom.rpt

INPUTS:                                  M   M M M M M M M M M M L L   L   L   L
                       M M M M M M M M M 1   1 1 1 1 1 1 1 1 1 2 # #   #   #   #
                       1 2 3 4 5 6 7 8 9 0   1 2 3 4 5 6 7 8 9 0 1 2   3   4   5
SCALE3 ... RBUS @M15-> x x x x x x x x x x   . . . . * . . . . . . .   x   .   x (1)
SCALE2 ... RBUS @M16-> x x x x x x x x x x   . . . . . * . . . . . .   x   .   x (2)
SCALE1 ... RBUS @M17-> x x x x x x x x x x   . . . . . . * . . . . .   x   .   x (3)
SCALE0 ... RBUS @M18-> x x x x x x x x x x   . . . . . . . * . . . .   x   .   x (4)
/RESET ... INP  @8  -> * * * * * * * * * *   . . . . . . . . * * . .   .   .   .
/CE ...... INP  @9  -> . * . . . . . . . .   . . . . . . . . . . * .   *   *   *
A1 ....... INP  @12 -> . * . . . . . . . .   . . . . . . . . . . * .   *   *   *
A0 ....... INP  @13 -> . * . . . . . . . .   . . . . . . . . . . * .   *   *   *
SERIAL7 .. RBUS @M3 -> . . * . . . . . . .   x x x x x x x x x x x x   .   x   . (17)
SERIAL6 .. RBUS @M4 -> . . . * . . . . . .   x x x x x x x x x x x x   .   x   . (18)
SERIAL5 .. RBUS @M5 -> . . . . * . . . . .   x x x x x x x x x x x x   .   x   . (19)
SERIAL4 .. RBUS @M6 -> . . . . . * . . . .   x x x x x x x x x x x x   .   x   . (20)
SERIAL3 .. RBUS @M7 -> . . . . . . * . . .   x x x x x x x x x x x x   .   x   . (21)
SERIAL2 .. RBUS @M8 -> . . . . . . . * . .   x x x x x x x x x x x x   .   x   . (22)
SERIAL1 .. RBUS @M9 -> . . . . . . . . * .   x x x x x x x x x x x x   .   x   . (23)
SERIAL0 .. RBUS @M10-> . . . . . . . . . *   x x x x x x x x x x x x   .   x   . (24)
/WR ...... WS   @25 -> . . . . . . . . . .   . . . . . . . . . . * .   *   .   .
/RD ...... RS   @36 -> . * . . . . . . . .   . . . . . . . . . . . *   .   .   .
SCALE7 ... RBUS @M11-> x x x x x x x x x x   * . . . . . . . . . . .   x   .   x (37)
SCALE6 ... RBUS @M12-> x x x x x x x x x x   . * . . . . . . . . . .   x   .   x (38)
SCALE5 ... RBUS @M13-> x x x x x x x x x x   . . * . . . . . . . . .   x   .   x (39)
SCALE4 ... RBUS @M14-> x x x x x x x x x x   . . . * . . . . . . . .   x   .   x (40)
                       S R S S S S S S S S   C C C C C C C C S S
                       T E E H H H H H H H   O O O O O O O O T T
                       R L R I I I I I I I   U U U U U U U U R R
                       E D O F F F F F F F   N N N N N N N N E E
                       G I U T T T T T T T   T T T T T T T T G G
                       O N T 6 5 4 3 2 1 0   7 6 5 4 3 2 1 0 2 1
                         T
```

## Table 4-4. Sample Utilization Report for EP1800J (Part 1 of 6)

```
ALTERA Design Processor Utilization Report                     hdxcntr.rpt
@(#) FIT Version 5.0     9/30/87 19:42:24 39.16
 ***** Design implemented successfully

BOB DUNCAN
ALTERA
SEPT 30, 1987
1.00
A
EP1800J
MANCHESTER UART HDXCNTR
Input files : hdxcntr.adf
ADP Options: Minimization = Yes,  Inversion Control = No,  LEF Analysis = Yes

OPTIONS: TURBO = ON, SECURITY = OFF

               T
               A        R  R  R  R                                      P
               K  S  P  C  C  C  C                                   L  R
               E  T  A  V  V  V  V        B  B  B  B  B  B  O  E
               D  A  R  B  B  B  B  L     I  I  I  I  I  I  C  S
               A  T  I  I  I  I  I  O  G  T  T  T  T  T  T  K  C
               T  U  T  T  T  T  T  C  N  1  1  1  1  1  1  I  L
               A  S  Y  0  1  2  3  K  D  0  1  2  3  4  5  N  2
             ---------------------------------------------------_
            /  9  8  7  6  5  4  3  2  1 68 67 66 65 64 63 62 61   |
    BIT4IN | 10                                               60 | PRESCL1
        v3 | 11                                               59 | PRESCL3
        v2 | 12                                               58 | v4
        v1 | 13                                               57 | BIT0
    DATAIN | 14                                               56 | BIT5IN
  TRANSMIT | 15                                               55 | BIT6IN
     RESET | 16                                               54 | BIT7IN
    BIT9IN | 17                                               53 | BIT8IN
       Vcc | 18                                               52 | Vcc
   XCOMDAT | 19                                               51 | CLOCK
   BIT15IN | 20                                               50 | BIT10IN
   BIT14IN | 21                                               49 | BIT11IN
   BIT13IN | 22                                               48 | BIT12IN
    DETECT | 23                                               47 | BIT0IN
      BIT1 | 24                                               46 | BIT1IN
      BIT3 | 25                                               45 | BIT2IN
      BIT9 | 26                                               44 | BIT3IN
           |_ 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43  _|
             -----------------------------------------------------
               B  B  B  B  B  B  C  G  G  N  N  M  M  X  M  X  X
               I  I  I  I  I  I  O  n  N  O  O  P  P  P  A  C  C
               T  T  T  T  T  T  M  d  D  D  D  X  X  A  N  N  N
               2  4  5  6  7  8  D        E  E  2  1  R  C  T  T
                                 A        3  2        I  H  5  4
                                 T        3  2        T  O
                                                      Y  U
                                                         T
```

## Table 4-4. Sample Utilization Report for EP1800J (Part 2 of 6)

```
                                                                     hdxcntr.rpt
**OUTPUTS**
                                                        FdBck
        Name   Pin  Resource  MCell  PTerms      |  Group |  Sync Clock  |  OE Group

        BIT0    57    ROIF      37    5/ 8       |   4G   |      -       |  VCC
        BIT1    24    ROIF      14    5/ 8       |   2G   |      -       |  VCC
        BIT2    27    RORF      17    5/ 8       |   2    |      -       |  VCC
        BIT3    25    ROIF      15    4/ 8       |   2G   |      -       |  VCC
        BIT4    28    RORF      18    4/ 8       |   2    |      -       |  VCC
        BIT5    29    RORF      19    4/ 8       |   2    |      -       |  VCC
        BIT6    30    RORF      20    4/ 8       |   2    |      -       |  VCC
        BIT7    31    RORF      21    4/ 8       |   2    |      -       |  VCC
        BIT8    32    RORF      22    4/ 8       |   2    |      -       |  VCC
        BIT9    26    ROIF      16    4/ 8       |   2G   |      -       |  VCC
       BIT10    68    RORF      48    4/ 8       |   4    |      -       |  VCC
       BIT11    67    RORF      47    4/ 8       |   4    |      -       |  VCC
       BIT12    66    RORF      46    4/ 8       |   4    |      -       |  VCC
       BIT13    65    RORF      45    4/ 8       |   4    |      -       |  VCC
       BIT14    64    RORF      44    4/ 8       |   4    |      -       |  VCC
       BIT15    63    RORF      43    4/ 8       |   4    |      -       |  VCC
      COMDAT    33    RORF      23    2/ 8       |   2    |      -       |  VCC
      DETECT    23    COIF      13    8/ 8       |   2G   |      -       |  VCC
        LOCK     2    COIF       1    1/ 8       |   1    |      -       |  VCC
    MANCHOUT    41    CONF      30    6/ 8       |   3    |      -       |  VCC
        MPX1    39    COIF      28    8/ 8       |   3    |      -       |  VCC
        MPX2    38    COIF      27    8/ 8       |   3    |      -       |  VCC
      NODE22    37    COCF      26    1/ 8       |   3    |      -       |  VCC
      NODE33    36    COCF      25    3/ 8       |   3    |      -       |  VCC
      PARITY     7    RORF       6    7/ 8       |   1    |      -       |  VCC
     PRESCL1    60    ROIF      40    1/ 8       |   4G   |      -       |  VCC
     PRESCL2    61    RORF      41    2/ 8       |   4    |      -       |  VCC
     PRESCL3    59    ROIF      39    2/ 8       |   4G   |      -       |  VCC
     RCVBIT0     6    TOTF       5    5/ 8       |   1    |      -       |  VCC
     RCVBIT1     5    TOTF       4    2/ 8       |   1    |      -       |  VCC
     RCVBIT2     4    TOTF       3    2/ 8       |   1    |      -       |  VCC
     RCVBIT3     3    TOTF       2    2/ 8       |   1    |      -       |  VCC
      STATUS     8    RORF       7    4/ 8       |   1    |      -       |  VCC
    TAKEDATA     9    CONF       8    1/ 8       |   1    |      -       |  VCC
          v1    13    ROIF      12    3/ 8       |   1G   |      -       |  VCC
          v2    12    ROIF      11    6/ 8       |   1G   |      -       |  VCC
          v3    11    ROIF      10    4/ 8       |   1G   |      -       |  VCC
          v4    58    ROIF      38    5/ 8       |   4G   |      -       |  VCC
       XCNT4    43    RORF      32    2/ 8       |   3    |  CLOCK       |  VCC
       XCNT5    42    RORF      31    3/ 8       |   3    |  CLOCK       |  VCC
     XPARITY    40    RORF      29    6/ 8       |   3    |  CLOCK       |  VCC
```

## Table 4-4. Sample Utilization Report for EP1800J (Part 3 of 6)

```
                                                                      hdxcntr.rpt
**BURIED REGISTERS**
                                                FdBck
     Name  Pin  Resource  MCell  PTerms  | Group | Sync Clock |  OE Group

     TEST (10)     NOCF       9    1/ 8  |   1   |     -      |     -
    XCNT0 (44)     NORF      33    5/ 8  |   3   | CLOCK      |     -
    XCNT1 (45)     NORF      34    2/ 8  |   3   | CLOCK      |     -
    XCNT2 (46)     NORF      35    2/ 8  |   3   | CLOCK      |     -
    XCNT3 (47)     NORF      36    3/ 8  |   3   | CLOCK      |     -

**INPUTS**
                                                FdBck
     Name  Pin  Resource  MCell  PTerms  | Group | Sync Clock |  OE Group

   BIT0IN   47      INP       -      -   |   -   |     -      |     -
   BIT1IN   46      INP       -      -   |   -   |     -      |     -
   BIT2IN   45      INP       -      -   |   -   |     -      |     -
   BIT3IN   44      INP       -      -   |   -   |     -      |     -
   BIT4IN   10      INP       -      -   |   -   |     -      |     -
   BIT5IN   56      INP       -      -   |   -   |     -      |     -
   BIT6IN   55      INP       -      -   |   -   |     -      |     -
   BIT7IN   54      INP       -      -   |   -   |     -      |     -
   BIT8IN   53      INP       -      -   |   -   |     -      |     -
   BIT9IN   17      INP       -      -   |   -   |     -      |     -
  BIT10IN   50      INP       -      -   |   -   |     -      |     -
  BIT11IN   49      INP       -      -   |   -   |     -      |     -
  BIT12IN   48      INP       -      -   |   -   |     -      |     -
  BIT13IN   22      INP       -      -   |   -   |     -      |     -
  BIT14IN   21      INP       -      -   |   -   |     -      |     -
  BIT15IN   20      INP       -      -   |   -   |     -      |     -
    CLOCK   51      CKR       -      -   |   3   |     -      |     -
   DATAIN   14      INP       -      -   |   -   |     -      |     -
   LOCKIN   62      INP      42    0/ 8  |   4   |     -      |     -
    RESET   16      INP       -      -   |   -   |     -      |     -
 TRANSMIT   15      INP       -      -   |   -   |     -      |     -
  XCOMDAT   19      INP       -      -   |   -   |     -      |     -


**UNUSED RESOURCES**
                                                FdBck
     Name  Pin  Resource  MCell  PTerms  | Group | Sync Clock |  OE Group

        -   34    MCELL      24      8   |   2   |     -      |     -


**PART UTILIZATION**

46/47  MacroCells (97%)
21/21  Input Pins (100%)
       PTerms Used 47%
```

## Table 4-4. Sample Utilization Report for EP1800J (Part 4 of 6)

```
Macrocell Interconnection Cross Reference                          hdxcntr.rpt

FEEDBACKS:                                    M M M   M M M M M M M M M M M   M M M M M M M M M M M M   M M M M M M M M M M M
                              M M M M M M M M M 1 1 1   1 1 1 1 1 1 1 2 2 2 2   2 2 2 2 2 3 3 3 3 3 3 3   3 3 3 4 4 4 4 4 4 4 4
                              1 2 3 4 5 6 7 8 9 0 1 2   3 4 5 6 7 8 9 0 1 2 3   5 6 7 8 9 0 1 2 3 4 5 6   7 8 9 0 1 3 4 5 6 7 8
LOCK ..... COIF @M1  -> . . . . . . . . . * * *   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @2
RCVBIT3 .. TOTF @M2  -> * . . . * . * * . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @3
RCVBIT2 .. TOTF @M3  -> * * . . * . * * . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @4
RCVBIT1 .. TOTF @M4  -> * * * . * . * * . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @5
RCVBIT0 .. TOTF @M5  -> * * * * * . * * . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @6
PARITY ... RORF @M6  -> . . . . . * * . . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @7
STATUS ... RORF @M7  -> . . . . . . * . . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @8
TAKEDATA . CONF @M8  -> . . . . . . . . . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x @9
TEST ..... NOCF @M9  -> . * * * * * * . . . . .   x x x x x x x x x x x   x x x x x x x x x x x x   x x x x x x x x x x x (10)
v3 ....... ROIF @M10-> . * * * * . . * * * * *   . * * * * * * * * * .   . . . . . . . . . . . .   * * . . . * * * * * * @11
v2 ....... ROIF @M11-> . * * * * . . * * * * *   . * * * * * * * * * .   . . . . . . . . . . . .   * * . . . * * * * * * @12
v1 ....... ROIF @M12-> . * * * * . . * * * * .   . * * * * * * * * * .   . . . . . . . . . . . .   * * . . . * * * * * * @13

DETECT ... COIF @M13-> * * * * * * * . . . . .   . * * * * * * * * * *   . . . . . . . . . . . .   * . . . . * * * * * * @23
BIT1 ..... ROIF @M14-> . . . . . * . . . . . .   * * . . * . . . . . .   . . . . . . . . . . . .   . . . . . . . . . . . @24
BIT3 ..... ROIF @M15-> . . . . . * . . . . . .   * . * . * * . . . . .   . . . . . . . . . . . .   . . . . . . . . . . . @25
BIT9 ..... ROIF @M16-> . . . . . . . . . . . .   * . . * . . . . . . .   . . . . . . . . . . . .   . . . . . . . . . . * @26
BIT2 ..... RORF @M17-> x x x x x x x x x x x x   * . * . * . . . . . .   x x x x x x x x x x x x   x x x x x x x x x x x @27
BIT4 ..... RORF @M18-> x x x x x x x x x x x x   * . . . . * * . . . .   x x x x x x x x x x x x   x x x x x x x x x x x @28
BIT5 ..... RORF @M19-> x x x x x x x x x x x x   * . . . . . * * . . .   x x x x x x x x x x x x   x x x x x x x x x x x @29
BIT6 ..... RORF @M20-> x x x x x x x x x x x x   * . . . . . . * * . *   x x x x x x x x x x x x   x x x x x x x x x x x @30
BIT7 ..... RORF @M21-> x x x x x x x x x x x x   * . . . . . . . * * .   x x x x x x x x x x x x   x x x x x x x x x x x @31
BIT8 ..... RORF @M22-> x x x x x x x x x x x x   * . . * . . . . . * .   x x x x x x x x x x x x   x x x x x x x x x x x @32
COMDAT ... RORF @M23-> x x x x x x x x x x x x   . . . . . . . . . . *   x x x x x x x x x x x x   x x x x x x x x x x x @33
```

## Table 4-4. Sample Utilization Report for EP1800J (Part 5 of 6)

```
NODE33 ... COCF @M25-> x x x x x x x x x x x x  x x x x x x x x x x x  . . . . . * . . . . . .  x x x x x x x x x x x @36
NODE22 ... COCF @M26-> x x x x x x x x x x x x  x x x x x x x x x x x  . . . . . . * . * . . *  x x x x x x x x x x x @37
MPX2 ..... COIF @M27-> x x x x x x x x x x x x  x x x x x x x x x x x  * . . . * . . . . . . .  x x x x x x x x x x x @38
MPX1 ..... COIF @M28-> x x x x x x x x x x x x  x x x x x x x x x x x  * . . . * . . . . . . .  x x x x x x x x x x x @39
XPARITY .. RORF @M29-> x x x x x x x x x x x x  x x x x x x x x x x x  * . . . * . . . . . . .  x x x x x x x x x x x @40
MANCHOUT . CONF @M30-> x x x x x x x x x x x x  x x x x x x x x x x x  . . . . . . . . . . . .  x x x x x x x x x x x @41
XCNT5 .... RORF @M31-> x x x x x x x x x x x x  x x x x x x x x x x x  * * * * . * * * * * * *  x x x x x x x x x x x @42
XCNT4 .... RORF @M32-> x x x x x x x x x x x x  x x x x x x x x x x x  * * * * . * * * * * * *  x x x x x x x x x x x @43
XCNT0 .... NORF @M33-> x x x x x x x x x x x x  x x x x x x x x x x x  . . . . * * * * * * * *  x x x x x x x x x x x (44)
XCNT1 .... NORF @M34-> x x x x x x x x x x x x  x x x x x x x x x x x  * * * * . * * * * * * *  x x x x x x x x x x x (45)
XCNT2 .... NORF @M35-> x x x x x x x x x x x x  x x x x x x x x x x x  * * * * . * * * * * * *  x x x x x x x x x x x (46)
XCNT3 .... NORF @M36-> x x x x x x x x x x x x  x x x x x x x x x x x  * * * * . * * * * * * *  x x x x x x x x x x x (47)

BIT0 ..... ROIF @M37-> . . . . . . * . . . . .  * * . . . . . . . . .  . . . . . . . . . . . .  * . . . . . . . . . . @57
v4 ....... ROIF @M38-> . . . . . . . . * * * *  . . . . . . . . . . .  . . . . . . . . . . . .  . * . . . . . . . . . @58
PRESCL3 .. ROIF @M39-> . . . . . . . . . . . .  . . . . . . . . . . .  . . . . * . * * * * * *  . . * . * . . . . . . @59
PRESCL1 .. ROIF @M40-> . . . . . . . . . . . .  . . . . . . . . . . .  . . . . * . * * * * * *  . . * * * . . . . . . @60
PRESCL2 .. RORF @M41-> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . . * . * . . . . . . @61
BIT15 .... RORF @M43-> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . . . . . * . . . . . @63
BIT14 .... RORF @M44-> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . . . . . * * . . . . @64
BIT13 .... RORF @M45-> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . . . . . . * * . . . @65
BIT12 .... RORF @M46-> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . . . . . . . * * . . @66
BIT11 .... RORF @M47-> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . . . . . . . . * * . @67
BIT10 .... RORF @M48-> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . . . . . . . . . * * @68
                       L R R R R P S T T v v v  D B B B B B B B B B C  N N M M X M X X X X X X  B v P P P B B B B B B
                       O C C C C A T A E 3 2 1  E I I I I I I I I I O  O O P P P A C C C C C C  I 4 R R R I I I I I I
                       C V V V V R A K S        T T T T T T T T T T M  D D X X A N N N N N N N  T   E E E T T T T T T
                       K B B B B I T E T        E 1 3 9 2 4 5 6 7 8 D  E E 2 1 R C T T T T T T  0   S S S 1 1 1 1 1 1
                         I I I I T U D          C                   A  3 2     I H 5 4 0 1 2 3      C C C 5 4 3 2 1 0
                         T T T T Y S A          T                   T  3 2     T O                  L L L
                         3 2 1 0     T                                         Y U                  3 1 2
                                     A                                           T
```

## Table 4-4. Sample Utilization Report for EP1800J (Part 6 of 6)

```
Macrocell Interconnection Cross Reference                              hdxcntr.rpt

INPUTS:                                  M M M  M M M M M M M M M M M  M M M M M M M M M M M M  M M M M M M M M M M M
                       M M M M M M M M M 1 1 1  1 1 1 1 1 1 1 2 2 2 2  2 2 2 2 2 3 3 3 3 3 3 3  3 3 3 4 4 4 4 4 4 4 4
                       1 2 3 4 5 6 7 8 9 0 1 2  3 4 5 6 7 8 9 0 1 2 3  5 6 7 8 9 0 1 2 3 4 5 6  7 8 9 0 1 3 4 5 6 7 8
BIT4IN ... INP  @10 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . . M9
DATAIN ... INP  @14 -> . . . . . * * . . * * *  . . . . . . . . . . .  . . . . . . . . . . . .  * * . . . . . . . . .
TRANSMIT . INP  @15 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . . * . * * * * * *  . . . . . . . . . . .
RESET .... INP  @16 -> . * * * * * * . . . . .  . * * * * * * * * * *  . . . . * . * * * * * *  * . . . . * * * * * *
BIT9IN ... INP  @17 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
XCOMDAT .. INP  @19 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . . . * . . . . . .  . . . . . . . . . . .
BIT15IN .. INP  @20 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
BIT14IN .. INP  @21 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
BIT13IN .. INP  @22 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
BIT3IN ... INP  @44 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . . M33
BIT2IN ... INP  @45 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . . M34
BIT1IN ... INP  @46 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . . M35
BIT0IN ... INP  @47 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . . M36
BIT12IN .. INP  @48 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
BIT11IN .. INP  @49 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
BIT10IN .. INP  @50 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
BIT8IN ... INP  @53 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . . * . . . . . . . .  . . . . . . . . . . .
BIT7IN ... INP  @54 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . .
BIT6IN ... INP  @55 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . .
BIT5IN ... INP  @56 -> . . . . . . . . . . . .  . . . . . . . . . . .  . . * . . . . . . . . .  . . . . . . . . . . .
LOCKIN ... INP  @62 -> x x x x x x x x x x x x  x x x x x x x x x x x  x x x x x x x x x x x x  . * . . . . . . . . . M42
                       L R R R R P S T T v v v  D B B B B B B B B B C  N N M M X M X X X X X X  B v P P P B B B B B B
                       O C C C C A T A E 3 2 1  E I I I I I I I I I O  O O P P P A C C C C C C  I 4 R R R I I I I I I
                       C V V V V R A K S        T T T T T T T T T T M  D D X X A N N N N N N N  T   E E E T T T T T T
                       K B B B B I T E T        E 1 3 9 2 4 5 6 7 8 D  E E 2 1 R C T T T T T T  O   S S S 1 1 1 1 1 1
                         I I I I T U D          C                   A  3 2     I H 5 4 0 1 2 3      C C C 5 4 3 2 1 0
                         T T T T Y S A          T                   T  3 2     T O                  L L L
                         3 2 1 0     T                                         Y U                  3 1 2
                                     A                                           T
```

# A+PLUS Messages

This section contains separate lists of all error, information, and warning messages generated by A+PLUS, the Altera Design Processor, and individual ADP modules. Each entry describes the likely cause of the message and, when appropriate, gives suggestions for corrective action.

☞ Messages generated by optional software packages are included in the documentation for each respective package.

## Message Format

During design processing, the Altera Design Processor displays information messages indicating that a particular ADP module has completed execution without error. All information messages, as well as error and warning messages, are also logged to a file called **ADP.LOG** in the current directory.

Messages appear in the following format:

**<prefix> <message text>**

Each prefix begins with *****ERR-**, *****INFO-**, or *****WARN-**, and is followed by a several letters that indicate which portion of the software generated the message. These prefixes are:

| | |
|---|---|
| **ANA** | LEF Analyzer module |
| **APLUS** | A+PLUS |
| **ADP** | Altera Design Processor |
| **ASM** | Assembler module |
| **EXP** | Expander module |
| **FIT** | Fitter module |
| **FLT** | Flattener module |
| **LIB** | Library module |
| **MIN** | Minimizer module |
| **XLT** | Translator module |

In the following pages, Error, Information, and Warning messages are listed in three separate groups; within groups, messages are listed alphabetically by the **<message text>** (not the **<prefix>**). Each entry describes the likely cause of the message and, when appropriate, gives suggestions for corrective action.

Many messages are preceded by information indicating the source of the message. This information may consist of one or more of the following indicators:

- **Input pin name: <pin name>**
- **Output pin name: <pin name>**
- **Pin #: <pin #>**
- **Primitive: <primitive name>**
- **Node name: <node name>**
- **Bit: <bit number>, Signal: <name>**
- **File: <filename>**
- **Line: <ADF line #>**
- **Expected Part: <EPLD name>**
- Display of ADF line with a "v" above the source of the message.

☞ If the **<node name>** specified is of the form **.pxxxyyy**, where **p**, **x**, and **y** are all digits, this indicates that the node is located at coordinates **X** = **xxx** and **Y** = **yyy** in the (p+1)th file submitted to the ADP. (Applies only to files created with LogiCaps.)

The notation used in this section uses #'s to represent decimal numbers. It also contains variables which are enclosed in <>'s. For example:

| | |
|---|---|
| **<name>** | represents the name of a library, macrocell, MacroFunction, pin, primitive, section, signal, etc. |
| **<str>** | represents any string of characters (e.g., the left-hand side of an equation). |

Questions regarding A+PLUS messages should be directed to:

Altera Corporation
Applications Dept.
3525 Monroe Street
Santa Clara, CA 95051

Telephone:

| | |
|---|---|
| USA | (800) 821-8124 |
| elsewhere | (408) 984-2805 x102 |

**ERR-ASM- Can't create JEDEC file <filename>**

CAUSE: The disk is full, write-protected, or corrupted, so the Assembler can't create the JEDEC file. Also, there may not be enough memory available to open the file.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and that the disk is not corrupted or write-protected.

**ERR-XLT- Can't find an appropriate EPLD**

CAUSE: You used the automatic part selection feature, but your design will not fit any known Altera EPLD.

ACTION: Consult the ***Altera Data Book*** for information on the resources available for each EPLD and change your design to fit an Altera EPLD.

**ERR-FLT- Can't find any MacroFunction libraries**

CAUSE: The Flattener could not find any user-defined MacroFunction libraries or **MACRO.LIB**.

ACTION: If you are using MacroFunction libraries created with **ADLIB**, make sure their full pathnames are included in the **set ADLIB** string. If not, you may need to (re)install **MACRO.LIB**.

**ERR-FLT- Can't find MacroFunction <name> in any library**

CAUSE: The Flattener could not find MacroFunction **<name>** in **MACRO.LIB** or any of the libraries you specified with the **set ADLIB** string.

ACTION: Check whether the MacroFunction name is misspelled. If it is a MacroFunction created with **ADLIB** and is spelled correctly, you must add (or correct) the full pathname of the library that contains it to the **set ADLIB** string. (You may use up to 15 libraries.)

**ERR-XLT- Can't mix latched and unlatched inputs/feedbacks**

CAUSE: You have used a combination of latched and unlatched inputs and pin feedbacks. Only one type is allowed on a single EP1210.

ACTION: Either use all **LINP**, **ROLF**, and **COLF** primitives, or all **INP**, **COIF**, **ROIF** and **TOIF** primitives in a single design.

ACTION: Check whether the name you entered at the **File Name(s)** prompt or in the **set ADLIB** string has the correct path/filename, or make sure that at least 128K of disk space is available.

### Can't open <str> LEF

CAUSE: The disk is full, write-protected, or corrupted, so the Minimizer can't open the specified file (i.e., an input or output LEF). Also, there may not be enough memory available to open the file.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and that the disk is not corrupted or write-protected.

### Can't open ADF

CAUSE: The Translator could not find the ADF, or encountered an I/O error when trying to open it.

ACTION: Check the path/filename and ensure that your disk is not corrupted. If you are combining more than 3 ADF files, be sure that **FILES=20** and **BUFFERS=12** have been included in your **CONFIG.SYS** file.

### Can't open temporary file

CAUSE: The disk is full, write-protected, or corrupted, so the ADP can't open a temporary file needed for design processing. Also, there may not be enough memory available to open the file.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and that the disk is not corrupted or write-protected. If you have more than 3 ADF files, be sure that **FILES=20** has been included in your **CONFIG.SYS** file.

### Can't reopen <str>

CAUSE: The disk is full, write-protected, or corrupted, so the Minimizer can't reopen the specified file (i.e., a logic equation file (LEF) or temporary file) which is needed for design processing. Also, there may not be enough memory available to open the file.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and that the disk is not corrupted or write-protected.

## ERR-MIN- Can't reopen <str>

CAUSE: The disk is full, write-protected, or corrupted, so the Minimizer can't reopen the specified file (i.e., a logic equation file (LEF) or temporary file) which is needed for design processing. Also, there may not be enough memory available to open the file.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and that the disk is not corrupted or write-protected.

## ERR-XLT- Can't select an EPLD because of pin assignments

CAUSE: You used the automatic part selection feature and have also specified a pin assignment(s). The Translator cannot select a part when pin assignments are specified.

ACTION: Remove pin assignments or specify an EPLD instead of using automatic part selection.

## ERR-XLT- Can't tie primitive outputs together

CAUSE: The Translator encountered a node that is connected to more than one primitive output (i.e., used on the left-hand side of more than one primitive statement). This type of "Wire **OR**" connection is not permitted.

ACTION: Use each node name only once on the left-hand side of a primitive statement. (Break all "Wire **OR**" connections.)

## ERR-XLT- Circle of NOTs

CAUSE: Your design contains **NOT** gates that are connected in a loop. Such a circuit is useless and invalid.

ACTION: Correct the design logic and resubmit the file to the ADP.

## ERR-XLT- Clk, Rs, and Ws must be fed only by an input pin or the default value

CAUSE: You have connected the Clock, Read Strobe or Write Strobe input to a Bus I/O primitive to something other than an input pin or the default value. These inputs are optional, but if they are used, they must be fed directly from an input pin or the default value (i.e., **VCC**, **VCC**, or **GND**, respectively). All other configurations are prohibited.

ACTION: Connect the **Clk**, **Rs**, or **Ws** input to an input pin or to **VCC**, **VCC**, or **GND**, respectively.

## ERR-XLT- Clock configuration not available for EPLD

CAUSE: This EPLD does not support the requested combination of clock polarities, clocks feeding the logic, or the number of pins used for clock inputs.

ACTION: Refer to *Appendix A* for legal clocking configurations for all Altera EPLDs.

## ERR-XLT- Clock input required for I/O primitive

CAUSE: You have not provided a clock input (**Clk**), which is required for all I/O primitives.

ACTION: Specify a **Clk** input for the primitive.

## ERR-XLT- Clock inputs cannot feed logic primitives

CAUSE: Since the clock line does not run through the array on this EPLD, you can't tie it to any logic primitives. All branches of the signal that drives a clock input in this EPLD must terminate at a clock input and cannot feed the logic array.

ACTION: Use asynchronous clocking (i.e., a non-clock pin) to feed both logic and the asynchronous clocks of registers or use an EPLD which permits the clock to feed the logic array.

## ERR-XLT- Clock inputs must connect directly to INP

CAUSE: This EPLD does not support clock inputs driven by logic (i.e., asynchronous clocking). Clock inputs must come from a dedicated clock pin.

ACTION: Tie clock inputs directly to the output of an **INP** primitive.

## ERR-XLT- Clock, Read or Write Strobe conflict

CAUSE: You have connected the same dedicated **Clk**, **Rs**, or **Ws** pin to two different types of inputs in Bus I/O primitives.

ACTION: Connect each **Clk**, **Rs**, and **Ws** input to the dedicated **Clk**, **Rs**, and **Ws** input pins, or to the default values **VCC**, **GND**, and **GND**, respectively. (Refer to *Appendix A* for legal clocking configurations.)

**ERR-XLT- Different pin assignments for dual I/O feedback pin**

CAUSE: You have assigned pin numbers in the Inputs Section and the Outputs Section that are different. When dual I/O feedback is used, both pin assignments must be the same.

ACTION: Ensure that the pin is assigned to the same pin number in both sections.

**ERR-XLT- Dual I/O feedback not available for this EPLD**

CAUSE: This EPLD does not support dual I/O feedback. This message may occur if you have mistakenly assigned the same pin name to both an input and an output pin.

ACTION: Use an EP1800 or EPB1400 (BUSTER) EPLD to implement dual I/O feedback or check pin assignments.

**ERR-XLT- Duplicate input pin name**

CAUSE: The same name was assigned to more than one input pin.

ACTION: Ensure that all input pin names are unique.

**ERR-XLT- Duplicate output pin name**

CAUSE: The same name was assigned to more than one output pin.

ACTION: Ensure that all output pin names are unique.

**ERR-XLT- Duplicate pin numbers**

CAUSE: You have assigned the same pin number to more than one pin name.

ACTION: Assign a different pin number to each unique pin name.

**ERR-XLT- Either Clk or Ole is required**

CAUSE: You have not provided a Clock (Clk) or Output Latch Enable (Ole) input to an LBUSO primitive.

ACTION: The Clk input is optional. If it is connected, it must be connected directly to a pin or VCC (output enabled). If it is connected to a pin, the Ole will default to VCC. If the Clk is left unconnected, logic must be connected to the Ole, which will have full control of the flipflop.

**ERR-XLT- Either Rs or Oe is required**

CAUSE: You have not provided an Rs or Oe input to a BUSX primitive.

ACTION: The **Rs** input is optional. If it is connected, it must be connected directly to a pin or **GND**. If it is connected to a pin, the default for Output Enable (**Oe**) is **VCC** (Output enabled). If the **Rs** is left unconnected, logic must be connected to the **Oe**, which will have full control of the flipflop.

**ERR-XLT- Either Ws or We is required**

CAUSE: You have not provided a **Ws** or **We** input to this primitive.
ACTION: The **Ws** input to the **LBUSI**, **LINP8**, **RBUSI**, and **RINP8** primitives is optional. If the **Ws** is connected, it must be connected directly to a pin or **GND**. If it is connected to a pin, the default for Write Enable (**We**) is **VCC** (output enabled). If the **Ws** is left unconnected, logic must be connected to the **We**, which will have full control of the flipflop.

**ERR-XLT- EPLD names don't match**

CAUSE: When multiple ADFs were combined, different Altera target EPLDs were specified in the files.
ACTION: Ensure that all ADFs used in a single design specify the same target EPLD.

**ERR-MIN- Equation has been minimized away**

CAUSE: The Boolean reduction module removes product terms which contain both a variable and its complement. If this occurs in the only remaining product term of an equation, this message is generated.
ACTION: Remove subcircuits that reduce to this form from the input design file.

**ERR-MIN- Equation is too big to minimize**

CAUSE: The equation presented to the Minimizer was too big to be encoded and placed into the Minimizer's internal tables.
ACTION: Try redesigning the circuit to use fewer levels of logic.

**ERR-XLT- Equation missing right-hand side**

CAUSE: There is no Boolean expression on the right-hand side of an equation.
ACTION: Ensure that all entries in the ADF's Equations Section have at least one node name (followed by a semicolon [;]) on the right-hand side of the equals sign (=).

**ERR-EXP- Equation will expand too large**
**ERR-EXP- Left-hand side of equation is <signal name>**
**ERR-EXP- Can't expand equation**

CAUSE: These three messages always occur together. The equation for which the left-hand side is **<signal name>** expands beyond the limit accepted by the Expander.
ACTION: Use smaller, intermediate equations to describe the function.

**ERR-XLT- Expected <character>**

CAUSE: The Translator was expecting to find the specified character (i.e., a ;, :, (, ), or =) at this point in the file.
ACTION: Look for syntax errors in the ADF.

**ERR-XLT- Expected keyword**

CAUSE: The keyword **INPUTS:**, **OUTPUTS:**, or **NETWORK:** was expected as the next symbol.
ACTION: Ensure that there are no blank spaces, tabs, or unrecognizable characters before a keyword.

**ERR-XLT- Expected option**

CAUSE: The ADF contains the **OPTIONS:** keyword but no option follows it.
ACTION: Specify **TURBO** and/or **SECURITY** as the option, then set each option to **ON** or **OFF** (e.g., **TURBO = ON**).

**ERR-XLT- Feedback required for I/O primitive**

CAUSE: All I/O primitives except **CONF**, **JONF**, **RONF**, **SONF**, and **TONF** require a name for the feedback node.
ACTION: Include a feedback node name in the I/O primitive statement.

**ERR-FIT-**
**ERR-XLT- I/O error -- can't <str> PDF**

CAUSE: The Part Description File (of the format **<EPLD#>.PDF**) can't be found anywhere on your DOS path, can't be opened/read because available memory is insufficient or the disk is full or corrupted, or the PDF is corrupted.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, that your path includes the **APLUS** directory, and that all PDF files are in the **APLUS** directory. If the message persists, reinstall the PDFs from the distribution diskette.

**ERR-ASM- I/O error -- JDF <filename>**

CAUSE: The specified JEDEC Description File can't be found anywhere on your DOS path, can't be opened/read because available memory is insufficient or the disk is full or corrupted, or the JDF is corrupted.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, that your path includes the **APLUS** directory, and that all JDF files are in the **APLUS** directory. If the message persists, reinstall the JDFs from the distribution diskette.

**ERR-EXP-**
**ERR-FIT- I/O error -- LEF**

CAUSE: This message may have several causes. For example, available memory/disk space may be insufficient; the diskette containing the file is missing, write-protected or corrupted; or A+PLUS has been installed incorrectly.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and check whether the diskette is write-protected, corrupted, or not in the drive. If this message follows an **Internal error** message, contact Altera Applications.

**ERR-FIT- I/O error -- Report File**

CAUSE: The disk is full or corrupted, or there has been a disk I/O error, so the Fitter can't read, write, or open the Report File (**.RPT**). Also, there may not be enough memory available to open the file.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and that the disk is not corrupted.

**ERR-FIT- I/O error -- temporary file**

CAUSE: The disk is full or corrupted, or there has been a disk I/O error, so the Fitter can't use a temporary file needed for design processing. Also, there may not be enough memory available to open the file.

ACTION: Ensure that you have 128K of disk space and enough memory to run A+PLUS, and that the disk is not corrupted.

## ERR-XLT- ILE input required

CAUSE: You have not provided an Input Latch Enable (**Ile**) input to a **LINP**, or **COLF**, or **ROLF** primitive. This input is required.

ACTION: Specify an **Ile** input for the primitive.

## ERR-XLT- ILE must be driven by INP or NOTs

CAUSE: The polarity of a latch clock in the EP1210 is determined by inverting the **INP** with a **NOT** primitive before feeding the signal to the latch clock input. You tried to connect something other than a **NOT** primitive to the latch clock input of a **COLF**, **ROLF**, or **LINP** primitive.

ACTION: Ensure that the only logic between a clock pin and the Input Latch Enable (**Ile**) of a **COLF**, **ROLF**, or **LINP** is a **NOT** primitive.

## ERR-XLT- ILEs must all be connected to the same pin

CAUSE: You have tried to use more than one Input Latch Enable (**Ile**) in a circuit design.

ACTION: Connect all **Ile**'s (i.e., in **COLFs**, **ROLFs**, and **LINPs**) to a single pin.

## ERR-XLT- ILEs must all have the same polarity

CAUSE: Some latch clock signals are inverted, others are not. This clock configuration is not supported in Altera EPLDs.

ACTION: Ensure that all **Ile**'s have the same polarity.

## ERR-XLT- Illegal clock pin number

CAUSE: An illegal pin number was requested for the clock pin (i.e., the pin assigned to the clock input is not a clock pin).

ACTION: Check pin assignments for the clock. Refer to *Appendix A* for legal clocking configurations.

## ERR-XLT- Illegal dual I/O feedback for asynchronous register

CAUSE: You have used both asynchronous clocking on a register and dual I/O feedback with **LINP8** or **RINP8** on the same pin. BUSTER EPLDs force **Oe** to **GND** if **LINP8** or **RINP8** is used on the same pin as an asynchronous register. Since asynchronous clocking implies **Oe** = **VCC**, it is not allowed on output registers.

ACTION: Use a synchronous clock for the register.

**ERR-XLT- Illegal pin name character**

CAUSE: The pin name contains a character which is not permitted by A+PLUS software.

ACTION: Ensure that pin names do not contain any percent symbols (%), commas (,), equal signs (=), at-symbols (@), or left and right parentheses (()).

**ERR-XLT- Illegal pin request for dual I/O feedback**

CAUSE: The specified pin does not support dual I/O feedback. This message may occur if you have mistakenly assigned the same pin name to both an input and an output pin.

ACTION: Use dual I/O feedback only on the global macrocells of the EP1800 (all BUSTER macrocells support it) or check pin assignments.

**ERR-XLT- Illegal pin request for primitive**

CAUSE: The pin number is valid, but the target EPLD does not support the requested primitive on the pin specified.

ACTION: Make sure that the capabilities of the pin match the resources requested for the pin. (Refer to the EPLD's data sheet.)

**ERR-XLT- Illegal pin request for this EPLD**

CAUSE: The requested pin assignment is illegal because it is a dedicated input, a power supply pin, or does not exist on the target device.

ACTION: Check pin assignments. You may have assigned a dedicated output to a dedicated input, VCC, or GND pin. Also, check the ADF's Part Section or the Part field of the title block—you may be using the part number suffix for a different package type (e.g. J instead of D, G instead of J).

**ERR-XLT- Illegal use of pin name**

CAUSE: You used a pin name as a node name, but it contains an illegal character. (Only alphanumeric characters are allowed in node names.)

ACTION: Ensure that node names contain only alphanumeric characters. (Refer also to *Network Section Requirements* in *Boolean Equation Entry* in the ***A+PLUS User Guide***.)

**ERR-FLT- Implementation limit <str>**

CAUSE: The design is too large or too complicated.
ACTION: Please contact Altera Applications and describe the number/text shown in the message.

**ERR-ASM- Incompatible or obsolete JDF <filename>**

CAUSE: When the Assembler opened the JEDEC Description File, it found that the file was corrupted or from an earlier version of A+PLUS.
ACTION: (Re)Install the JDFs (provided on the distribution diskettes) in the **APLUS** directory on your computer.

**ERR-FLT- Input line truncated: <str>**

CAUSE: Your input design file contains a line that is more than 512 characters long.
ACTION: Break the line into smaller lines. (For schematic capture design files, reduce the length of pin and node names.)

**ERR-XLT- Input name required for I/O primitive**

CAUSE: You have not provided the name of the primary input node, which is required for all I/O primitives.
ACTION: Connect the main input to all I/O primitives.

**ERR-XLT- Input primitive must have only one output**

CAUSE: You have entered more than one node name as the output of an input primitive.
ACTION: Ensure that each input primitive has only one output.

**ERR-XLT- Internal Bus input is required**

CAUSE: You have not provided an internal bus input, which is required for the **BUSX**, **LBUSI**, and **RBUSI** primitives.
ACTION: Assign an internal bus input to the primitive.

**ERR-ANA-**
**ERR-ASM-**
**ERR-EXP-**
**ERR-FIT-**
**ERR-FLT-**
**ERR-LIB-**
**ERR-MIN-**
**ERR-XLT-** **Internal error: <str>**

CAUSE: There is an internal error.
ACTION: Should you encounter this message, please contact Altera Applications and describe the error text/number.

**ERR-XLT-** **Invalid CLKB statement**

CAUSE: There is a syntax error in the **CLKB** statement.
ACTION: Check the ADF for errors and resubmit the file to the ADP. Each **CLKB** must have only one input and one output.

**ERR-XLT-** **Invalid operator**

CAUSE: A character other than **!**, /, ', +, **#**, *****, **&**, or ; was encountered in an equation.
ACTION: Ensure that all equations use only the legal operators listed above and are terminated with a semicolon (;).

**ERR-XLT-** **JK or SR inputs required for I/O primitive**

CAUSE: You have not provided both primary inputs to a JK or SR register. I/O primitives have two primary inputs, both of which are required.
ACTION: Assign inputs to all J and K (or S and R) registers.

**ERR-XLT-** **LBUSO inputs must be primitive feedbacks or INPs**

CAUSE: You have specified an **LBUSO** input (i.e., one or more of **D0** through **D7**) that is not a primitive feedback or the output of an **INP** primitive.
ACTION: Ensure that all **LBUSO** inputs are primitive feedbacks or the outputs of **INP** primitives.

**ERR-FLT-** **Library corrupted: <str>**

CAUSE: The specified MacroFunction library is corrupted and the Flattener is unable to use it.

ACTION: If the library specified is **MACRO.LIB**, you must reinstall it from the distribution diskette. If the library specified is one you have created with **ADLIB**, try recreating the specified MacroFunction and/or putting it in a different library.

**ERR-FLT-**
**ERR-XLT- Missing <str> Section**

CAUSE: The required Part, Inputs, Outputs, or Network Section is either missing from the ADF or not in the proper sequence.

ACTION: Ensure that your design file has all four of these sections, that they appear in the order given above, and that the **PART:**, **INPUTS:**, **OUTPUTS:**, and **NETWORK:** keywords are the first text in the line in which they appear (i.e., do not use white space before these keywords).

**ERR-XLT- Name too long**

CAUSE: You have entered a pin or node name that contains more than 8 characters, which is the maximum allowed.

ACTION: Shorten the name.

**ERR-XLT- No feedback required for primitive**

CAUSE: Both an output and feedback name were provided for a **CONF**, **JONF**, **RONF**, **SONF**, or **TONF** primitive. These primitives do not have a feedback line associated with them.

ACTION: Remove the feedback node (i.e., use only one output node name on the left-hand side of the primitive statement).

**ERR-XLT- Node missing source**

CAUSE: The Translator found a node name that is not associated with an input primitive statement, I/O primitive statement, or equation.

ACTION: Check the design file and give the node an origin in a primitive.

**ERR-FIT-**
**ERR-XLT- Obsolete PDF**

CAUSE: The Fitter has opened the Part Description File (PDF) used by an earlier version of A+PLUS. This file is no longer valid.

ACTION: Install the A+PLUS version 5.0 PDFs in the **APLUS** directory.

**ERR-XLT- Oe must be VCC in order to promote**

| | |
|---|---|
| CAUSE: | You have used a primitive that is not available for this EPLD. The Translator can usually promote this primitive to one that is appropriate for this EPLD, but is unable to do so because the Output Enable (**Oe**) input is not connected to **VCC**. |
| ACTION: | Use a primitive that is permitted for this EPLD or connect **Oe** to **VCC**. |

**ERR-XLT- Oe not allowed with asynchronous clock**

| | |
|---|---|
| CAUSE: | The Output Enable (**Oe**) and asynchronous clock inputs use the same product term in this EPLD. Therefore, they are mutually exclusive. |
| ACTION: | If you use an asynchronous clock, you must set **Oe** = **VCC**. Use synchronous clocking if you require **Oe** control. |

**ERR-XLT- Oe not allowed with JOIF**

| | |
|---|---|
| CAUSE: | You have used a **JOIF** primitive in your design and have not set **Oe** = **VCC**. This configuration is not allowed in Altera EPLDs. |
| ACTION: | If you use a **JOIF** primitive, you must set **Oe** = **VCC**. Otherwise, you must use a **JOJF** primitive. (Although **JOIF** is not explicitly supported by A+PLUS software, it is accepted if **Oe** is set to **VCC** because it may then be promoted to a **JOJF**.) |

**ERR-ANA-**
**ERR-ASM-**
**ERR-EXP-**
**ERR-FIT-**
**ERR-FLT-**
**ERR-MIN-**
**ERR-XLT- Out of memory**

| | |
|---|---|
| CAUSE: | Available memory is insufficient to complete the current processing step. Other resident programs may be occupying memory, such as RAM-disks, print spoolers, communication packages, and keyboard enhancers. This message may also occur if you have used the **DOS Command** (**<F8>**) function and then *re*-invoked A+PLUS from the temporary DOS environment (a duplicate copy is read into memory if you reinvoke A+PLUS). |

ACTION: Use the DOS command **CHKDSK** to check available memory and disk space. Temporarily relocate other programs to make enough memory available to run A+PLUS and do not use background processes (e.g., Sidekick) when running A+PLUS. If you have used **<F8>**, quit A+PLUS and type **EXIT** to return to the "original" A+PLUS. If you are processing a very large input file, you may wish to run the ADP directly from DOS, which will increase the available memory by approximately 70K (for details, see *A+PLUS and ADP Reference*).

**ERR-FLT-**
**ERR-MIN- Output Error**

CAUSE: This message indicates one of the following problems: insufficient disk space is available, the disk is write protected, or that a disk hardware I/O error occurred.

ACTION: Ensure that 128K of disk space is available, that the disk is not write-protected, and/or check for disk hardware problems.

**ERR-FLT- Output line truncated: <str>**

CAUSE: The Flattener has created an output string over 512 characters long, so it has been truncated. This indicates that after the Flattener has performed symbol substitution, the output line is too long.

ACTION: Use fewer symbols or shorten the symbol names.

**ERR-XLT- Pin missing primitive**

CAUSE: A pin name declared in the Inputs or Outputs Section of the ADF has not been assigned to a primitive in the Network Section.

ACTION: Ensure that all pin names declared in the Inputs and Outputs Sections are also used in the Network Section of the ADF.

**ERR-XLT- Primitive not available for this EPLD**

CAUSE: You have used a primitive that is not supported by the target EPLD.

ACTION: Refer to the foldout in *Appendix F* that shows which primitives are available for each EPLD.

**ERR-XLT- Read Enable input is required**

CAUSE: You have not provided a Read Enable (Re) input, which is required for the **LBUSO** primitive.
ACTION: Assign an **Re** input to the primitive.

**ERR-XLT- Register Clk must be driven by INP or NOTs**

CAUSE: The polarity of a register clock (**Clk**) in the EP1210 is determined by inverting the **INP** with a **NOT** primitive before feeding the signal to the register clock input. You tried to connect something other than a **NOT** to the register clock input of a primitive.
ACTION: Ensure that the only logic between a clock pin and the register **Clk** input is a **NOT** primitive. (Refer to *Appendix A* for legal clock configurations.)

**ERR-XLT- Register Clks must all be connected to the same pin**

CAUSE: You have tried to use more than one register clock input pin in an EP1210 design.
ACTION: Connect all clock inputs to a single pin. (Refer to *Appendix A* for legal clock configurations.)

**ERR-XLT- Register Clks must all have the same polarity**

CAUSE: Some synchronous register clock signals are inverted, others are not. This clock configuration is not supported in Altera EPLDs.
ACTION: Ensure that all clock signals have the same polarity.

**ERR-EXP- Signal name is <feedback node name>**
**ERR-EXP- Illegal combinatorial feedback detected**

CAUSE: While expanding an equation, the Expander found combinatorial feedback that does not use an I/O primitive.
ACTION: Use an I/O primitive (e.g., **COCF, NOCF, ROCF, COIF, ROIF, TOIF**) in your schematic design or in the Network Section of the ADF to obtain combinatorial feedback.

**ERR-XLT- Specify option value as ON or OFF**

CAUSE: The **OPTIONS:** keyword in the ADF is followed by **TURBO** and/or **SECURITY**, but you have not specified **ON** or **OFF**. (All other option values are invalid.)
ACTION: Specify either **ON** or **OFF** for each option.

**ERR-XLT- Synchronous clock cannot drive logic**

CAUSE: You have used a synchronous clock signal as an input to the logic array. Since it doesn't go through the array, the clock signal can only be connected to a flipflop clock. (A signal that feeds a synchronous clock cannot feed logic or asynchronous clocks.)

ACTION: Remove any clock-to-logic connections.

**ERR-XLT- Too many '( )'s**

CAUSE: The equation exceeds the maximum nesting level permitted for parentheses in an ADF.

ACTION: Reduce the number of parentheses.

**ERR-XLT- Too many '('s**

CAUSE: The equation has too many left parentheses.

ACTION: Ensure that an equal number of left and right parentheses are present in each equation in the Equations Section.

**ERR-XLT- Too many ')'s**

CAUSE: The equation has too many right parentheses.

ACTION: Ensure that an equal number of left and right parentheses are present in each equation in the Equations Section.

**ERR-XLT- Too many <name(s)> primitives**

CAUSE: You have used too many of the specified Bus primitives in the circuit design for a BUSTER EPLD.

ACTION: In a single BUSTER EPLD, use at most 1 **BUSX** primitive; 2 **LBUSO** primitives; and 2 primitives selected from the following group: **LBUSI**, **LINP8**, **RBUSI** and **RINP8**.

**ERR-XLT- Too many bits on a Bus primitive**

CAUSE: You have specified more than 8 bits on a Bus primitive, which is the maximum allowed.

ACTION: Reduce the number of bits to exactly 8.

**ERR-XLT- Too many CLKBs**

CAUSE: Available memory is insufficient to handle all the CLKB primitives in your design. Since at most one CLKB is required for each asynchronous clock resource, this message probably indicates that your design will not fit into any existing EPLD or that you have used CLKB incorrectly.

ACTION: Use fewer CLKBs in your design. (Refer to *Altera Primitive Library* and *Appendix A* for descriptions of CLKB and legal clocking configurations.)

**ERR-XLT- Too many clock pins for this EPLD**

CAUSE: You requested more register/latch clock inputs than the target EPLD can support.

ACTION: Refer to *Appendix A* for information on legal clocking configurations. You may need to use a different EPLD.

**ERR-XLT- Too many dual I/O feedback macrocells for this EPLD**

CAUSE: You requested more dual I/O feedback macrocells than the target EPLD can support.

ACTION: Use only the global macrocells on the EP1800 for dual I/O feedback. (All BUSTER macrocells support it.) Refer also to the data sheet for the target EPLD.

**ERR-XLT- Too many input pins for any EPLD**

CAUSE: You have requested more input pins in your design than are available on any existing Altera EPLD.

ACTION: Break the design into two smaller functional parts and fit them separately.

**ERR-XLT- Too many internal buses**

CAUSE: You have used two different names for the same internal bus or have not connected all internal buses used in the design.

ACTION: Use the same name for all internal buses. Ensure that all internal buses are connected together.

**ERR-XLT- Too many macrocells for any EPLD**

CAUSE: You have requested more macrocells in your design than are available on any existing Altera EPLD.

ACTION: Break the design into two smaller functional parts and fit them separately.

**ERR-XLT- Too many macrocells for this EPLD**

CAUSE: You have requested more macrocells than are available in the target EPLD.

ACTION: Modify your design or use a different EPLD.

**ERR-XLT- Too many NOTs**

CAUSE: You have requested more NOT gates in your design than are available on any existing Altera EPLD.

ACTION: Try redesigning the logic to reduce the number of NOT gates.

**ERR-XLT- Too many pins for this EPLD**

CAUSE: You have requested more clock, input, or output pins than the target EPLD can support.

ACTION: Modify your design or use a different EPLD.

**ERR-EXP- Too many symbols in equations**

CAUSE: Available memory is insufficient for internal representation of equations. It is possible that the equations are already in expanded form or that other resident programs are occupying memory.

ACTION: Use intermediate equations to remove common subexpressions from the equations. Example:

```
A   =   B + C + D + (E * F)
G   =   B + C + D + (H * I)

J   =   B + C + D
A   =   J + (E * F)
G   =   J + (H * I)
```

Other resident programs that use memory may include RAM-disks, print spoolers, communication packages, and keyboard enhancers.)

**ERR-EXP- Too many unique symbols**

CAUSE: Available memory is insufficient for storing symbols. It is possible that there are too many parentheses, because the Expander generates a unique symbol for each pair of parentheses. Other resident programs may be occupying memory, such as RAM-disks, print spoolers, communication packages, and keyboard enhancers.

ACTION: Reduce the number of parentheses by removing unneeded parentheses and using intermediate equations to remove common subexpressions from the equations. Ensure that enough memory is available for running A+PLUS by temporarily relocating other programs that are resident in memory.

### ERR-XLT- Too many/few inputs to logic primitive

CAUSE: You have entered too many/few inputs for a logic primitive or have left some inputs unconnected.

ACTION: If you are using a schematic capture package, ensure that **AND**, **NAND**, **NOR**, and **OR** gates have 2, 4, 6, 8, or 12 inputs. **NOT** gates must have exactly 1 input; **XOR** gates must have exactly 2 inputs. Connect the unused inputs to other inputs that *are* used in the same logic gate. If you are using Boolean equation entry, you may use any number between 2 and 12 inputs for **AND**, **NAND**, **NOR**, and **OR** primitives.

### ERR-XLT- Too many/few inputs to primitive

CAUSE: You have entered too many/few of the input parameters required on the right-hand side of a primitive. This message may occur if you intended to use some default values but have too many/few of the commas that are required to delimit the fields.

ACTION: Check the Network Section for errors and resubmit the file to the ADP.

### ERR-XLT- Too many/few signals on left-hand side

CAUSE: You have entered more signal names on the left-hand side of a primitive statement than the primitive supports.

ACTION: Add/remove signal names and resubmit the file to the ADP. (Refer to *Altera Primitive Library* for information on primitive syntax.

### ERR-XLT- Undeclared input pin name

CAUSE: The input node associated with an Input primitive was not declared in the Inputs Section of the ADF. (You probably forgot to name the pin.)

ACTION: Declare all input pin names in the Inputs Section of the ADF.

**ERR-XLT- Undeclared output pin name**

CAUSE: The output node associated with an I/O primitive was not declared in the Outputs Section of the ADF. (You probably forgot to name the pin.)

ACTION: Declare all output pin names in the Outputs Section of the ADF.

**ERR-XLT- Unexpected end of ADF**

CAUSE: The Translator encountered an End-of-File prematurely, i.e., **END$** was not found.

ACTION: Ensure that the **END$** statement is at the end of the file, and that there are no comments, tabs, or blank spaces before it.

**ERR-XLT- Unexpected keyword**

CAUSE: The Translator encountered a keyword for another section before the previous section was terminated.

ACTION: Check the last line of the previous section for syntax errors.

**ERR-XLT- Unrecognized character in header**

CAUSE: The Translator encountered a non-printing character or asterisk (*) in the header section of the Altera Design File.

ACTION: Remove asterisks and non-printing characters from the ADF's Header Section.

**ERR-XLT- Unrecognized EPLD**

CAUSE: You have specified an EPLD name that is not recognized as an Altera EPLD.

ACTION: Check the Part Section in the ADF or the EPLD field in your schematic design. Refer to *Altera Design File Format* for a list of all legal EPLD names.

**ERR-XLT- Unrecognized primitive name**

CAUSE: The primitive name is not that of an Altera primitive. This message may occur if non-recognizable characters, i.e., lowercase letters, are used in a primitive name.

ACTION: Ensure that all primitive names are given in capital letters. Refer also to *Altera Primitive Library* for a list of all available primitives.

**ERR-APLUS- While using the DOS Command <F8> function, you ran a program that is still occupying memory. The only way to recover this memory is to *reboot* DOS. A+PLUS is quitting -- Sorry!**

CAUSE: You have run a DOS program with the **<F8>** function that is continuing to occupy memory that is needed to run A+PLUS. DOS commands such as **print** and **graphics** will cause this message, as well as programs like Sidekick.

ACTION: Type **<Ctrl><Alt><Del>** to reboot DOS and type **APLUS** to reload A+PLUS.

# Information Messages

During normal processing, the Altera Design Processor displays information messages indicating that a particular ADP module has completed execution without error. These messages begin with the prefix *****INFO-ADP-** and require no corrective action.

Other information messages describe the actions performed by individual ADP modules, and rarely require corrective action. However, many Fitter module information messages describe where design modifications are required to implement the design on the target EPLD.

**INFO-FIT- A fit may be possible with the proper pin assignments, but the Fitter was unable to determine what those assignments should be**

CAUSE: Your design is too complex to be automatically fitted by the ADP's Fitter module.

ACTION: Try to fit the design manually. Contact Altera Applications for assistance.

**INFO-FIT- A+PLUS is unable to fit this design**

CAUSE: Based on the current input file(s), A+PLUS is unable to fit the design.

ACTION: See the explanation of the other message(s) that accompanies this message. (Refer to the Utilization Report file for additional information.)

**INFO-ADP- ADF converted to LEF**

CAUSE: The Altera Design File has been successfully converted into a file in which the Boolean and non-Boolean elements of the design have been identified and separated. The file containing these elements is called a Logic Equation File (LEF).

ACTION: No action is required.

**INFO-XLT Automatic part selection:**

•

•

•

**Selected part: <part name>**

CAUSE: Your input design file specified automatic part selection, so the ADP's Translator module will choose the part most likely to fit your design. This message includes several lines (indicated with "...") documenting the resources the Translator has considered when performing automatic part selection. These resources may include the numbers of inputs, macrocells, macrocells used as buried macrocells, synchronous and asynchronous clocks, Bus I/O primitives, the types of flipflops needed, etc. The last line of the message indicates the part selected by the Translator.

ACTION: No action is required.

**INFO-ADP- Beginning design processing**

CAUSE: The Altera Design Processor (ADP) has begun compiling your design file.

ACTION: No action is required.

**INFO-FIT- Can't promote <name> to a global I/O resource (Oe = <node name>)**

CAUSE: In the EP1800, a global signal (<name>) that feeds other feedback groups must use I/O feedback. The current macrocell cannot be promoted to an I/O resource because it uses Oe control.

ACTION: Try changing the primitive to one that uses I/O feedback (e.g. COIF, ROIF, TOIF).

**INFO-FIT- Can't put both <name> and <name> on pin**

CAUSE: The Fitter was forced to assign conflicting resources to the same pin. (In the 1800, this message may occur if the Fitter was forced to promote a buried register to an I/O resource.)

ACTION: Assign the signals to different pins. (Refer to the Utilization Report for information.) If you need assistance, contact Altera Applications.

## INFO-FIT- Clear/Preset inverted

CAUSE: De Morgan's inversion caused the asynchronous Clear to become an asynchronous Preset and the synchronous Preset to become a synchronous Clear. This message may appear for any macrocell in the EP310 and buried macrocells in the EP1210.

ACTION: If this condition is undesirable, you may (re)run your design through the Altera Design Processor (ADP) and request control over the inversion of each equation (enter **N** at the **<F6> Inversion Control** prompt). When the indicated equation appears, you should request **no inversion** (or **re-inversion**) to ensure that the resulting equation is not inverted.

## INFO-FIT- Clocks need to be tied to additional pins

CAUSE: A synchronous clock input feeds more macrocells than are available in a single clock group, so you must wire multiple clock pins together to implement the design. The extra clock pins required will reduce the number of dedicated input pins available to the remainder of the design.

ACTION: Use asynchronous clocking (feed the clock input through a **CLKB** primitive).

## INFO-FIT- Conflicting clock groups

CAUSE: Two signals with different clocks have been assigned to the same clock group. These signals were either assigned explicitly with pin requests, or a local signal in one clock group was feeding an unassigned signal with a conflicting clock.

ACTION: Use different pin assignments or allow the Fitter to make pin assignments.

## INFO-FIT- Dedicated input used as output: <node name>

CAUSE: You have assigned a macrocell to a dedicated input pin.

ACTION: Assign macrocells only to I/O pins.

## INFO-ADP- Design fitting complete

CAUSE: The Fitter has matched the data produced in the previous step with the resources available in the target EPLD. The resultant information has been tabulated and readied for conversion into JEDEC file format, and a Utilization Report file (with the extension **.RPT**) has been written to disk.

ACTION: No action is required.

### INFO-ADP- Design fitting complete

CAUSE: The Fitter has matched the data produced in the previous step with the resources available in the target EPLD. The resultant information has been tabulated and readied for conversion into JEDEC file format, and a Utilization Report file (with the extension **.RPT**) has been written to disk.

ACTION: No action is required.

### INFO-MIN- Equation with LHS: <str> can't be inverted

CAUSE: The equation with left-hand side (LHS) <str> will not fit into the Minimizer's internal tables in its inverted form. The equation is output in the form it had before the failed inversion attempt. Note that the Minimizer may have received the equation with the LHS already complemented, i.e., inversion may have occurred during expansion.

ACTION: No action is required.

### INFO-MIN- Equation with LHS: <str> may not be fully minimized

CAUSE: An internal table was filled to capacity during the reduction process. The equation whose left-hand side (LHS) is represented by <str> has been retained in its correct but only partially minimized form.

ACTION: Try redesigning the circuit to use fewer levels of logic.

### INFO-FIT- Externally connect clock signal <name> to pins

CAUSE: A synchronous clock feeds multiple clock groups.

ACTION: You must physically wire the specified clock pins together to implement the design.

### INFO-FIT- Illegal inversion of <str> input

CAUSE: An equation feeding a secondary input, (e.g., **Oe**, **Clk**, **Clr**, or **P**) has been minimized with De Morgan's inversion. The secondary inputs on this register may not be inverted. This message indicates that you have too many product terms on **Oe**, **Clk**, **Clr**, or **P** or an I/O primitive and that A+PLUS has performed a De Morgan's inversion on the logic to reduce the number of product terms.

ACTION: Change the logic or insert an **NOCF**.

**INFO-FIT- Illegal pin request for this EPLD**

CAUSE: The requested pin assignment is illegal because the pin is dedicated to another function (e.g., a clock, latch clock, strobe, power supply, etc.) or does not exist on the target device.

ACTION: Check pin assignments. (For example, you may have assigned a dedicated output to a dedicated input, **VCC**, or **GND** pin.) Also, check the ADF's Part Section or the Part field of the title block—you may be using the part number suffix for a different package type (e.g. J instead of D, G instead of J).

**INFO-FIT- Input latches cannot control each other**

CAUSE: You have created a BUSTER design where the input latches feed each other. These input latches use local feedback and reside in different feedback groups; therefore, this condition is illegal.

ACTION: Redesign the circuit to ensure that the inputs to latches in BUSTER do not feed each other. You may need to use a different EPLD.

**INFO-FIT- Input latches cannot feed the same primitive**

CAUSE: You have used the output of two input latches to feed another primitive. The latches in BUSTER use local feedback and reside in different feedback groups; therefore this condition is illegal.

ACTION: Redesign the circuit to ensure that latches in BUSTER do not feed the same primitives. You may need to use a different EPLD.

**INFO-MIN- Inversion/re-minimization specified**
**( current pterm count: # )**
**( current left-hand side: <str> )**

CAUSE: After you have requested equation-by-equation control of De Morgan's inversion, the Minimizer informs you of the current product term count and the name of the signal on the left-hand side of the equation.

ACTION: No action is required.

## INFO-ADP- JEDEC file output

CAUSE: The tabulated information produced by the Fitter has been converted to a JEDEC file and has been written to the default drive as **<filename>.JED**. If more than one input file was specified, the JEDEC **<filename>** is the same as the first filename you entered. This message indicates that design processing has been completed.

ACTION: No action is required.

## INFO-ADP- LEF analyzed

CAUSE: The LEF Analyzer has converted the file output by the Expander (or Minimizer) into readable form and processing control has returned to the ADP. The Logic Equation File produced (with the extension **.LEF**) will appear in the same directory as the input design file.

ACTION: No action is required. You may examine the LEF after design processing is completed.

## INFO-ADP- LEF reduced

CAUSE: The Minimizer has reduced the sum-of-products portion of the LEF according to the theorems of Boolean algebra and I/O architecture requirements have been optimized.

ACTION: No action is required.

## INFO-FIT- Local signal <name> on pin # (Feedback group #), Local signal <name> on pin # (Feedback group #): Both can't feed <name>

CAUSE: Two signals assigned to local feedback pins are in different feedback groups yet feed the same signal. This is an impossible condition to fit.

ACTION: Change your pin assignments or allow the Fitter to select pin assignments.

## INFO-FIT- Local signal <name> on pin # (Feedback group #): Can't reach <name> on pin # (Feedback group #)

CAUSE: A signal assigned to a local feedback pin feeds a signal in a different feedback group. No fit is possible.

ACTION: Change your pin assignments or allow the Fitter to select pin assignments.

## INFO-ADP- MacroFunction ready for ADLIB

CAUSE: The LEF Analyzer has converted your new MacroFunction design file into a Logic Equation File (with the extension .LEF).

ACTION: Refer to the ***ADLIB*** manual for instructions on how to use the new MacroFunction.

## INFO-FIT- No fit possible

CAUSE: The Fitter has determined that your design won't fit on the target EPLD.

ACTION: Check the Utilization Report (**.RPT**) file to determine why the design could not fit. Try a different EPLD. If your target EPLD was an EP1800, try changing synchronous clocks into asynchronous clocks by inserting **CLKB**s.

## INFO-FIT- No fit possible with current pin assignments

CAUSE: The Fitter cannot fit your design with the pre-assigned pin requests.

ACTION: When the Fitter asks you whether pin assignments may be ignored, answer **Y** to allow it to fit the design with new pin assignments. If you receive this message after typing **N** in response to the prompt "**OK to ignore pin assignments?**", you must resubmit the input file to the ADP (remove pin assignments from the file, or type **Y** the next time the prompt appears).

## INFO-FIT- Output latch's macrocells cannot be fed by more than one input latch

CAUSE: All macrocells that feed an output latch in BUSTER must reside in the same feedback group. Both of the input latches in your design feed one or more of these macrocells. Since the input latches in BUSTER can use only local feedback and reside in different feedback groups, this is an impossible condition to fit.

ACTION: Ensure that the input latches do not feed the same macrocell that belongs to an output latch. Redesign the circuit or use a different EPLD.

**INFO-FIT- Output latches cannot be controlled by more than one input latch**

CAUSE: Both of the input latches in your design feed the same output latch. Since the input latches in BUSTER can use only local feedback and reside in different feedback groups, this is an impossible condition to fit.

ACTION: Ensure that the input latches do not feed the same output latch. Redesign the circuit or use a different EPLD.

**INFO-FIT- Output latches must be directly fed by macrocells or dedicated inputs**

CAUSE: You have either omitted the input to an output latch or used combinatorial logic as an input to an output latch.

ACTION: Provide inputs for the output latch and/or insert an **NOCF** between the combinatorial logic and the macrocell.

**INFO-XLT- Part: <part name>**

CAUSE: The Translator is reminding you of the part name you specified in your input design file.

ACTION: No action is required.

**INFO-FIT- Permission to remove pin requests denied**

CAUSE: The Fitter is reminding you that you typed **N** in response to the prompt "**OK to ignore pin assignments?**" This message will probably be followed by a message stating that no fit is possible with current pin assignments.

ACTION: Resubmit your design file to the ADP. When the Fitter asks you whether pin assignments may be ignored, answer **Y** to allow it to fit the design with new pin assignments.

**INFO-FIT- Pin assignments differ from those specified in ADF**

CAUSE: When you responded **Y** to the prompt "**OK to ignore pin assignments?**", the Fitter made new pin assignments in order to fit the design onto the target EPLD.

ACTION: We recommend that you transfer the Fitter's pin assignments to your input design file.

## INFO-XLT- Promoted <str1>(s) to <str2>(s)

CAUSE: The EPLD selected does not support the I/O primitive specified in <str1>. The Translator has promoted it to the primitive specified in <str2> to accommodate the part selection.

ACTION: No action is required.

## INFO-FLT- Reading MacroFunction library <name>

CAUSE: The Flattener is reading the contents of a library that was specified in the **set ADLIB** string, i.e., a library that contains a MacroFunction used in your input design file.

ACTION: No action is required.

## INFO-ADP- S.O.P. LEF produced

CAUSE: The Boolean elements of the design have been expanded into sum-of-products (S.O.P.) form, and passed to the next ADP module as an intermediate Logic Equation File (LEF).

ACTION: No action is required.

## INFO-MIN- "Stuck at 1" equation (LHS: <str>) inverted to "Stuck at 0"

CAUSE: An equation input to the Minimizer was reduced to the "Stuck at 1" condition (e.g., **S = A + A' ...**). De Morgan's inversion was then performed and the Minimizer generated an equation of the form **S' = A' * A;**.

ACTION: If this was not your intention, examine your design and correct logic that reduces to this form.

## INFO-FIT- Stuck on pre-assigned pin requests

CAUSE: The Fitter is unable to fit the design with the current pin assignments.

ACTION: Use different pin assignments, allow the Fitter to make pin assignments, or provide more pin assignments in your input design file.

**INFO-FIT- Too many <resource> groups for this EPLD**

CAUSE: Your designed requires more synchronous clocks, Clear, Preset, or Output Enable than the target EPLD. This message indicates that there are more unique equations than groups in the device.

ACTION: If the specified **<resource>** is a synchronous clock, redesign the circuit to use asynchronous clocking (if possible). You may need to use a different EPLD.

**INFO-FIT- Too many clock pins for this EPLD**

CAUSE: Your design requires more synchronous clock inputs than the target EPLD can support.

ACTION: Check whether you have made any clock pins unavailable by using them as dedicated inputs. Refer to *Appendix A* for information on legal clocking configurations. You may need to use a different EPLD.

**INFO-FIT- Too many I/O pins and buried macrocells for this EPLD**

CAUSE: You requested more I/O pins and buried macrocells than the target EPLD can support. Not all of the "buried" registers in the design could be truly buried. Instead, they were placed on I/O macrocells whose pins became reserved. Consequently, there are not enough I/O pins to fit the remaining input and I/O macrocell requests.

ACTION: You may need to use a different EPLD. Refer to the data sheets for Altera EPLDs for information.

**INFO-FIT- Too many I/O pins for this EPLD**

CAUSE: Your design required more I/O pins than the target EPLD can support. This condition may have resulted from the need to externally connect synchronous clock pins.

ACTION: Try using asynchronous instead of synchronous clocks. You may need to use a different EPLD.

**INFO-FIT- Too many macrocells for this EPLD**

CAUSE: You have requested more macrocells than are available in the target EPLD.

ACTION: Modify your design or use a different EPLD.

**INFO-FIT- Too many pterms for <resource> input**

CAUSE: An equation feeding a secondary input, e.g., Clock (Clk), Clear (C), Preset (P), or Output Enable (Oe), uses more than one product term. (In BUSTER, = 2)

ACTION: Change the logic or insert an NOCF to reduce the number of product terms.

**INFO-FIT- Too many pterms for any macrocell**

CAUSE: An equation uses has more product terms than are available in any macrocell on this EPLD.

ACTION: Change the logic or insert NOCFs to reduce the number of product terms.

**INFO-FIT- Too many pterms for this macrocell**

CAUSE: A signal has too many product terms to fit in the macrocell of the pin to which it is assigned.

ACTION: In the EP1210, using a different macrocell may eliminate the problem. Otherwise, change the logic or insert NOCFs to reduce the number of product terms.

## Warning Messages

Warning messages indicate *potential* problems which do not always require corrective action. Design processing will continue until an error is encountered.

**WARN-ADP- Available disk space (#K) may be inadequate. Please make at least 128K available on the disk containing your input file(s)**

CAUSE: A+PLUS has determined that available disk space may be insufficient for processing your design file.

ACTION: Make at least 128K of disk space available on the disk containing your input file(s).

**WARN-FLT- Can't find MACRO.LIB**

CAUSE: The Flattener could not find MACRO.LIB. (Processing will continue if your design employs only user-defined MacroFunctions.)

ACTION: Check the DOS path. You may need to (re)install MACRO.LIB.

**WARN-XLT- Equation missing left-hand side**

CAUSE: The first character the Translator recognized in this equation was an equals sign.

ACTION: Ensure that all entries in the ADF's Equations Section have one node name on the left-hand side.

**WARN-XLT- Feedback not defined for I/O primitive**

CAUSE: You have not provided a feedback node name as the second parameter on the left-hand side of an I/O primitive.

ACTION: You may wish to use a No-Feedback primitive (CONF, RONF, TONF, or JONF) or specify a feedback for the primitive.

**WARN-FLT- Library <name> is empty**

CAUSE: The specified MacroFunction library, which appears in your **set ADLIB** string, is empty.

ACTION: Check the path and filename entered, and remove empty libraries from the **set ADLIB** string.

**WARN-FLT- Library <name> is not a MacroFunction library**

CAUSE: The name you specified in the **set ADLIB** string is not recognized as a valid MacroFunction library. This may mean that your library has been corrupted, or that you have named a non-A+PLUS library as **MACRO.LIB**.

ACTION: Reinstall the library or check the pathnames in the **set ADLIB** string.

**WARN-FLT- Library <name> is too big**

CAUSE: The specified library contains over 256 MacroFunctions, which is the maximum permitted.

ACTION: Move the overflow to another library. (You may use up to 15 MacroFunction libraries for each design.)

**WARN-APLUS- Memory available (approx. #K) may be insufficient to compile your design. Make at least #K available. Memory may be low if you have reinvoked APLUS while using the <F8> (DOS Command) function. Type EXIT at the DOS prompt to return to the "original" APLUS.**

CAUSE: A+PLUS has determined that there may not be enough memory available to process your design file.

ACTION: Follow the guideline given in the message for making additional memory available (i.e., temporarily remove other programs that are resident in memory).

**WARN-XLT- Node missing destination**

CAUSE: A signal node was found that has no primitive output associated with it, i.e., it was not used in the Equations or Network Section of the ADF. This message is common for many designs, and does not necessarily indicate an error. However, you may have forgotten to give this node a destination (i.e., an equation or primitive).

ACTION: Check whether you have forgotten to give the node a destination.

## WARN-XLT- Primitive missing output

CAUSE: You have not provided an output signal for the primitive. (The first character the Translator recognized in the primitive statement was an equals sign.)

ACTION: To prevent this message from occurring, delete the primitive or assign an output.

## WARN-FLT- Too many libraries: can't include library <name>

CAUSE: You have specified more than 15 MacroFunction libraries in the **set ADLIB** string, which is the maximum allowed. (The specified **<name>** was the 16th library found by the Flattener.)

ACTION: Shorten the **set ADLIB** string to contain only 15 user-defined MacroFunction libraries.

APPENDIX A

# Part-Specific Information

This Appendix provides information on Altera's A+PLUS-programmable EPLDs, including macrocell group tables, macrocell block diagrams, and synchronous and asynchronous clocking diagrams for each EPLD. For additional information on device architecture and timing parameters, refer to the ***Altera Data Book*** and each device's data sheet.

# The Altera EPLD Family of Parts

The Altera family of A+PLUS-programmable parts consists of the following logic devices:

- EP310
- EP320
- EP600, EP610
- EP900, EP910
- EP1210
- EPB1400 ("BUSTER")
- EP1800

Altera EPLDs are made with reprogrammable, erasable logic arrays that use CMOS EPROM bits as the programmable connections between the logic functions. If you wish to program a new design, the array is exposed to ultraviolet light that erases the current logic design and allows you to immediately program a new one.

These devices use sum-of-products architecture that enables you to quickly and easily program complex custom logic functions. In addition, since output architecture can be programmed, each output pin can be assigned either combinatorial or registered output in active high or active low mode.

Altera EPLDs are available in three package configurations: DIP, J-lead, or Pin-Grid array. The suffix D designates a DIP package; the suffix J designates a J-lead package; and the G suffix designates a Grid-array package. In all EPLDs except the EP1800, the lack of a suffix also designates a DIP package; in the EP1800, the lack of a suffix designates a J-lead package. (For example: EP310 or EP310D indicates an EP310 in a DIP package; EP900J indicates an EP900 in a J-lead package; EP1800G indicates an EP1800 in a Grid-array package.)

Pin numbers given in macrocell group tables pertain only to DIP packages. The corresponding pin numbers in J-lead and Grid-array packages are shown in parentheses in the block diagrams and in the text.

## EP310 and EP320

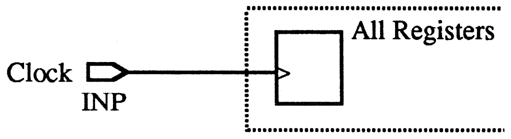
The EP310 and EP320 are 20-pin DIP packages that provide 10 dedicated input and 8 programmable I/O pins. Each I/O pin is associated with a macrocell with 8 product terms and an I/O architecture control block.

Each macrocell can be configured to provide registered or combinatorial output. Feedback from each macrocell can be combinatorial, regardless of how the output is configured. A dedicated clock input, pin 1, feeds the macrocell registers and can also feed the p-term array. The Output Enables for each macrocell are each supported by single p-terms.

The EP310 provides Clear and Preset signals to the macrocells, each of which is supported by a single p-term. The EP320 does not provide these functions and thus has lower propagation delays.

Tables A-1 and A-2 show the pin assignments for each macrocell, as well as the feedback, Clock group, Clear/Preset group, Output Enable group, and I/O group to which each macrocell belongs for the EP310 and EP320, respectively. Figure A-1 shows all legal clock configurations for the EP310 and EP320. Block diagrams of the EP310 and 320 are shown in Figures A-2 and A-3, respectively.

## Pin Clocking



## Pin Clocking with Logic Fanout

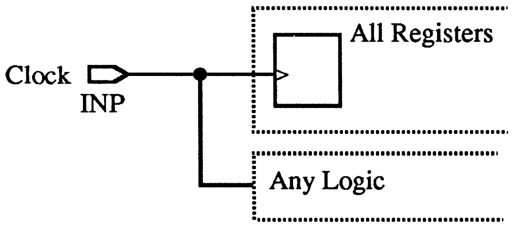


**Figure A-1. EP310/EP320 Clocking**

**Table A-1. EP310 Macrocell Groups**

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 1 | 19 | global | - | 1 | 1 | 1 |
| 2 | 18 | global | - | 1 | 2 | 2 |
| 3 | 17 | global | - | 1 | 3 | 3 |
| 4 | 16 | global | - | 1 | 4 | 4 |
| 5 | 15 | global | - | 1 | 5 | 5 |
| 6 | 14 | global | - | 1 | 6 | 6 |
| 7 | 13 | global | - | 1 | 7 | 7 |
| 8 | 12 | global | - | 1 | 8 | 8 |

**Table A-2. EP320 Macrocell Groups**

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 1 | 19 | global | - | - | 1 | 1 |
| 2 | 18 | global | - | - | 2 | 2 |
| 3 | 17 | global | - | - | 3 | 3 |
| 4 | 16 | global | - | - | 4 | 4 |
| 5 | 15 | global | - | - | 5 | 5 |
| 6 | 14 | global | - | - | 6 | 6 |
| 7 | 13 | global | - | - | 7 | 7 |
| 8 | 12 | global | - | - | 8 | 8 |

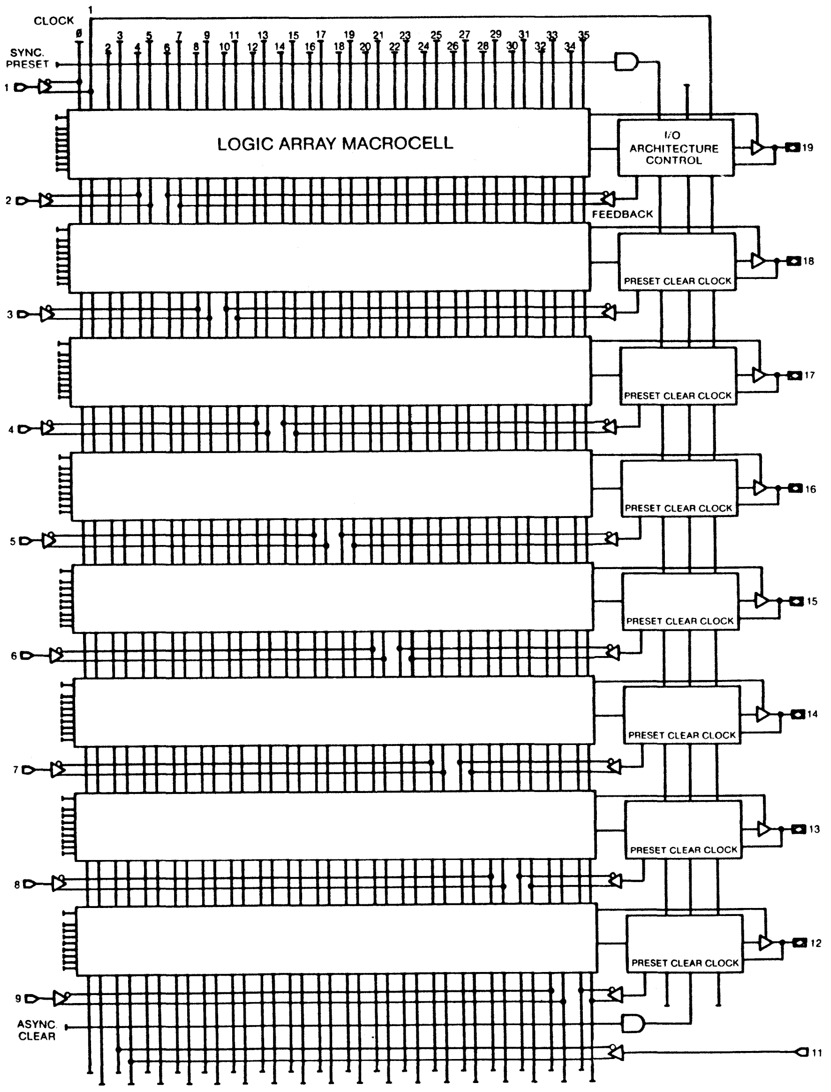


**Figure A-2. EP310 Block Diagram**

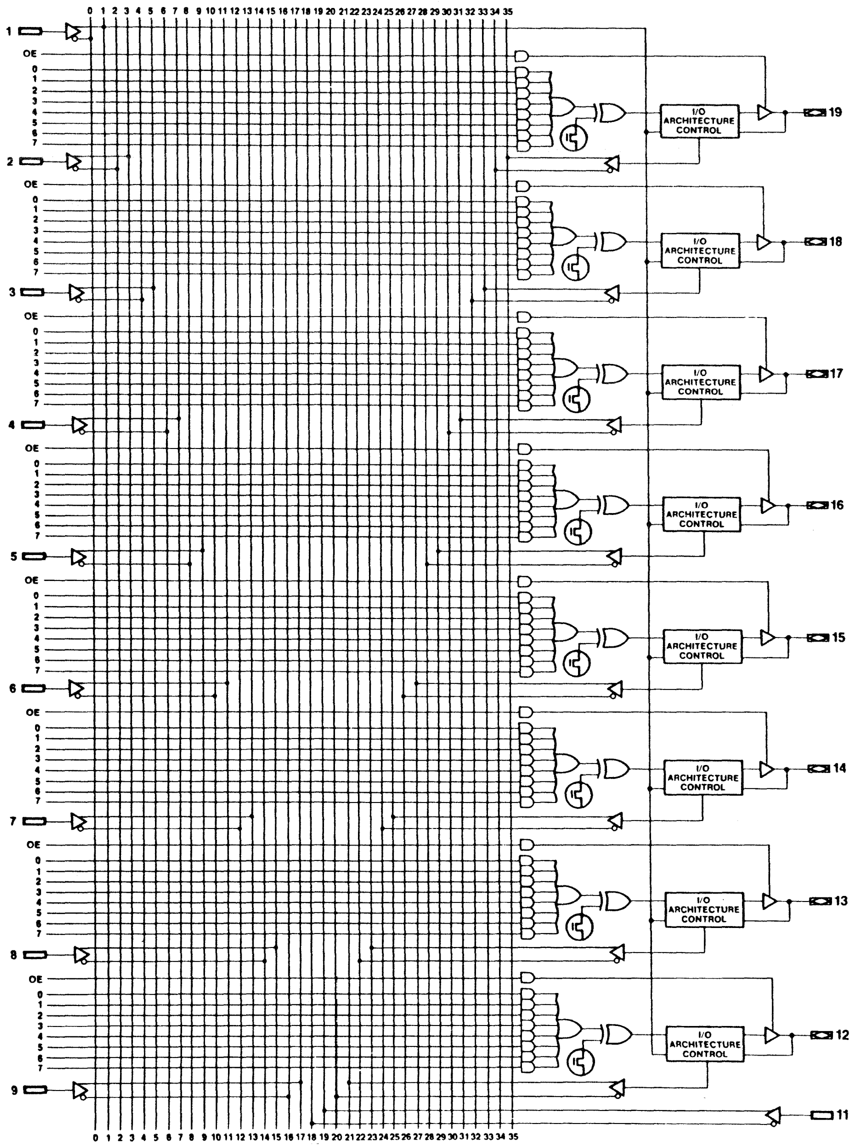


**Figure A-3. EP320 Block Diagram**

## EP600 and EP610

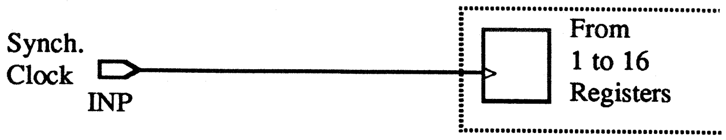
The EP600 and EP610, available in 24-pin DIP or 28-pin J-lead packages, provides 4 dedicated input, 2 dedicated clock input, and 16 programmable input/output pins. Each I/O pin is associated with a macrocell with 10 product terms and an I/O architecture control block. Eight product terms are used for logic requirements, one for Output Enable/Asynchronous Clock implementation, and one for flipflop clear control. The I/O architecture control block allows selection of D, T, JK, or SR flipflops, or combinatorial output. Each macrocell allows programmable selection of a combinatorial output path with I/O feedback or a registered output path with registered or I/O feedback. The two dedicated clock pins provide positive-edge-triggered synchronous clock input to the registers.

Table A-3 shows the pin assignments for each EP600/EP610 macrocell, as well as the feedback, Clock group, Clear/Preset group, Output Enable group, and I/O group to which each macrocell belongs. Figure A-5 shows a block diagram of the EP600/EP610.

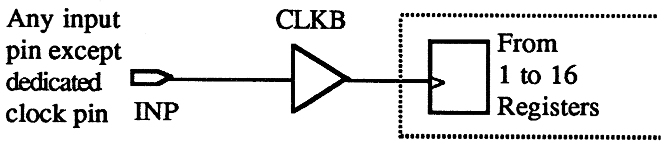
Pins 1 and 13 (2 and 16) are the two synchronous clock inputs. Pin 1 (2) feeds the registers on the left half of the device—pins 3-10 (4-12); pin 13 (16) feeds the registers on the right half—pins 15-22 (18-26). If more than 8 registers are connected to the same synchronous clock input, both clock pins are needed to implement the circuit. In this case, the pin diagram in the Utilization Report contains a reminder to electrically tie the pins together. Figure A-4 shows all legal clock configurations for the EP600/EP610.

Any register can have an asynchronous clock.

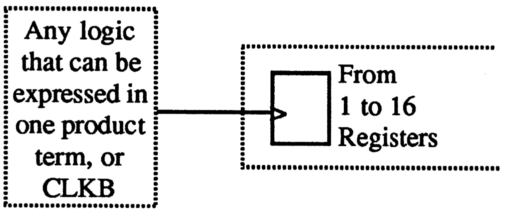
## Pin Clocking—Synchronous



## Pin Clocking—Asynchronous



## Logic or Feedback Clocking



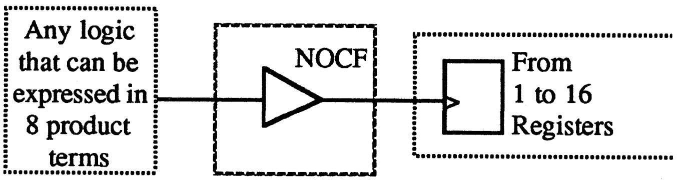


**Figure A-4. EP600/EP610 Clocking**

## Table A-3. EP600/EP610 Macrocell Groups

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 1 | 22 | global | 2 | 1 | 1 | 1 |
| 2 | 21 | global | 2 | 2 | 2 | 2 |
| 3 | 20 | global | 2 | 3 | 3 | 3 |
| 4 | 19 | global | 2 | 4 | 4 | 4 |
| 5 | 18 | global | 2 | 5 | 5 | 5 |
| 6 | 17 | global | 2 | 6 | 6 | 6 |
| 7 | 16 | global | 2 | 7 | 7 | 7 |
| 8 | 15 | global | 2 | 8 | 8 | 8 |
| 9 | 3 | global | 1 | 9 | 9 | 9 |
| 10 | 4 | global | 1 | 10 | 10 | 10 |
| 11 | 5 | global | 1 | 11 | 11 | 11 |
| 12 | 6 | global | 1 | 12 | 12 | 12 |
| 13 | 7 | global | 1 | 13 | 13 | 13 |
| 14 | 8 | global | 1 | 14 | 14 | 14 |
| 15 | 9 | global | 1 | 15 | 15 | 15 |
| 16 | 10 | global | 1 | 16 | 16 | 16 |

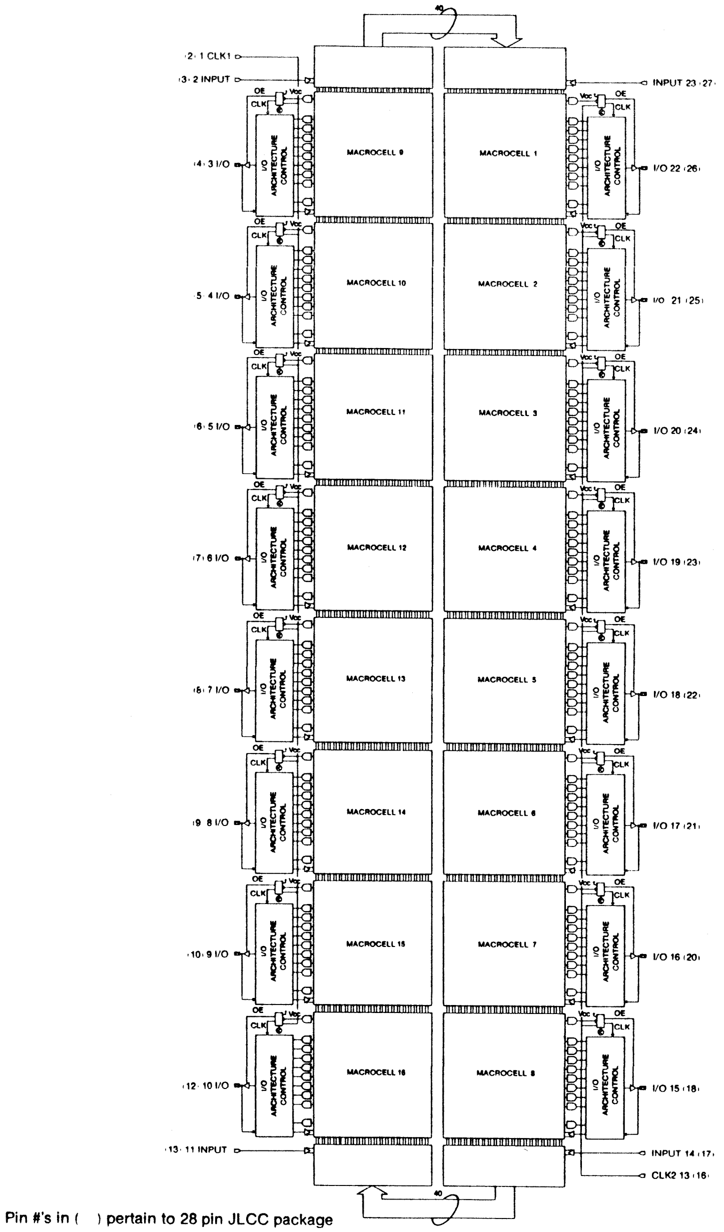


**Figure A-5. EP600/EP610 Block Diagram**

## EP900 and EP910

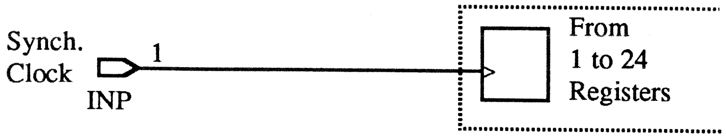
The EP900 and EP910, available in 40-pin DIP or 44-pin J-lead packages, provide 12 input and 24 programmable input/output pins. Each I/O pin is associated with a macrocell that has 10 product terms and an I/O architecture control block. Eight product terms are used for logic requirements, one for Output Enable/Asynchronous Clock implementation, and 1 for flipflop clear control. The I/O architecture control block allows selection of D, T, JK, or SR flipflops, or combinatorial output. Each macrocell allows programmable selection of a combinatorial output path with I/O feedback or a registered output path with registered or I/O feedback. The two dedicated clock pins provide positive-edge-triggered synchronous clock input to the registers.

Table A-4 shows the pin assignments for each EP900/EP910 macrocell, as well as the feedback, Clock group, Clear/Preset group, Output Enable group, and I/O group to which each macrocell belongs. Figure A-7 shows a block diagram of the EP900/EP910.

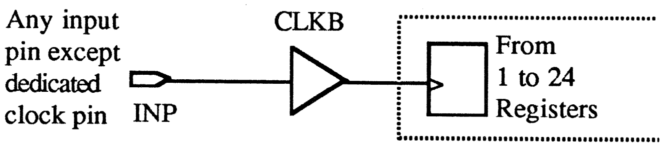
Pins 1 and 21 (2 and 24) are the two dedicated synchronous clock pins. Pin 1 feeds the registers on the left half of the device—pins 5 to 16 (6-18); pin 21 (24) feeds the registers on the right half—pins 25 to 36 (28-40). If more than 12 registers are connected to the same synchronous clock input, both pins are are needed to implement the circuit. In this case, the pin diagram in the Utilization Report contains a reminder to electrically tie the pins together. Figure A-6 shows all legal clock configurations for the EP900/EP910.

Any register can have an asynchronous clock.

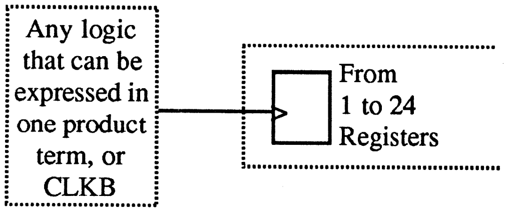
## Pin Clocking—Synchronous



## Pin Clocking—Asynchronous



## Logic or Feedback Clocking



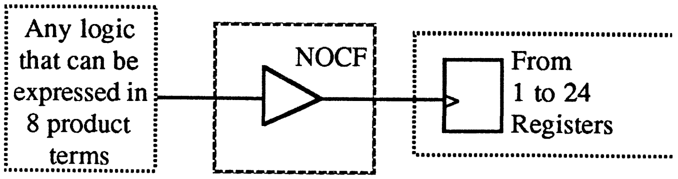


**Figure A-6. EP900/EP910 Clocking**

## Table A-4. EP900/EP910 Macrocell Groups

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 1 | 36 | global | 2 | 1 | 1 | 1 |
| 2 | 35 | global | 2 | 2 | 2 | 2 |
| 3 | 34 | global | 2 | 3 | 3 | 3 |
| 4 | 33 | global | 2 | 4 | 4 | 4 |
| 5 | 32 | global | 2 | 5 | 5 | 5 |
| 6 | 31 | global | 2 | 6 | 6 | 6 |
| 7 | 30 | global | 2 | 7 | 7 | 7 |
| 8 | 29 | global | 2 | 8 | 8 | 8 |
| 9 | 28 | global | 2 | 9 | 9 | 9 |
| 10 | 27 | global | 2 | 10 | 10 | 10 |
| 11 | 26 | global | 2 | 11 | 11 | 11 |
| 12 | 25 | global | 2 | 12 | 12 | 12 |
| 13 | 5 | global | 1 | 13 | 13 | 13 |
| 14 | 6 | global | 1 | 14 | 14 | 14 |
| 15 | 7 | global | 1 | 15 | 15 | 15 |
| 16 | 8 | global | 1 | 16 | 16 | 16 |
| 17 | 9 | global | 1 | 17 | 17 | 17 |
| 18 | 10 | global | 1 | 18 | 18 | 18 |
| 19 | 11 | global | 1 | 19 | 19 | 19 |
| 20 | 12 | global | 1 | 20 | 20 | 20 |
| 21 | 13 | global | 1 | 21 | 21 | 21 |
| 22 | 14 | global | 1 | 22 | 22 | 22 |
| 23 | 15 | global | 1 | 23 | 23 | 23 |
| 24 | 16 | global | 1 | 24 | 24 | 24 |

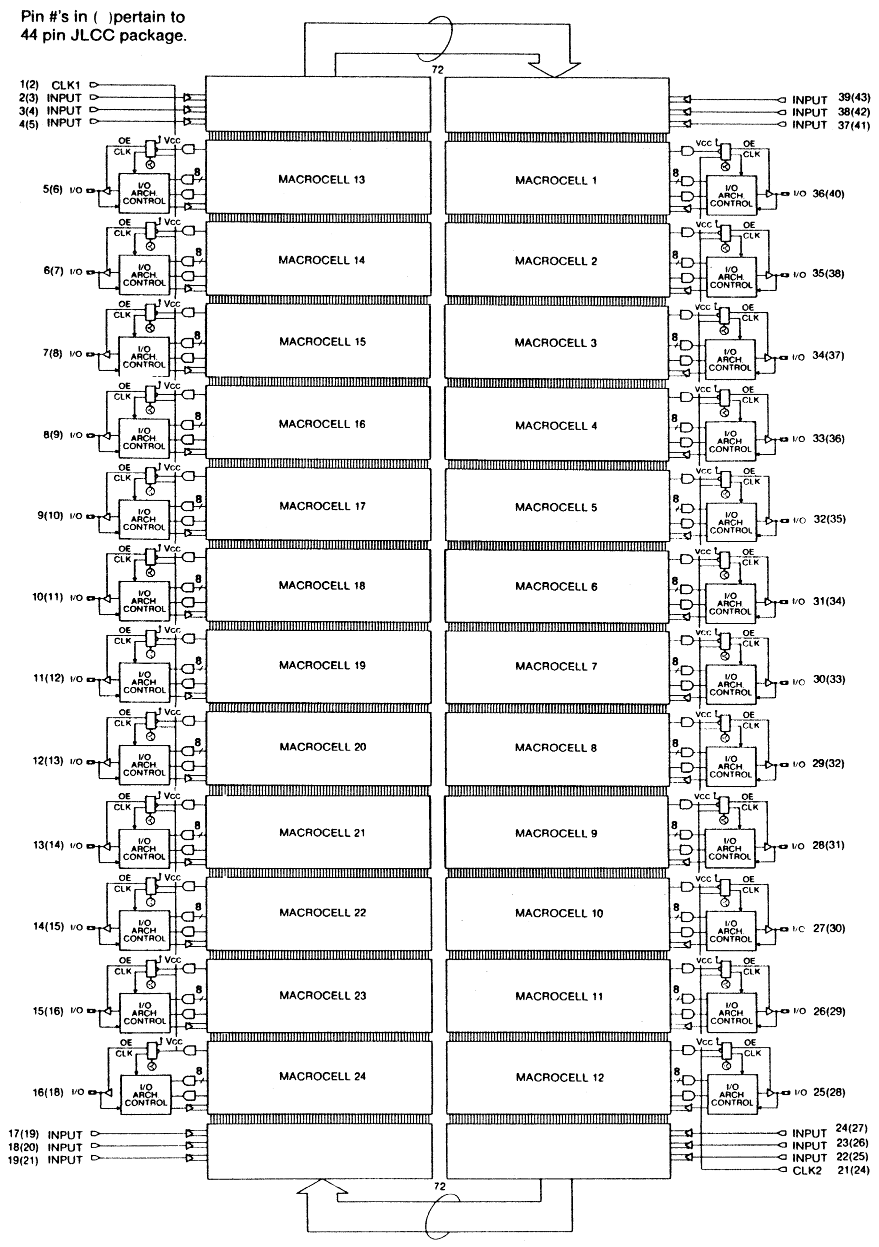


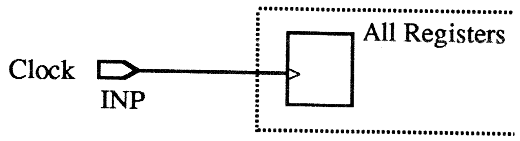
**Figure A-7. EP900/EP910 Block Diagram**

## EP1210

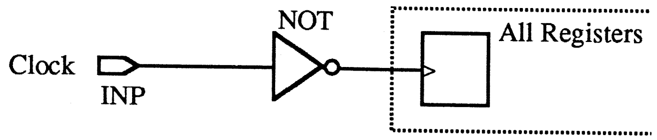
The EP1210, available in a 40-pin DIP or 44-pin J-lead package, supports 12 input pins and one dedicated clock pin, as well as 24 input/output pins. Inputs may be latched or unlatched. Each I/O pin is associated with a macrocell that has from 4 to 12 product terms and an I/O architecture control block. To conserve I/O architecture resources, macrocells are arranged in groups of 4. Four of the 8 macrocell groups have local feedback capability, which allows them to direct feedback to a specific group of macrocells. In addition, four "buried" macrocells that are not associated with any pin may be used to implement internal logic (e.g., state machines). These buried macrocells also support combinatorial feedback.

Table A-5 shows the pin assignments for each EP1210 macrocell, as well as the feedback, Clock group, Clear/Preset group, Output Enable group, and I/O group to which each macrocell belongs. Figure A-8 shows the wide variety of legal clock configurations for the EP1210; Figure A-9 shows a block diagram of the EP1210.

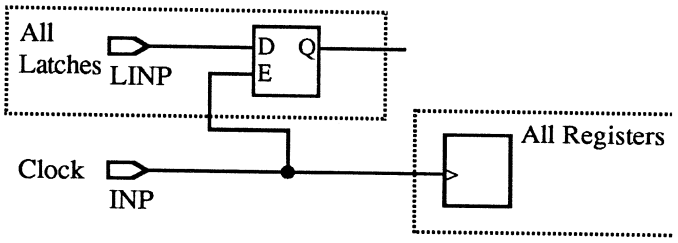
## Pin Clocking: Register Only (Mode 3)



## Inverted Pin Clocking: Register Only (Mode 2)



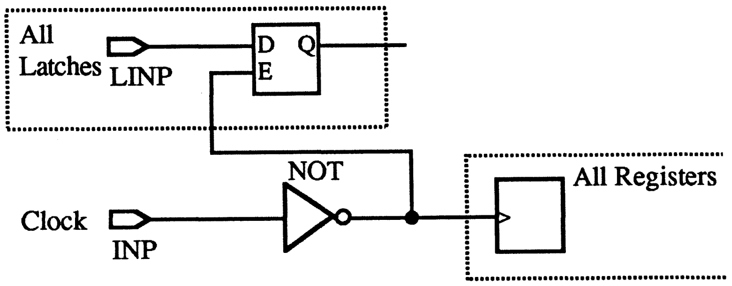
## Pin Clocking: Register and Latch (Mode 1)



*Note: Use of the registers is optional*

**Figure A-8. EP1210 Clocking (Part 1 of 4)**

## Inverted Pin Clocking: Register and Latch (Mode 0)



*Note: Use of the registers is optional*

## Separate Pins: Register, Latch Clocking (Mode 7)

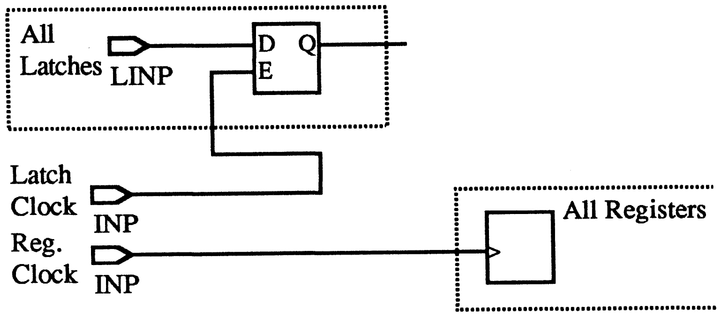
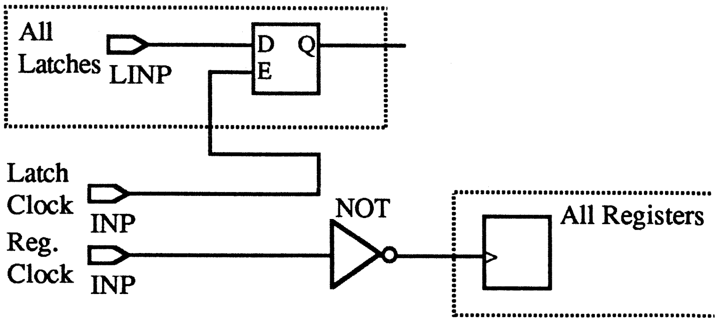


**Figure A-8. EP1210 Clocking (Part 2 of 4)**

**Separate Pins: Inverted Register, Latch Clocking (Mode 5)**



**Separate Pins: Register, Inverted Latch Clocking (Mode 6)**

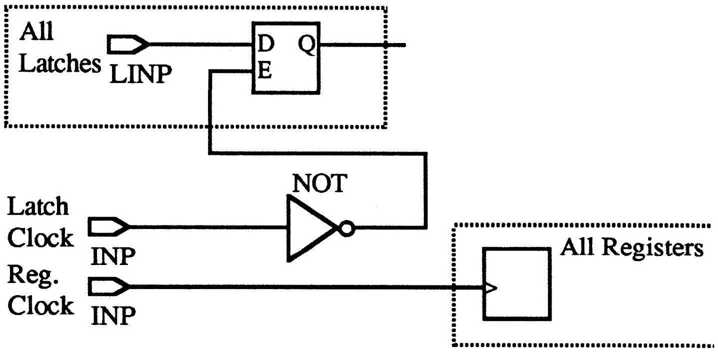


**Figure A-8. EP1210 Clocking (Part 3 of 4)**

## Separate Pins: Inverted Register, Inverted Latch Clocking (Mode 4)

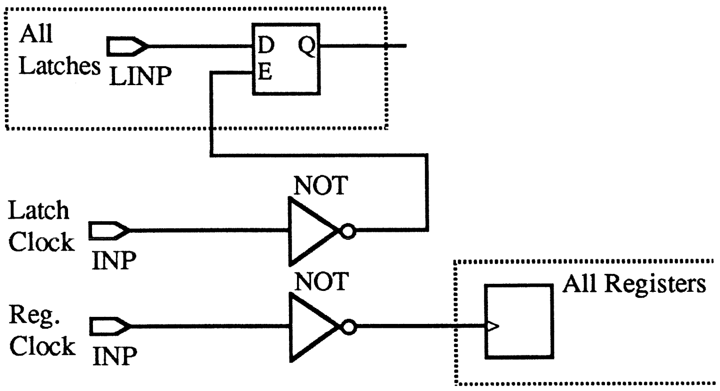


**Figure A-8. EP1210 Clocking (Part 4 of 4)**

## Table A-5. EP1210 Macrocell Groups

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 1 | 32 | local | - | 1 | 1 | 1 |
| 2 | 31 | local | - | 1 | 1 | 1 |
| 3 | 30 | local | - | 1 | 1 | 1 |
| 4 | 29 | local | - | 1 | 1 | 1 |
| 5 | 28 | local | - | 2 | 2 | 2 |
| 6 | 27 | local | - | 2 | 2 | 2 |
| 7 | 26 | local | - | 2 | 2 | 2 |
| 8 | 25 | local | - | 2 | 2 | 2 |
| 9 | 24 | global | - | 3 | 3 | 3 |
| 10 | 23 | global | - | 3 | 3 | 3 |
| 11 | 22 | global | - | 3 | 3 | 3 |
| 12 | 21 | global | - | 3 | 3 | 3 |
| 13 | none | global | - | 3 | none | 4 |
| 14 | none | global | - | 3 | none | 4 |
| 15 | none | global | - | 4 | none | 5 |
| 16 | none | global | - | 4 | none | 5 |
| 17 | 19 | global | - | 4 | 4 | 6 |
| 18 | 18 | global | - | 4 | 4 | 6 |
| 19 | 17 | global | - | 4 | 4 | 6 |
| 20 | 16 | global | - | 4 | 4 | 6 |
| 21 | 15 | local | - | 5 | 5 | 7 |
| 22 | 14 | local | - | 5 | 5 | 7 |
| 23 | 13 | local | - | 5 | 5 | 7 |
| 24 | 12 | local | - | 5 | 5 | 7 |
| 25 | 11 | local | - | 6 | 6 | 8 |
| 26 | 10 | local | - | 6 | 6 | 8 |
| 27 | 9 | local | - | 6 | 6 | 8 |
| 28 | 8 | local | - | 6 | 6 | 8 |

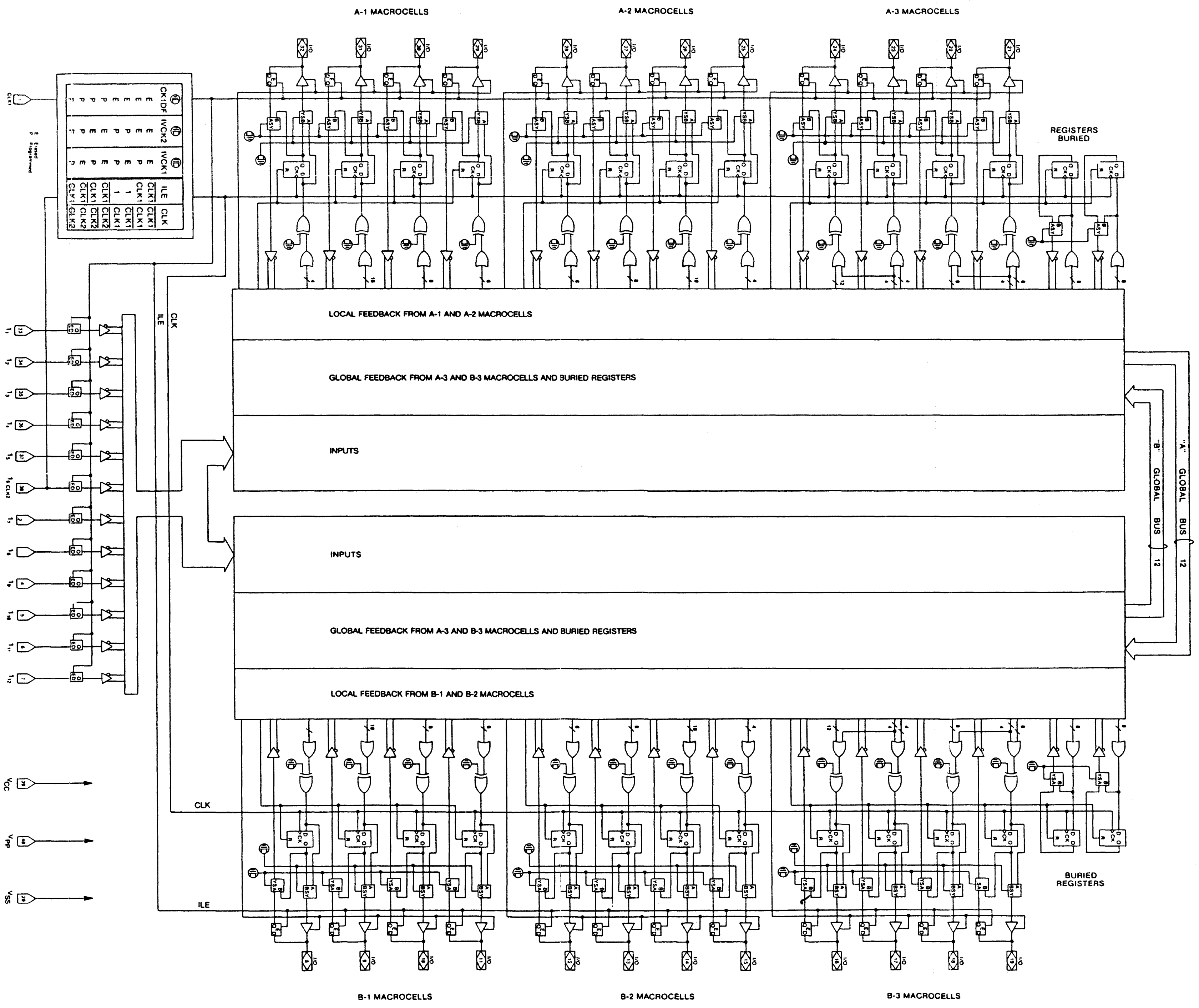


**Figure A-9. EP1210 Block Diagram**

## EPB1400 ("BUSTER")

The EPB1400 (BUSTER) is available in a 40-pin DIP or 44-pin J-lead package. This EPLD supports 8 dedicated input, 8 bus port, and 20 programmable input/output pins. BUSTER operation is controlled by two key functional blocks, the Microprocessor Interface Block (MIB) and the Programmable Logic Core Block (PLCB).

The PLCB controls 10 general-purpose macrocells in each half of the device. Eight of these macrocells are associated with the byte-wide input and output latches of the MIB. Each of the 20 general-purpose macrocells uses 8 product terms for logic requirements, 2 for Output Enable/Asynchronous Clock implementation, and one for flipflop Clear control. Each macrocell may be configured with programmable D, T, JK, or SR flipflops, or combinatorial output; registered, combinatorial, or I/O feedback; and programmable clocks. Dual I/O feedback is also available, allowing each macrocell to be used internally for buried logic functions while the associated macrocell pin is used as an input pin. Each macrocell has an asynchronous clock option that allows both synchronous and asynchronous operation on the same chip.

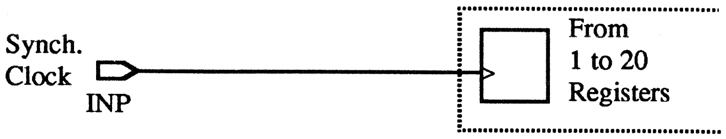
The MIB architecture provides access to an external microprocessor bus via five byte-wide elements: one bidirectional bus port with 8 dedicated pins, two input flipflops which can be configured as latched or edge-triggered, and two output latches. These five elements are controlled by dedicated Read Strobe (**/CRS**) and Write Strobe (**/CWS**) pins and seven Control macrocells. The three Control macrocells on Side 1 of the chip provide the Write Enable (**We**), Output Latch Enable (**Ole**), and Read Enable (**Re**) inputs. The four Control macrocells on Side 2 provide **We**, **Ole**, **Re**, and Output Enable (**Oe**) inputs. These Control macrocells permit user-defined logic functions to act as control inputs to the five byte-wide elements. The Clock (**Clk**), **Ole**, **Oe**, **Re**, and **We** inputs may contain up to two p-terms of logic. Table A-6 shows the pin assignments for each BUSTER macrocell, as well as the feedback, Clock group, Clear/Preset group, Output Enable group, and I/O group to which each macrocell belongs. Figure A-11 shows a detailed block diagram for BUSTER.

In addition to providing inputs to the logic array, four of the dedicated input pins may optionally be used as strobe inputs to the byte-wide elements in the MIB.

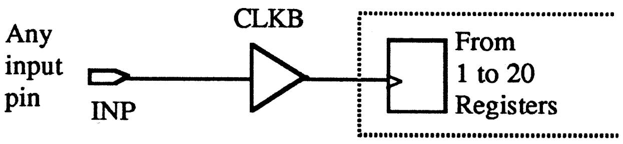
The **Ws** input to any of the 8-bit Bus primitives (**LINP8**, **RINP8**, **LBUSI**, and **RBUSI**) may be tied either directly to **GND** or the dedicated **Ws** pin—pin 25 (17). The **Rs** input to the **BUSX** primitive may be tied either directly to **GND** or the dedicated **Rs** pin—pin 36 (29). If the **Ws** and **Rs** are not connected to a pin, the logic supplied by **We** and **Oe** Control macrocells, respectively, will have full control of the flipflop. The **Clk** inputs to **LBUSO** primitive(s) may be tied either to **VCC** or to one of the dedicated clock pins on either side of the device—pins 7 and 14 (41 and 5). If the **Clk** input is unconnected, the logic supplied by the **Ole** Control macrocell will have full control of the flipflop. Pin 7 (41) feeds the registers associated with pins 1-6 and 37-40 (30-33 and 35-40); pin 14 (5) feeds the registers associated with pins 15-20 and 21-24 (6-11 and 13-16). If more than 10 registers are connected to the same synchronous clock input, both clock pins are needed to implement the circuit. In this case, the pin diagram in the Utilization Report contains a reminder to electrically tie the pins together. Figure A-10 shows all legal clock configurations for BUSTER (EPB1400).

Any register can have an asynchronous clock.

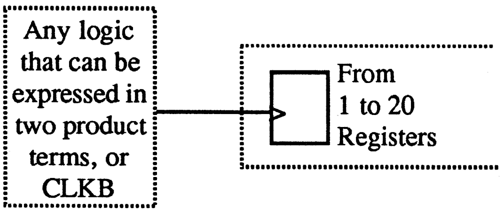
## Pin Clocking—Synchronous



## Pin Clocking—Asynchronous



## Logic or Feedback Clocking


Any logic that can be expressed in two product terms, or CLKB
From
1 to 20
Registers

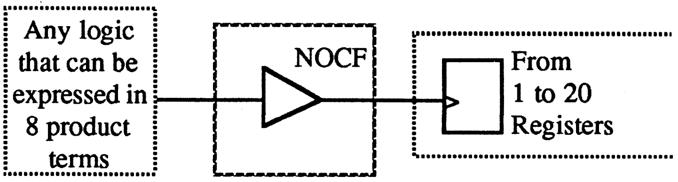
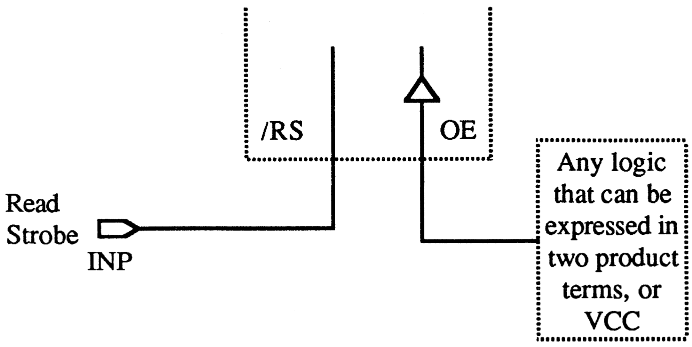


**Figure A-10. EPB1400 Clocking (Part 1 of 4)**

## Pin Clocking—BUSX


/RS
OE
Read Strobe
INP
Any logic that can be expressed in two product terms, or VCC

## Programmable Clocking—BUSX

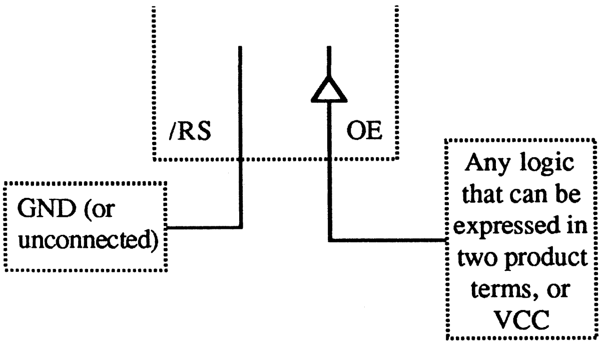
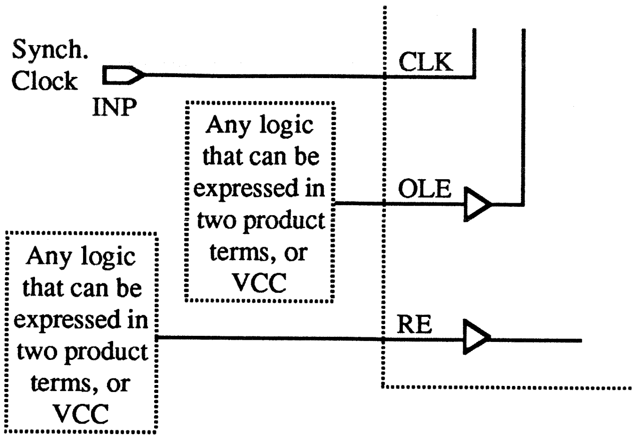

/RS
OE
GND (or unconnected)
Any logic that can be expressed in two product terms, or VCC

**Figure A-10. EPB1400 Clocking (Part 2 of 4)**

**Pin Clocking—LBUSO**


Synch.
Clock
INP
CLK
Any logic that can be expressed in two product terms, or VCC
OLE
Any logic that can be expressed in two product terms, or VCC
RE

**Programmable Clocking—LBUSO**

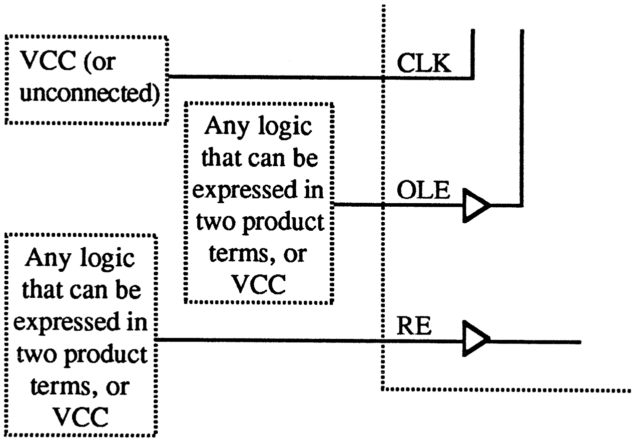
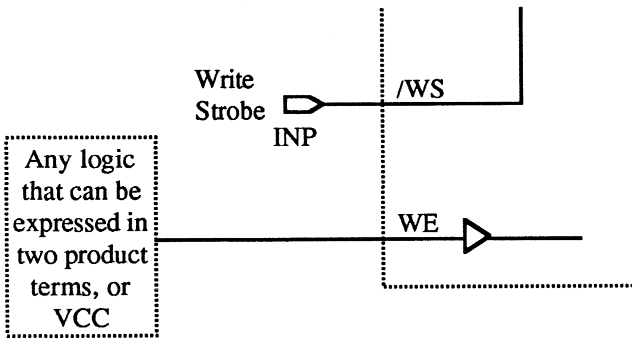

VCC (or unconnected)
CLK
Any logic that can be expressed in two product terms, or VCC
OLE
Any logic that can be expressed in two product terms, or VCC
RE

**Figure A-10. EPB1400 Clocking (Part 3 of 4)**

## Pin Clocking—LBUSI, LINP8, RBUSI, RINP8


Write
Strobe
INP
/WS
Any logic that can be expressed in two product terms, or VCC
WE

## Programmable Clocking—LBUSI, LINP8, RBUSI, RINP8

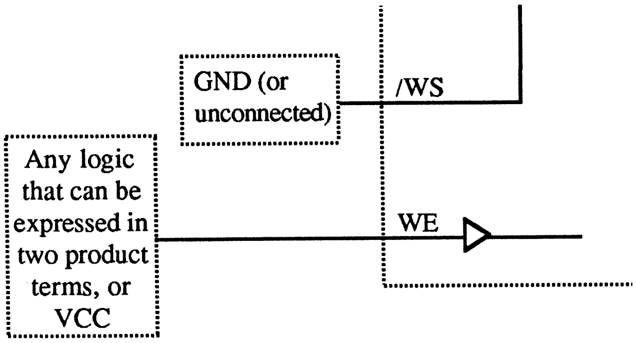

GND (or unconnected)
/WS
Any logic that can be expressed in two product terms, or VCC
WE

**Figure A-10. EPB1400 Clocking (Part 4 of 4)**

## Table A-6. EPB1400 Macrocell Groups

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 1 | 15 | global | 1 | 1 | 1 | 1 |
| 2 | 16 | global | 1 | 2 | 2 | 2 |
| 3 | 17 | global | 1 | 3 | 3 | 3 |
| 4 | 18 | global | 1 | 4 | 4 | 4 |
| 5 | 19 | global | 1 | 5 | 5 | 5 |
| 6 | 20 | global | 1 | 6 | 6 | 6 |
| 7 | 21 | global | 1 | 7 | 7 | 7 |
| 8 | 22 | global | 1 | 8 | 8 | 8 |
| 9 | 23 | global | 1 | 9 | 9 | 9 |
| 10 | 24 | global | 1 | 10 | 10 | 10 |
| 11 | 37 | global | 2 | 11 | 11 | 11 |
| 12 | 38 | global | 2 | 12 | 12 | 12 |
| 13 | 39 | global | 2 | 13 | 13 | 13 |
| 14 | 40 | global | 2 | 14 | 14 | 14 |
| 15 | 1 | global | 2 | 15 | 15 | 15 |
| 16 | 2 | global | 2 | 16 | 16 | 16 |
| 17 | 3 | global | 2 | 17 | 17 | 17 |
| 18 | 4 | global | 2 | 18 | 18 | 18 |
| 19 | 5 | global | 2 | 19 | 19 | 19 |
| 20 | 6 | global | 2 | 20 | 20 | 20 |

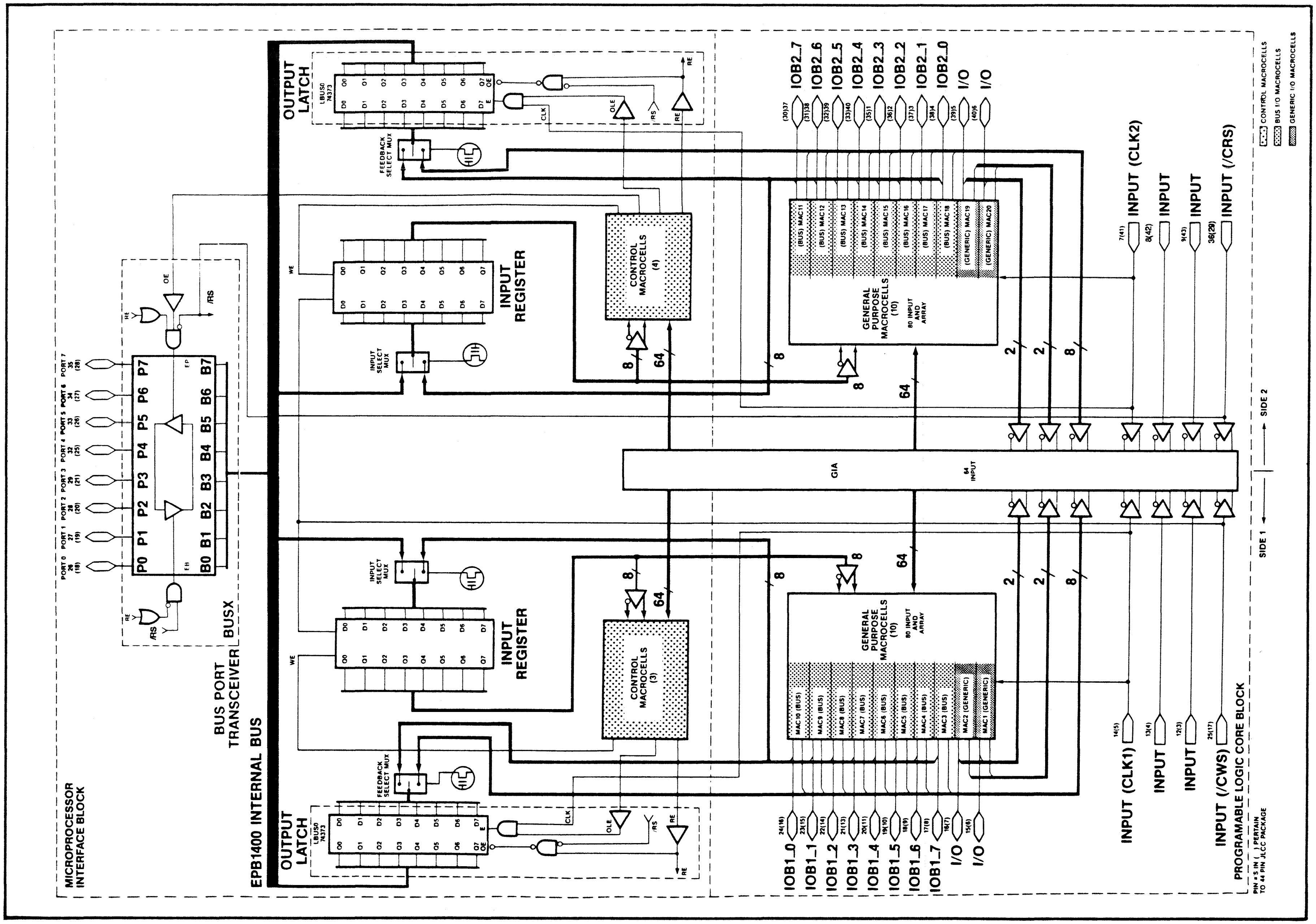


Figure A-11. EPB1400 Block Diagram

## EP1800

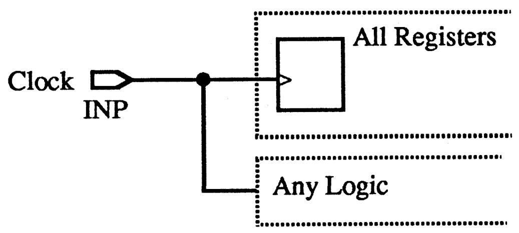
The EP1800, available in a 68-pin J-lead or 68-pin grid-array package, supports 16 dedicated input and 48 programmable input/output pins. It consists of four quadrants (A,B, C, and D), each of which contains 12 macrocells. Each I/O pin is associated with a macrocell that has 10 product terms and an I/O architecture control block. Eight product terms are used for logic requirements, one for Output Enable/Asynchronous Clock implementation, and one for flipflop clear control. The I/O architecture control block allows programmable selection of D, T, JK, or SR flipflops, or combinatorial output for each of the 48 macrocells. In all macrocells, the feedback path can be programmed to be registered, combinatorial, or I/O. The global macrocells 9-12, 13-16, 44-47, and 57-60 can be programmed as local macrocells that support dual I/O feedback, i.e., simultaneous internal and I/O feedback. One input pin in each quadrant may optionally be used as a synchronous clock input. Each macrocell also has an asynchronous clock option that allows both synchronous and asynchronous operation on the same EPLD.

Table A-7 shows the pin assignments for each EP1800 macrocell, as well as the feedback, Clock group, Clear/Preset group, Output Enable group, and I/O group to which each macrocell belongs. Figure A-13 shows a block diagram of the EP1800.

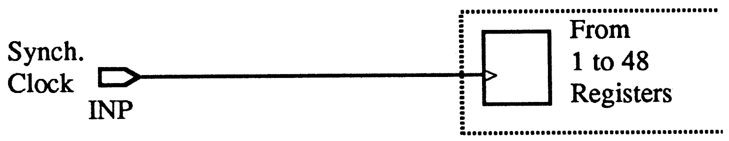
Pins 17, 19, 51, and 53 (L5, L6, A7 and A6) are the four synchronous clock inputs. Pin 17 (L5) feeds the registers on quadrant A—pins 2 to 13 (F1-L3); pin 19 (L6) feeds the registers on quadrant B—pins 23 to 34 (L8-G11); pin 51 (A7) feeds the registers on quadrant C—pins 36 to 47 (F11-A9); and pin 53 (A6) feeds the registers on quadrant D—pins 57 to 68 (A4-E1). If more than 12 registers are connected to the same synchronous clock input, additional clock inputs are used to accomplish the required clocking. In this case, the pin diagram in the Utilization Report contains a reminder to electrically tie the pins together. Figure A-12 shows all legal clock configurations for the EP1800.

Any register can have an asynchronous clock.

**Pin Clocking with Logic Fanout**



**Pin Clocking—Synchronous**



**Pin Clocking—Asynchronous**

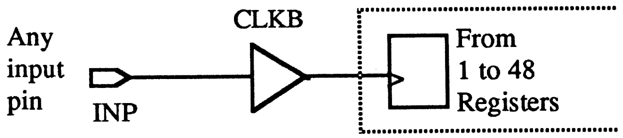


**Figure A-12. EP1800 Clocking (Part 1 of 2)**

## Logic or Feedback Clocking

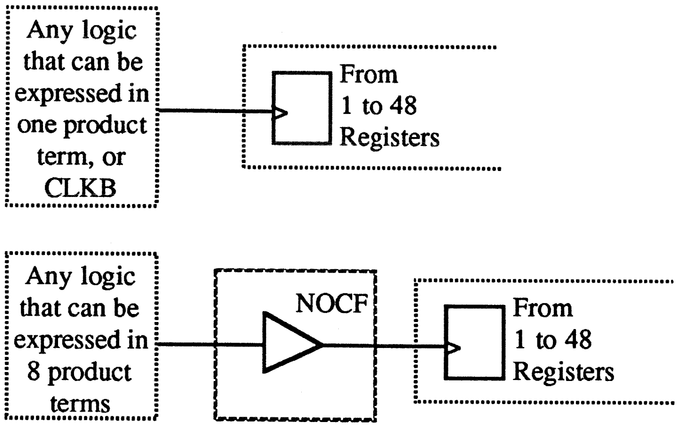

Any logic that can be expressed in one product term, or CLKB
From 1 to 48 Registers
Any logic that can be expressed in 8 product terms
NOCF
From 1 to 48 Registers

**Figure A-12. EP1800 Clocking (Part 2 of 2)**

## Table A-7. EP1800 Macrocell Groups (Part 1 of 2)

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 1 | 2 | local | 1 | 1 | 1 | 1 |
| 2 | 3 | local | 1 | 2 | 2 | 2 |
| 3 | 4 | local | 1 | 3 | 3 | 3 |
| 4 | 5 | local | 1 | 4 | 4 | 4 |
| 5 | 6 | local | 1 | 5 | 5 | 5 |
| 6 | 7 | local | 1 | 6 | 6 | 6 |
| 7 | 8 | local | 1 | 7 | 7 | 7 |
| 8 | 9 | local | 1 | 8 | 8 | 8 |
| 9 | 10 | dual | 1 | 9 | 9 | 9 |
| 10 | 11 | dual | 1 | 10 | 10 | 10 |
| 11 | 12 | dual | 1 | 11 | 11 | 11 |
| 12 | 13 | dual | 1 | 12 | 12 | 12 |
| 13 | 23 | dual | 2 | 13 | 13 | 13 |
| 14 | 24 | dual | 2 | 14 | 14 | 14 |
| 15 | 25 | dual | 2 | 15 | 15 | 15 |
| 16 | 26 | dual | 2 | 16 | 16 | 16 |
| 17 | 27 | local | 2 | 17 | 17 | 17 |
| 18 | 28 | local | 2 | 18 | 18 | 18 |
| 19 | 29 | local | 2 | 19 | 19 | 19 |
| 20 | 30 | local | 2 | 20 | 20 | 20 |
| 21 | 31 | local | 2 | 21 | 21 | 21 |
| 22 | 32 | local | 2 | 22 | 22 | 22 |
| 23 | 33 | local | 2 | 23 | 23 | 23 |
| 24 | 34 | local | 2 | 24 | 24 | 24 |

*Note: "**dual**" indicates a local as well as a global feedback path.*

## Table A-7. EP1800 Macrocell Groups (Part 2 of 2)

| Macro-cell | Pin | Feedback | Clock Group | Clear/ Preset Group | OE Group | I/O Group |
|---|---|---|---|---|---|---|
| 25 | 36 | local | 3 | 25 | 25 | 25 |
| 26 | 37 | local | 3 | 26 | 26 | 26 |
| 27 | 38 | local | 3 | 27 | 27 | 27 |
| 28 | 39 | local | 3 | 28 | 28 | 28 |
| 29 | 40 | local | 3 | 29 | 29 | 29 |
| 30 | 41 | local | 3 | 30 | 30 | 30 |
| 31 | 42 | local | 3 | 31 | 31 | 31 |
| 32 | 43 | local | 3 | 32 | 32 | 32 |
| 33 | 44 | dual | 3 | 33 | 33 | 33 |
| 34 | 45 | dual | 3 | 34 | 34 | 34 |
| 35 | 46 | dual | 3 | 35 | 35 | 35 |
| 36 | 47 | dual | 3 | 36 | 36 | 36 |
| 37 | 57 | dual | 4 | 37 | 37 | 37 |
| 38 | 58 | dual | 4 | 38 | 38 | 38 |
| 39 | 59 | dual | 4 | 39 | 39 | 39 |
| 40 | 60 | dual | 4 | 40 | 40 | 40 |
| 41 | 61 | local | 4 | 41 | 41 | 41 |
| 42 | 62 | local | 4 | 42 | 42 | 42 |
| 43 | 63 | local | 4 | 43 | 43 | 43 |
| 44 | 64 | local | 4 | 44 | 44 | 44 |
| 45 | 65 | local | 4 | 45 | 45 | 45 |
| 46 | 66 | local | 4 | 46 | 46 | 46 |
| 47 | 67 | local | 4 | 47 | 47 | 47 |
| 48 | 68 | local | 4 | 48 | 48 | 48 |

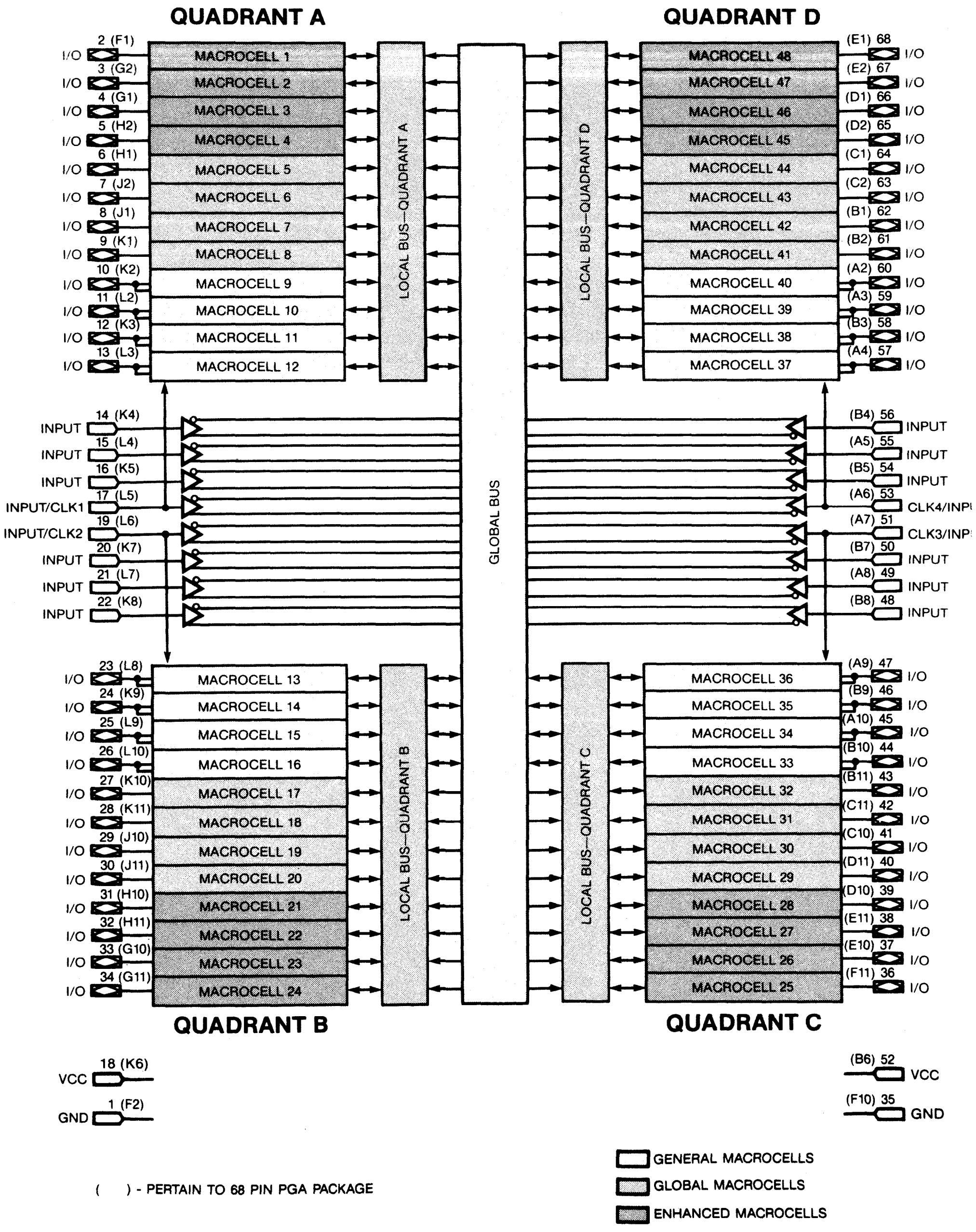


**Figure A-13. EP1800 Block Diagram**

## APPENDIX B

# ADF-to-LEF Translation

Altera Design File-to-Logic Equation File (ADF-to-LEF) translation occurs when the ADP's Translator module converts the Flattener module output into a Logic Equation File (LEF). During translation, I/O primitives are copied to the resource section, and logic primitives are translated into Boolean logic equations and written to the LEF's Equations Section. The design is thus expressed as a series of Boolean equations.

Table B-1 shows the possible inputs and resulting outputs for logic primitives in an Altera Design File-to-Logic Equation File translation.

Table B-1. ADF-to-LEF Translation

| LOGIC PRIMITIVES | LEF EQUATIONS SECTION |
|---|---|
| Out = AND (In1, In2, ... Inn) | Out = In1 * In2 *...* Inn ; |
| Out = OR (In1, In2 ... Inn) | Out = In1 + In2 +...+ Inn ; |
| Out = NAND (In1, In2, ... Inn) | Out = In1' + In2' +...+ Inn ; |
| Out = NOR (In1, In2, ... Inn) | Out = In1' * In2' *...* Inn ; |
| Out = NOT (In) | Out = In' ; |
| Out = XOR (In1, In2) | Out = In1' * In2 + In1 * In2'; |

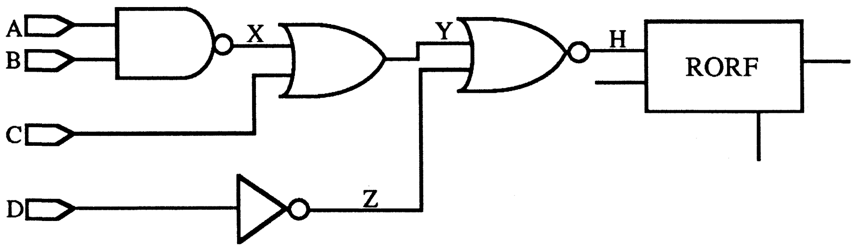


Figure B-1. Sample Circuit for ADF-to-LEF Translation

Figure B-1 shows a sample circuit. The ADF-to-LEF translation for this circuit is as follows:

| ADF | | | LEF | | |
|---|---|---|---|---|---|
| X | = | NAND (A,B) | X | = | A' + B' ; |
| Y | = | OR (X,C) | Y | = | X + C ; |
| Z | = | NOT (D) | Z | = | D' ; |
| H | = | NOR (Y,Z) | H | = | Y' * Z' ; |

As the equation undergoes further processing, the translation yields the following:

$$Y = (A' + B') + C$$

and

$$H = ((A' + B') + C)' * (D')'$$

The final sum-of-products of the logic equation for this macrocell is:

$$H = A * B * C' * D$$

Each I/O primitive (e.g., **COCF**, **NORF**) has an equation associated with it. The **P**, **C**, and **Oe** inputs to these primitives may also have simple equations.

Figure B-2 shows the post-minimization LEF Analyzer output for the **BEVDIS** sample design described in *Boolean Equation Entry* in the ***A+PLUS User Guide***.

The LEF does not retain user comments from the input design file. However, the LEF Header Section always includes new lines that record the input filename(s) and the ADP options requested for design processing. The LEF Analyzer also inserts/completes the Options Section with default values for the Turbo-Bit and Security Bit if one or both were not specified in the input design file. In some minimized designs, the LEF Analyzer inserts new comments noting that primitives were minimized and changed.

```
Your Name
Your Company
9/30/87
1.00
B
EP310
Beverage Dispenser Controller

Input files : BEVDIS.ADF
ADP Options: Minimization = Yes, Inversion Control = No, LEF Analysis = Yes

<LEF Version information>

OPTIONS:  TURBO  =  ON, SECURITY  =  OFF
PART:
        EP310
INPUTS:
        ENABLE@7, RESET@5, COINDROP, CUPFULL, CLOCK
OUTPUTS:
        DROPCUP@14, STROBE, POURDRNK
NETWORK:
        CLOCK  =   INP(CLOCK)
        ENABLE  =   INP(ENABLE)
        RESET  =   INP(RESET)
        COINDROP  =   INP(COINDROP)
        CUPFULL  =   INP(CUPFULL)
        DROPCUP,DROPCUP  =   RORF(DROPCUPd, CLOCK, NEWCYCLE,
                                 RESET, ENABLE)
        STROBE  =   CONF(STROBEc, VCC)
        POURDRNK,POURDRNK  =   RORF(POURDRKd, CLOCK,
                                 NEWCYCLE, RESET, ENABLE)
EQUATIONS:
        POURDRKd  =   DROPCUP * POURDRNK'
                  +   DROPCUP' * POURDRNK * CUPFULL';

        STROBEc  =   CLOCK' * DROPCUP' * POURDRNK
                 +   CLOCK' * DROPCUP * POURDRNK';

        NEWCYCLE  =   DROPCUP * POURDRNK;

        DROPCUPd  =   COINDROP * DROPCUP' * POURDRNK';

END$
```

**Figure B-2. BEVDIS.LEF**

APPENDIX C

# BNF Rules

This document uses the Backus-Naur Form (BNF) to define the syntax of the Altera Design File, the State Machine File and the JEDEC Standard File. BNF adheres to the following notation rules:

| SYMBOL | DEFINITION |
|---|---|
| ::= | "is defined as" |
| '...' | literal string of characters |
| <...> | identifiers |
| [...] | optional items |
| { ... } | repeated items (zero or more times) |
| ...\|... | indicates a choice between items |
| **:n:n** | suffix indicates a range (e.g., **<name char>:1:8** means "from 1 to 8 name characters") |

APPENDIX D

# JEDEC Standard

This appendix defines Altera's implementation of the JEDEC (Joint Electron Device Engineering Council) Standard (from JEDEC Council Ballot JCB-82-2, formulated under the cognizance of JC-42.1 Committee on Bipolar Memory standardization). JEDEC files provide a standard data transfer format between the A+PLUS software and the Logic Device Programmer unit.

## Design Specification

The design specification is the first field in the format. It consists of the following:

1. An ASCII **STX** (02 hex)
   (This character begins the transmission and is followed by ASCII characters representing the design specification identifiers.)

2. User's Name

3. Company's Name

4. Date of the Design Specification

5. Part Number of the Design Specification

6. Revision of the Design Specification

7. Altera Part Number

8. Comments

9. Asterisk (*)
(This character indicates termination of the identification information.)

## Note (N) Field

The **N** fields are used to save part name and pin name and number information. These fields appear in the following format:

**ND_EPnnnn_***
**N@_PINNAME_@nn***

where **EPnnnn** is an Altera part name, **PINNAME** is the user-assigned pin name, and **nn** is the virtual pin number.

## Programming Information

Each EPROM bit of an Altera EPLD is assigned a decimal number. Each numbered bit has two possible states:

- **0** (zero): specifies a low-resistance link;
- **1** (one): specifies a high-resistance link.

Bit information is presented in the Link (**L**) field.

## Bit Count (QF) Field

The **QF** field has the following format:

**QFnnnn***

where **nnnn** is the (decimal) number of programmable bits in the EPLD.

## Security (G) Field

This field indicates whether or not to program the security bit.

**G0*** indicates not to program the security bit.
**G1*** indicates to program the security bit.

## Link (L) Field

Specific bit information is preceded by an **L**. It is followed immediately by a variable-length decimal number that indicates the starting EPROM cell or bit number for a string of data. The first **1** or **0** is preceded by a space, and the data string is terminated by an asterisk (*). The **L** field can be any desired length; any number of **L** fields may be specified. If the state of a bit is specified more than once, the last state replaces all preceding entries for that bit.

## End of Transmission

A JEDEC File is concluded with an **ETX** (03 hex). A sum-check of 4 ASCII hex characters follows immediately. This sum-check is the 16-bit sum of the ASCII values of the transmitted characters between, and including, the **STX** and the **ETX**. The parity bit is excluded from this calculation.

## Legal Characters

Table D-1 shows legal ASCII characters for JEDEC files (all others are invalid). Note that carriage returns and line feeds may be used anywhere in the format to improve readability.

**Table D-1. Legal Characters in the JEDEC File**

| CHARACTER | HEX VALUE | DEFINITION |
|---|---|---|
| STX | 02 hex | Start of Text |
| ETX | 03 hex | End of Text |
| LF | 0A hex | Line Feed |
| CR | 0D hex | Carriage Return |
| All printable characters | 20 to 7E hex | |

APPENDIX E

# Electronic Design Support Service

Altera's Electronic Design Support Service includes an electronic bulletin service, named **Fido**, as its vehicle for modem-based user support. Fido's hierarchical menu structure provides easy access to Electronic Application Briefs and Electronic Application Utilities, messages to and from Altera Applications, and file upload and download services. Fido supports the Crosstalk, Xmodem, Kermit, Ascii, Minitel, Modem7, and Telink file-transfer protocols.

**Modem Number:**

**(408) 249-1100**

This appendix is divided into eight steps, each of which is described in the following pages:

1. Logging On
2. Main Menu
3. Message Section
4. Reading Messages
5. Files Section
6. Uploading a File to Altera
7. Downloading a File From Altera
8. Logging Off

## 1. Logging On

First, set your modem communication parameters to 8 bits, 1 stop bit, no parity, and dial the modem number given above. Fido operates at 300 and 1200 baud and automatically adjusts to the baud rate of your system after reading the first two characters sent by your modem.

You must press **<Space>** several times to set the baud rate before anything can be typed back to you.

After the baud rate is set, the following display appears on the screen:

```
+--------------------------------------------------------------------------+
| Altera Electronic Bulletin Service    Call (408) 984-2805 x102 for Help  |
+-------------------------------------+------------------------------------+
| If you haven't purchased an         |                                    |
| Extended Software Warranty --       | If you have --                     |
|                                     |                                    |
| Log on as shown below:              | Log on using the logon name and    |
|                                     | password that came with your       |
|                                     | Altera Extended Software Warranty  |
| Logon name: Guest                   | certificate.                       |
| Press <Enter>:                      |                                    |
| Altera? [Y,n]: y                    |                                    |
| Wait ...                            |                                    |
| Password: EPLD                      |                                    |
+-------------------------------------+------------------------------------+
  Logon  name:
```

If you have purchased an Extended Software Warranty, you will have access to all Fido file and message areas. If not, your access will be limited to a subset of these areas.

If you have an Extended Software Warranty, go through the following steps:

1. Type your logon name and press **<Enter>**.
2. Press **<Enter>** again.
3. Type **Y** and press **<Enter>**.
4. Type your password and press **<Enter>**.

If not, log on as follows:

1. Type **Guest** and press **<Enter>**.
2. Press **<Enter>** again.
3. Type **Y** and press **<Enter>**.
4. Type the password **EPLD** and press **<Enter>**.

After you have logged on, the following display appears on the screen:

```
            Welcome to Altera's 24-hour Electronic Bulletin Service

   ------------File Areas------------

  1 ... To Altera                      Some file areas are accessible only
  2 ... From Altera                    to customers who have purchased
                 .                     an Extended Software Warranty.
                 .
                 .
                 .                     NOTE: Crosstalk users should
                 .                     use XXMODEM and RXMODEM
                 .                     commands instead of XMIT and
                 .                     RECV.
                 .                     See Message #1 in the Message
                 .                     Section for details.
 12 ... Electronic Application Briefs
 13 ... Documentation

 Type y or n after "More?"             Type ^C or ^K to abort a display.
 More?
```

If you have an Extended Software Warranty, you will have access to all areas. If not, you may enter File Areas 1, 2, and 12.

Type **n** at the **More?** prompt to display Fido's Main Menu, or, type **y** to display bulletin information, as shown in the following example. Altera Bulletins provide information on electronic Applications Department publications, file transfer protocols, commands needed to communicate with Fido, software update information, and new product announcements.

```
June 23, 1987: Electronic Application Briefs in File Area #12
------------------------------------------------------------------------------

Electronic Application Briefs (EABs) provide up-to-date information for
effectively using Altera EPLDs and Electronic Application Utilities (EAUs)
provide design-support software. For information on downloading EABs,
type EAB000-1.TXT in File Area #12 or call Altera Applications at
(408) 984-2805  x102.

New EABs and EAUs as of June 22, 1987:
   EAB015:  How to inplement Monostable, RC, and Crystal Oscillators in
            EPLDs

Current list of EABs and EAUs.
   EAB000:  How to use EAB's.
   EAB009:  Designing Asynchronous Latches with EPLDs.
      .
      .
      .
   EAB055:  Using Dual Feedback with the EP1800

More?
```

Press **y** to continue reading bulletins. Press **n** to exit to the Main Menu.

## 2. Main Menu

Main Menu commands are as follows:

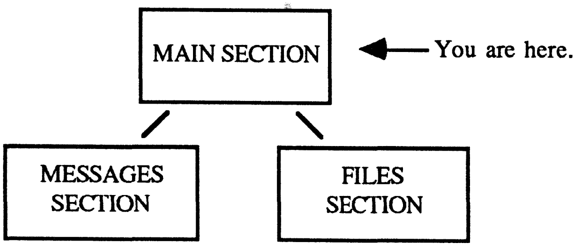
```
MAIN Commands:
M   =  Messages,    F  =  Files,
G   =  Goodbye,     A  =  ALTERA-LOGON,
B   =  Bulletin,    V  =  Version

Main: M F G A B V (? for help)
>
```

Type:

**? <Enter>**

to display the help screen for the Main Menu:

```
ALTERA-LOGON. .  Do this first                  Altera Applications
Files. . . . . . . . . .  Go to the FILES SECTION        (408) 984-2805 X-102
                 to upload or download files
Messages. . . . . . .  Go to the MESSAGES SECTION
                 to read messages
Goodbye . . . . . . .  Log off
Bulletin  . . . . . . .  Read the Bulletin
```



If you need assistance, you may type ? at any time.

## 3. Message Section

To view messages in the message area, type at the Main Menu prompt:

**m <Enter>**

The screen displays the Messages Menu:

```
Msg Area #1: General Messages
1 messages, highest is #1, last you read was #1
Want to check for mail? [Y,n]:_
```

If you type **n <Enter>**, the screen displays the Messages Menu:

```
Msg Area #1: General Messages
L   =  List-messages,    R  =  Read-messages,
I   =  Search,           G  =  Goodbye,
M   =  Main-menu
Msg: L R I G M (? for help)
>
```

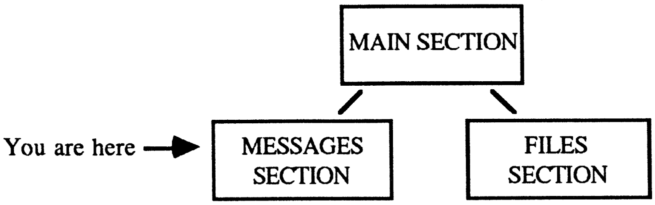
Type:

**? <Enter>**

to display the help screen for the Messages Menu:

```
List-messages . . List message headers
                  "L" - lists most recent message
                        first
Read-messages . . Read messages
Inquire . . . . . . . Search message headers for a word
Goodbye. . . . . . Log off
Main-menu . . . . Return to the MAIN SECTION
```

Altera Applications
(408) 984-2805 X-102



## 4. Reading Messages

To read a message, type at the Messages Menu prompt:

**r <Enter>**

to display the Read Commands shown here:

```
Read Message:
number, *, Next, Previous,
Quit
[1] 1 - 1 n P Q (? for help)
>
```

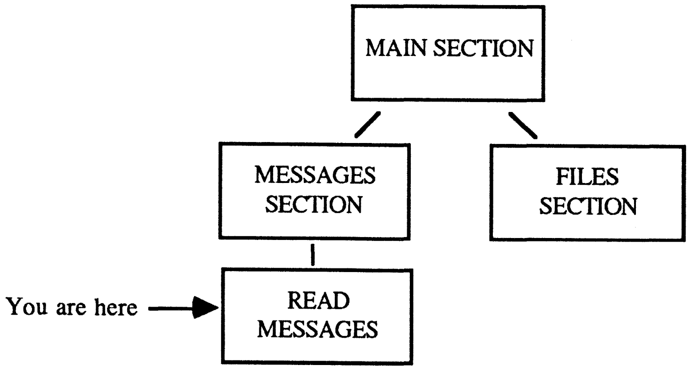
Type:

**? <Enter>**

to display the help screen for the Read Commands:

| | |
|---|---|
| * . . . . . . . . . . | Read the most recent message |
| <number> . . . . . | Read message <number> |
| Next . . . . . . . . | Read the NEXT highest message |
| Previous . . . . . . | Read the PREVIOUS message |
| Quit. . . . . . . . . | Return to MESSAGE SECTION prompt |

Altera Applications
(408) 984-2805 X-102



Type:

```
1 <Enter>
```

to display message #1 shown here:

```
#1    106   10 May 1986    17:57:23
From:   Applications
To:     All
Subj:   Using XTALK (Crosstalk) to send and receive files.

Crosstalk XVI supports two error-free file transfer protocols:
XMODEM and Crosstalk's own protocol. To Upload (send) and
Download (receive) files, use Crosstalk's XMODEM commands
XXMODEM and RXMODEM instead of XMIT and RQ.
```

Read any additional messages by pressing <Enter> or type:

```
q <Enter>
```

to return to the Message Section menu. To return to the Main Menu, type:

```
m <Enter>
```

The Main Menu is displayed:

```
MAIN Commands:
M   =  Messages,    F  =  Files,
G   =  Goodbye,     A  =  ALTERA-LOGON,
B   =  Bulletin,    V  =  Version

Main: M F G A B V (? for help)
>
```

Next, go to the Files Section.

## 5. Files Section

To enter the Files Section, type:

**f <Enter>**

The Files Section menu is displayed:

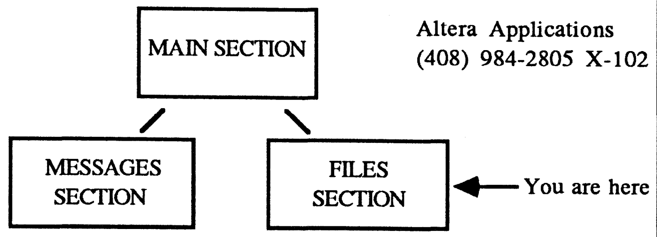
```
File Area #2: From Altera
A   =   AREA-CHANGE,       L  =  Locate,
F   =   File-Directory,    T  =  Type a .TXT file,
G   =   Goodbye,           U  =  Upload-to-Altera,
D   =   Download-to-You,   M  =  Main-menu

File: A L F T G U D M (? for help)
>
```

Type:

**? <Enter>**

to display the help screen for the Files Section Menu:

```
AREA (DO THIS FIRST) Select a FILE AREA
Files  . . . . . . . . . List directory of files in this area
Locate  . . . . . . . Locate a file or files, searching all areas
Type  . . . . . . . . Type a .TXT file to the screen
Goodbye  . . . . . . Log off
Upload-to-Altera  . . Send a file to Altera (Crosstalk users see
                      Message #1).
Download-to-You . . Get a file from Altera (Crosstalk users see
                      Message #1).
Main-menu  . . . . Return to the MAIN SECTION
```



## 6. Uploading a File to Altera

☞ Go through the following steps only if you actually plan to upload a file to Altera. If you don't transfer a file after executing the steps, the system will wait for about two minutes before typing anything back to you.

To send (i.e., upload) a file to Altera, you must go to File Area #1. First, execute the **CHANGE-AREA** command, which selects the File Area needed to execute the remaining commands:

**a <Enter>**

The File Areas Menu is displayed:

```
--------File  Areas--------
1     ... To  Altera
2     ... From  Altera
3     ... TTL  Library:  Flip-flops
              .
              .
              .
13    ... Documentation
File  Area,  or  Quit:_
```

To change to the Upload File Area, type:

**1 <Enter>**

The following menu is displayed:

```
File  Area  #1:  To  Altera
A    =   AREA-CHANGE,        L   =   Locate,
F    =   File-Directory,     T   =   Type  a  .TXT  file,
G    =   Goodbye,            U   =   Upload-to-Altera,
D    =   Download-to-You,    M   =   Main-menu

File:  A  L  F  T  G  U  D  M  (?  for  help)
>
```

Type:

**u <Enter>**

to display the following message:

```
A)scii, K)ermit, X)modem, B)atch, T)elink, ? for help
add C for CRC, i.e. TC, etc:_
```

Type:

**? <Enter>**

to display the help screen shown here:

| Uploading a file to Altera | Downloading a file from Altera |
|---|---|
| 1. If using XMODEM type "X" | 1. If using XMODEM type "X" |
| If using Crosstalk type "X" | If using Crosstalk type "X" |
| If using MODEM7 type "B" | If using MODEM7 type "B" |
| If using TELINK type "T" | If using TELINK type "T" |
| If using MINITEL type "T" | If using MINITEL type "T" |
| If using KERMIT type "K" | If using KERMIT type "K" |
| For an ASCII transfer type "A" | For an ASCII transfer type "A" |
| 2. and press <Enter> key. | 2. and press <Enter> key. |
| 3. Type name of file you are sending | 3. Type name of file you are getting |
| 4. and press <Enter> key. | 4. and press <Enter> key. |
| 5. If using Crosstalk press Attention key, type "XXMODEM", type the name of the file you are sending, and press the <Enter> key. | 5. If using Crosstalk press Attention key, type "RXMODEM", type the name of the file you are getting, and press the <Enter> key. |
| If not using Crosstalk, type your terminal program's "send file" command. | If not using Crosstalk, type your terminal program's "receive file" command. |

Select the desired file-transfer protocol by typing the appropriate letter and pressing **<Enter>**.

For example, to select the XMODEM protocol for Crosstalk, type:

**x <Enter>**

The following information is displayed:

```
XMODEM transfer
Filename(s):
```

Type the name of the file to be uploaded. The following message is displayed:

```
Ready to receive <filename>
38 blocks, 00:54 transfer time
Start now, or Control-C to abort
```

Fido now expects you to activate the Send File command on the communication program. For example, in Crosstalk type the Attention character (normally **<Esc>**) followed by **XX <filename>**.

After you have sent the file, Fido requests you to specify the name of the Altera Applications Engineer who is expecting your file by asking:

```
Who gets this file?
```

Type the name of the Altera Applications Engineer.

☞ For protocols supported by your system, refer to your own system documentation.

## 7. Downloading a File From Altera

☞ Go through the following steps only if you actually plan to download a file from Altera. If you don't transfer a file after executing the steps, the system will wait for about one minute before typing anything back to you.

To download a file from Altera, you must go to File Area #2. First, execute the **CHANGE-AREA** command, which selects the File Area needed to execute the remaining commands:

**a <Enter>**

The File Areas Menu is displayed:

```
--------File Areas--------
1     ... To Altera
2     ... From Altera
3     ... TTL Library: Flip-flops
              .
              .
              .
13    ... Documentation
File Area, or Quit:
```

To change to the Download File Area, type:

**2 <Enter>**

The following menu is displayed:

```
File Area #2: From Altera
A   =   AREA-CHANGE,      L  =  Locate,
F   =   File-Directory,   T  =  Type a .TXT file,
G   =   Goodbye,          U  =  Upload-to-Altera,
D   =   Download-to-You,  M  =  Main-menu

File: A L F T G U D M (? for help)
>
```

Type:

**d <Enter>**

to display the following message:

```
A)scii, K)ermit, X)modem, B)atch, T)elink, ? for help
add C for CRC, i.e.. TC, etc:_
```

Type:

**? <Enter>**

to display the help screen shown here:

| Uploading a file to Altera | Downloading a file from Altera |
|---|---|
| 1. If using XMODEM type "X"<br>If using Crosstalk type "X"<br>If using MODEM7 type "B"<br>If using TELINK type "T"<br>If using MINITEL type "T"<br>If using KERMIT type "K"<br>For an ASCII transfer type "A"<br>2. and press <Enter> key.<br><br>3. Type name of file you are sending<br>4. and press <Enter> key.<br><br>5. If using Crosstalk press Attention key, type "XXMODEM", type the name of the file you are sending, and press the <Enter> key.<br><br>If not using Crosstalk, type your terminal program's "send file" command. | 1. If using XMODEM type "X"<br>If using Crosstalk type "X"<br>If using MODEM7 type "B"<br>If using TELINK type "T"<br>If using MINITEL type "T"<br>If using KERMIT type "K"<br>For an ASCII transfer type "A"<br>2. and press <Enter> key.<br><br>3. Type name of file you are getting<br>4. and press <Enter> key.<br><br>5. If using Crosstalk press Attention key, type "RXMODEM", type the name of the file you are getting, and press the <Enter> key.<br><br>If not using Crosstalk, type your terminal program's "receive file" command. |

Select the desired file-transfer protocol by typing the appropriate letter and pressing **<Enter>**. For example, to select the XMODEM protocol for Crosstalk, type:

**x <Enter>**

The following information is displayed:

```
XMODEM transfer
Filename(s):
```

Type the name of the file to be downloaded. The following message is displayed:

```
Ready to send <filename>
38 blocks, 00:54 transfer time
Start now, or Control-C to abort
```

Fido now expects you to activate the Receive File command on the communication program. For example, in Crosstalk type the Attention character (normally **<Esc>**) followed by the command **RX <filename>**.

☞ For protocols supported by your system, refer to your own system documentation.

## 8. Logging Off

To log off, type:

**g <Enter>**

The following message is displayed:

```
Do you wish to leave a suggestion: [y,N]:
```

If you do not wish to leave a suggestion, type **N <Enter>** at the prompt. The following message is displayed:

```
Logging Altera off at 30 Sep 87 13:18:54. Hang up now.
```

Or you may just hang up the phone.

If you type **y <Enter>**, the following message is displayed:

```
Wait...
This will be message #1
From:      Your name
To:        Sysop
Subject:
```

Type the subject of your message and press **<Enter>**.

The following message appears:

```
Enter your message, blank line to end.
Words will wrap automatically
```

After you have finished typing in your message and entered a blank line to finish, the following message is displayed:

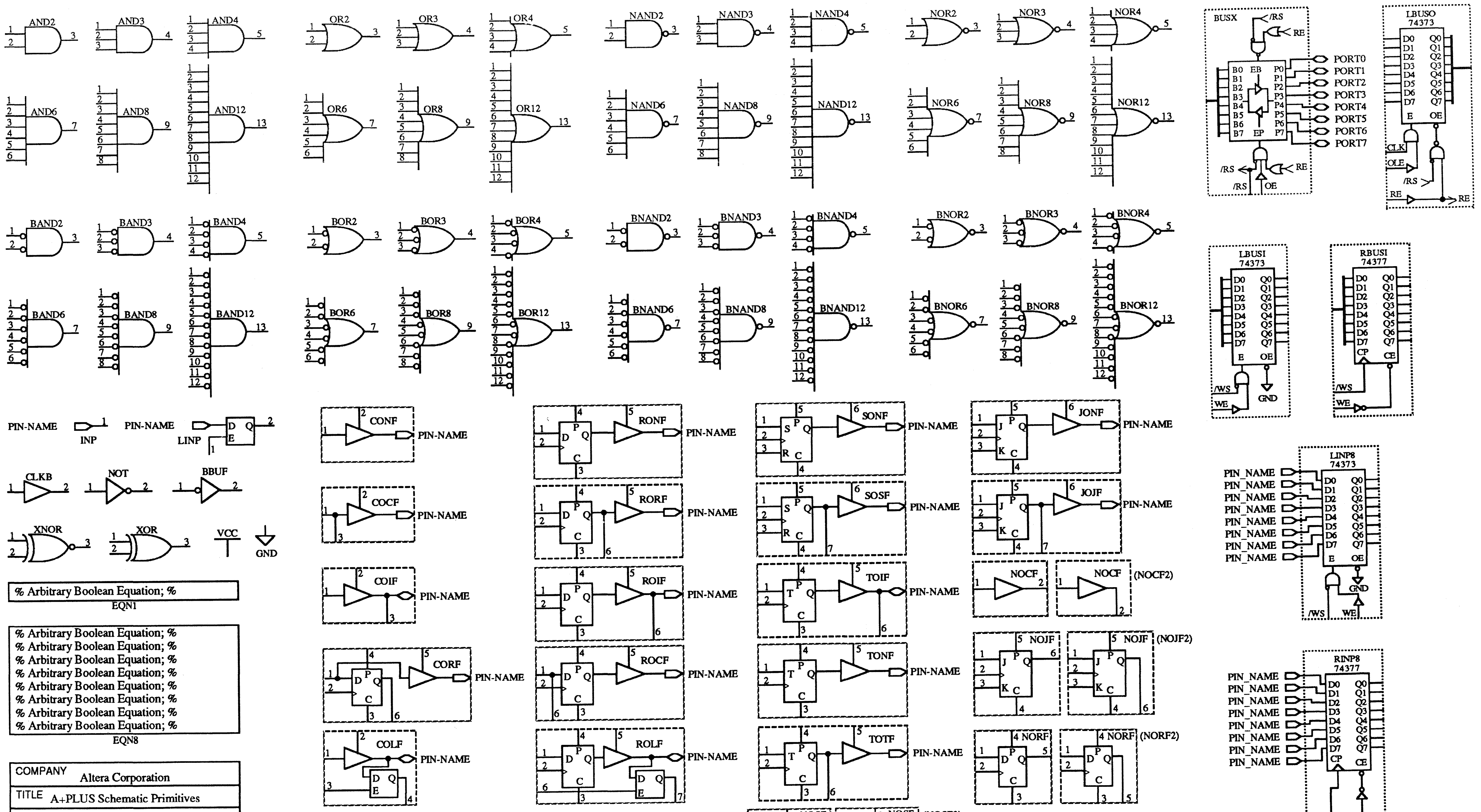
```
1:
Enter-Msg Command:
L)ist A)bort D)elete I)nsert T)o subJ)ect C)ontinue E)dit S)ave
? for help
L A D I T J C E S ?:
```

You may now edit or abort sending the message. If you press **? <Enter>**, a help screen with explanations of the message commands is displayed.

## APPENDIX F

# Altera Primitive Library

The following foldout pages may be used as a quick reference for the primitives available in the Altera Primitive Library. The first page shows a schematic drawing of each Altera primitive; the second lists the primitives available for each EPLD.



|  |  |  |  |
|---|---|---|---|
| COMPANY Altera Corporation | | | |
| TITLE A+PLUS Schematic Primitives | | | |
| DESIGNER Your Name | | | |
| SIZE B | EPLD ALL | NUMBER 1.00 | REV A |
| DATE SEPT 30, 1987 | | SHEET 1 OF 1 | |
| TURBO ON | | SECURITY OFF | |

# Altera Primitives

## Primitives Available for Altera EPLDs

| EP310 | EP320 | EP600/EP610 | EP900/EP910 | EP1210 | EP1800 | EPB1400 |
|---|---|---|---|---|---|---|
| AND(BNOR) | AND(BNOR) | AND(BNOR) | AND(BNOR) | AND(BNOR) | AND(BNOR) | AND(BNOR) |
| BBUF | BBUF | BBUF | BBUF | BBUF | BBUF | BBUF |
| COCF | COIF | CLKB | CLKB | COIF | CLKB | BUSX |
| COIF | CONF | COIF | COIF | COLF | COCF | CLKB |
| CONF | CORF | CONF | CONF | CONF | COIF | COCF |
| CORF | EQN1 | EQN1 | EQN1 | CORF | CONF | COIF |
| EQN1 | EQN8 | EQN8 | EQN8 | EQN1 | EQN1 | CONF |
| EQN8 | GND | GND | GND | EQN8 | EQN8 | EQN1 |
| GND | INP | INP | INP | GND | GND | EQN8 |
| INP | NAND(BOR) | JOJF | JOJF | INP | INP | GND |
| NAND(BOR) | NOR(BAND) | JONF | JONF | LINP | JOJF | INP |
| NOCF | NORF | NAND(BOR) | NAND(BOR) | NAND(BOR) | JONF | JOJF |
| NOR(BAND) | NOT | NOJF | NOJF | NOCF | NAND(BOR) | JONF |
| NORF | OR(BNAND) | NOR(BAND) | NOR(BAND) | NOR(BAND) | NOCF | LBUSI |
| NOT | RONF | NORF | NORF | NORF | NOJF | LBUSO |
| OR(BNAND) | RORF | NOSF | NOSF | NOSF | NOR(BAND) | LINP8 |
| ROCF | VCC | NOT | NOT | NOT | NORF | NAND(BOR) |
| ROIF | XNOR | NOTF | NOTF | OR(BNAND) | NOSF | NOCF |
| RONF | XOR | OR(BNAND) | OR(BNAND) | ROIF | NOT | NOJF |
| RORF | | ROIF | ROIF | ROLF | NOTF | NOR(BAND) |
| VCC | | RONF | RONF | RONF | OR(BNAND) | NORF |
| XNOR | | RORF | RORF | RORF | ROIF | NOSF |
| XOR | | SONF | SONF | VCC | RONF | NOT |
| | | SOSF | SOSF | XNOR | RORF | NOTF |
| | | TOIF | TOIF | XOR | SONF | OR(BNAND) |
| | | TONF | TONF | | SOSF | RBUSI |
| | | TOTF | TOTF | | TOIF | RINP8 |
| | | VCC | VCC | | TONF | ROIF |
| | | XNOR | XNOR | | TOTF | RONF |
| | | XOR | XOR | | VCC | RORF |
| | | | | | XNOR | SONF |
| | | | | | XOR | SOSF |
| | | | | | | TOIF |
| | | | | | | TONF |
| | | | | | | TOTF |
| | | | | | | VCC |
| | | | | | | XNOR |
| | | | | | | XOR |

# Glossary

**A+PLUS** Altera's Programmable Logic User System. A+PLUS is a set of computer programs and hardware support products that facilitate design and implementation of custom logic circuits with Altera EPLDs.

**ADP** See **Altera Design Processor**.

**ADP.LOG** A file output by the Altera Design Processor that records all error, information, and warning messages output by the Altera Design Processor during design processing. **ADP.LOG** is overwritten each time the ADP is invoked.

**Algorithmic State Machine (ASM) Chart** A diagrammatic representation of the output function and the next-state function in a state machine. Used as an aid in designing a state machine.

**Altera Design File (ADF)** The entry file format for the Altera Design Processor, which uses a netlist format and Boolean equations to describe the user's design. The ADF is the output of the LogiCaps schematic capture program, the State Machine Converter, and the FutureNet Pin List Converter. The ADF is converted into a JEDEC file for device programming by the ADP.

**Altera Design Processor (ADP)** The component of A+PLUS that processes an Altera Design File (ADF). It is the user interface to the A+PLUS software. It controls the individual design processing modules and converts the input design file into a JEDEC File used to program an Altera EPLD.

**Assembler** The final ADP module, which uses the output of the Fitter module to produce a JEDEC Standard File used for programming an Altera EPLD.

**Asynchronous Reset** An input which resets the register without the need for a clock signal.

**AUTOEXEC.BAT** A DOS system file that is modified during installation of A+PLUS software. (The original **AUTOEXEC.BAT** file is saved as AUTOEXEC.BAK.) This file is executed by DOS whenever the computer is booted.

**Automatic Part Selection** A design processing option that directs the ADP to select an appropriate EPLD, based on the number of inputs, outputs, and resources used in the design. It is implemented by entering **AUTO** in the Part Section of the input design file.

**Backus-Naur Form (BNF)** A notational convention used to describe the syntax of the Altera Design File and State Machine File formats (see *Appendix C*).

**Boolean Logic** Describes logic that obeys the theorems of Boolean algebra. The Boolean portion of a design is that portion which may be implemented in the AND-OR matrix of an Altera EPLD.

**Buried Register** A register in an Altera EPLD that is not associated with any pin and can be used to implement internal logic, e.g., state machines.

**Combinatorial Feedback** Feedback which is the direct function of the inputs, without regard to the clock; i.e., it does not retain values resulting from earlier inputs.

**Combinatorial Output** Output that is a direct function of the inputs, without regard to the clock; i.e., it does not retain values resulting from earlier inputs.

**CONFIG.SYS** A DOS system file that is modified during installation of A+PLUS software. (The original **CONFIG.SYS** file is saved as **CONFIG.BAK**.) DOS uses this file to set up the DOS environment.

**De Morgan's Inversion Theorem** A theorem used in Boolean algebra, which states that the complement of the product of the factors equals the sum of the complements of the addends; or that the complement of the sum of the addends equals the product of the complement of each factor. Example: **(A*B)' = A' + B'**.

**DIP** Dual In-line Package.

**Dual I/O Feedback** Feedback from the output pin on an EPLD that allows I/O pins to be used for buried registers and dedicated input.

**EOF (End-of-File)** A control character indicating that the last record in a file has been read.

**EOL (End-of-Line)** Equivalent to **<CR><LF>**.

**EPLD** Erasable Programmable Logic Device, i.e., an Altera part.

**EPLD.SYS** The A+PLUS hardware installation file is called **EPLD.SYS**. You may access this file with Item 3 on the A+PLUS Installation Menu if you wish to change the address location of the programming card or disable the color display option of LogicMap II.

**Expander** The ADP module that expands the Boolean equations in the Logic Equation File (LEF) into sum-of-products form, checks them for evidence of combinatorial feedback, simplifies the results of the expansion, and produces another LEF.

**Fitter** The ADP module that matches design requirements (and optional pin assignments) with the resources of an Altera EPLD. If the Fitter is successful, it passes the output on to the Assembler; if not, it notifies the user. In either case, it produces a Utilization Report file with the extension **.RPT**.

**Functional Simulator (FSIM)** Tests the logical operation of your EPLD design. FSIM uses specified design and part information to model the operation of the EPLD before the design is actually committed to hardware. FSIM can be run in either interactive or batch mode.

**Global Feedback** A macrocell's capability to pass its output back to all other macrocells of the EPLD.

**I/O Feedback** Feedback from the output pin on an Altera EPLD. It allows an output pin to also be used as an input pin.

**JEDEC file (.JED)** An industry-wide standard for the transfer of information between a data preparation system and a logic device programmer. The ADP converts your input design file into a JEDEC file that describes your design.

**LEF Analyzer** The ADP module which converts a binary Logic Equation File (LEF) output by the Minimizer or Expander into a human-readable file.

**LHS** Left-hand side.

Local Feedback A macrocell's capability to pass back its output to a restricted number of macrocells, rather than to all of the macrocells in the EPLD. (See **Global Feedback**.)

**Logic Array Input** A signal which is fed to the Boolean portion of a design.

**Logic Equation File (LEF)** A data structure used by several ADP modules. In the LEF, the Boolean portion of the design is separated from the non-Boolean portion to allow down-stream software to process the design according to its own specifications.

**Logic Programmer Card** The expansion card required to run A+PLUS and program an EPLD with the LogicMap II program. It occupies a single slot in the computer.

**Logic Programming Unit** The logic device programming box supplied by Altera. It contains zero-insertion force sockets into which the Altera parts are inserted. An indicator lamp on the unit shows when the unit is active.

**LogicMap II** The A+PLUS program that acts as the interface between the JEDEC file and the programming unit and allows the user to program an Altera EPLD at the bit level. The program includes an interactive data editor for manual data entry (not available for BUSTER parts). It performs timing and transfer functions for EPLD programming. Data are stored in standard JEDEC format.

**Macrocell** A basic building block in Altera EPLDs. A macrocell consists of two sections: combinatorial logic and output logic. The combinatorial logic allows a wide variety of logic functions. The output logic has two data paths: one leads to the other macrocells or feeds back to the macrocell itself; the other is configured as a pin connection acting as input, output, or bidirectional I/O port on the EPLD.

**MacroFunction** A high-level building block used in combination with existing gate and flip-flop primitives to provide a versatile design environment for EPLD logic development.

**Minimizer** The ADP module that takes the Boolean equations output by the Expander and performs logic reduction using various theorems of Boolean algebra.

**Operator** Boolean operators (*, &, +, #, /, ', !) modify data values.

**Pin** The actual pin of an EPLD, i.e., a node that is connected to an Input or I/O Primitive on one end and a pin of the EPLD on the other.

**Primitive** One of the basic functional blocks used to design circuits for Altera EPLDs. Primitives may be used in all design entry interfaces. Together, primitives make up the Altera Primitive Library, which is used exclusively when generating a design.

**Product term (p-term)** Two or more factors in a Boolean expression combined with the AND-operator constitute a product term, "product" meaning "logic product."

**Race Condition** An undesirable logic state resulting from Boolean combinations of signals which are the outputs of logic elements having finite propagation delays.

**READ.ME** File A file on the A+PLUS **INSTALL** distribution diskette. It contains information on any changes made to the A+PLUS software since the release of the A+PLUS documentation, and should be read before A+PLUS is installed.

**Register** An internal storage location.

**Registered Feedback** Feedback which is the output of a clocked storage device.

**Registered Output** The output of a clocked storage device.

**Resource** A portion of an Altera EPLD that performs a specific, user-defined task (e.g., pins, macrocells).

**RHS** Right-hand side.

**Security Bit** A feature that prevents a device from being interrogated or inadvertently reprogrammed.

**SMF** See **State Machine File.**

**Stand-Alone Microsequencer (SAM)** These Altera EPLDs (EPS444, EPS448) are supported by SAM+PLUS software.

**State Diagram** A pictorial representation of a design, where circles represent different states of the machine and the arrows between circles represent state changes. It shows the sequence of states, i.e., the current state, the previous state, the conditions for the transitions to other states, and the output for each state.

**State Machine Converter** The program that translates a State Machine File (SMF) into a standard Altera Design File (ADF) before the design is processed by the ADP.

**State Machine File (SMF)** A user-generated input design file which describes a state machine design. It contains a symbolic representation of the data for a circuit, i.e., in terms of inputs, outputs, and transitions between states. This file is converted into an ADF by the State Machine Converter prior to processing by the ADP.

**Sum-of-Products** A Boolean expression is said to be in sum-of-products form if it consists of product terms combined with the OR-operator.

**Synchronous Preset** An input which has no impact on the register until an appropriate transition of the clock is effected.

**Translator** The ADP module that converts an Altera Design File (ADF) into a Logic Equation File. It acts as initial screen for input errors by checking the validity of the requests in the ADF. In addition, it determines which EPLD is selected if automatic part selection is requested.

**Tri-State Buffer** A buffer with an input, output, and controlling signal. If the controlling signal is high (1), the output is a defined function of the input. If the controlling signal is low (0), the output is not a defined function of the input.

**Turbo-Bit** A control bit that allows the user to choose the speed and power characteristics of a device.

**Utilization Report (.RPT)** A file generated by the Fitter module indicating how EPLD resources are used by the design. This report is particularly important if the user has not specified all pin assignments. It contains header information, utilization information, and a graphical representation of pin assignments.

**wildcard characters** Provide a shorthand notation for listing filenames. The asterisk (*) represents any string of characters; the question mark (?) represents any single character.

# Index

This index covers the ***A+PLUS User Guide***, ***A+PLUS Reference Guide***, and the *State Machine Entry* and *Functional Simulator* options.

Page number prefixes refer to the following sections:

| | |
|---|---|
| ***U*** | User Guide |
| ***U***BE | Boolean Equation Entry |
| ***U***SM | State Machine Entry |
| ***U***(SM)M | State Machine Converter Messages |
| ***U***FS | Functional Simulator |
| ***U***LA | Virtual Logic Analyzer |
| ***U***(FS)M | Functional Simulator Messages |
| ***R*** | Reference Guide |
| ***R***M | A+PLUS Messages |
| ***R***A – ***R***F | Appendixes A through F |