

The
ALTERA
Advantage

LPM Quick Reference Guide

December 1996

December 1996

The **LPM Quick Reference Guide** provides information on functions in the library of parameterized modules (LPM) and on custom parameterized functions created by Altera®.

How to Contact Altera

For additional information about Altera products, consult the sources shown in [Table 1](#).

<i>Table 1. How to Contact Altera</i>			
Information Type	Access	USA & Canada	All Other Locations
Literature	Altera Express	(800) 5-ALTERA	(408) 894-7850
	Altera Literature Services	(888) 3-ALTERA, <i>Note (1)</i>	(888) 3-ALTERA; lit_req@altera.com, <i>Note (1)</i>
Non-technical customer service	Telephone hotline	(800) SOS-EPLD	(408) 894-7000
	Fax	(408) 954-8186	(408) 954-8186
Technical support	Telephone hotline (8:00 a.m. to 5 p.m. Pacific Time)	(800) 800-EPLD	(408) 894-7000, <i>Note (1)</i>
	Fax	(408) 954-0348	(408) 954-0348, <i>Note (1)</i>
	Bulletin board service	(408) 954-0104	(408) 954-0104
	Electronic mail	sos@altera.com	sos@altera.com
	FTP site	ftp.altera.com	ftp.altera.com
	CompuServe	go altera	go altera
General product information	Telephone	(408) 894-7104	(408) 894-7104, <i>Note (1)</i>
	World-wide web	http://www.altera.com	http://www.altera.com

- Note:**
- (1) You can also contact your local Altera sales office or sales representative. See “[Altera Sales Offices](#)” in this quick reference guide.

Typographic Conventions

This *LPM Quick Reference Guide* uses the typographic conventions shown in [Table 2](#).

Visual Cue	Meaning
<i>Bold italics</i>	Book titles are shown in bold italics, with initial capital letters. Example: <i>LPM Quick Reference Guide</i>
“Subheading Title”	Subheadings within a quick reference guide section and titles of MAX+PLUS II Help topics are shown in quotation marks. Example: “Altera Sales Offices”
Courier font	Function names and port names are shown in lowercase Courier. For example: <code>lpm_and, data[]</code> Parameter names are shown in uppercase Courier. For example: <code>LPM_WIDTH,</code> <code>LPM_DIRECTION</code>

December 1996

About this Quick Reference Guide.....	iii
How to Contact Altera	iii
Typographic Conventions	iv
Section 1: Introduction	1
Overview	1
History of LPM.....	1
LPM Features.....	2
LPM Functions	2
Section 2: Gate Functions.....	3
lpm_and.....	4
lpm_bustri.....	5
lpm_clshift.....	6
lpm_constant.....	7
lpm_decode.....	8
lpm_inv.....	9
lpm_mux.....	10
busmux.....	11
mux.....	12
lpm_or.....	13
lpm_xor.....	14
Section 3: Arithmetic Functions	15
lpm_abs.....	16
lpm_add_sub.....	17
lpm_compare.....	19
lpm_counter.....	21
lpm_mult	23

Section 4: Storage Functions	25
lpm_ff	26
lpm_latch	28
lpm_ram_dq	29
lpm_ram_io	31
lpm_rom	33
lpm_shiftreg	34
 Section 5: Custom Parameterized Functions	 37
csfifo	38
csdpram	39
 Section 6: Altera Sales Offices	 41
Altera Regional Offices	41
Altera International Sales Offices	42

December 1996

Overview

Digital logic designers today must create designs consisting of tens-of-thousands of gates while meeting increased pressure to shorten time-to-market. At the same time, designers must maintain architecture-independence without sacrificing silicon efficiency.

Meeting these requirements with today's EDA software tools is not easy. Schematic-based design entry provides superior efficiency but implements architecture-dependent, low-level functions. High-level hardware description languages (HDLs) offer architecture-independence, but offer reduced silicon efficiency and performance.

Because a standard set of functions supported by all EDA and integrated circuit (IC) vendors was not previously available, bridging the gap between architecture-independence and efficiency was difficult. However, with the introduction of EDA software tools that support the library of parameterized modules (LPM), designers can now create architecture-independent designs that have high silicon efficiency.

History of LPM

The LPM standard was proposed in 1990 to enable efficient mapping of digital designs to diverse architectures such as programmable logic devices (PLDs), gate arrays, and standard cells. The LPM was accepted as an Electronic Industries Association (EIA) Interim standard in April 1993 as an adjunct standard to the Electronic Design Interface Format (EDIF), an industry-standard syntax that describes a structural netlist. EDIF can be used to transfer designs between the different software tools of EDA vendors and from EDA tools to IC tools. LPM functions describe the logical operation of the netlist. LPM functions used in a design can be directly passed to the IC vendor's design implementation software through an EDIF netlist file. Before the arrival of the LPM standard, each EDIF netlist would typically contain architecture-specific logic functions, which made architecture-independent design impossible.

LPM functions are compatible with any text or graphic design entry tool, and are supported by Altera® through MAX+PLUS® II and major EDA tool vendors, including Cadence, Exemplar, Mentor Graphics, MINC, Summit Design, Synopsys, VeriBest, and Viewlogic. Altera has supported the standard since 1993, and many other silicon companies will support the LPM standard by the end of 1997.

LPM Features

The primary objective for the LPM is to enable architecture-independent design without sacrificing efficiency. The LPM meets the following key criteria:

- *Architecture-independent design entry*—Designers can work with LPM functions during design entry and verification without specifying the target architecture. Design entry and simulation tools remain architecture-independent, relying on the synthesis or fitting tools to efficiently map the design to various architectures.
- *Efficient design mapping*—The LPM allows designers to create architecture-independent designs without sacrificing efficiency. The IC vendor is responsible for the mapping of LPM functions; thus, optimum solutions are guaranteed.
- *Tool-independent design entry*—The LPM enables designers to migrate designs between EDA tools while maintaining a high-level logic description of the functions. For example, designers can use one vendor's tool for logic synthesis and another vendor's tool for logic simulation.
- *Specification of a complete design*—LPM functions completely specify the digital logic for any design. Designers can create new functions with LPM functions.

LPM Functions

The LPM presently contains 25 functions. Despite its small size, the LPM can duplicate the functionality of other design libraries that contain many more functions; each function contains parameters that allow it to expand in many dimensions. For example, the `lpm_counter` function allows the user to create counters with widths ranging from 1 to 256 bits.

In addition to width, the user can specify the features and functionality of the counter. For example, parameters indicate whether the counter counts up or down, or loads synchronously or asynchronously. Thus, the single `lpm_counter` function can replace over thirty 74-series counters.

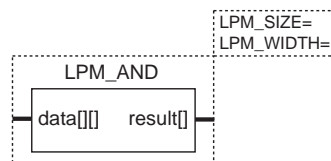


December 1996

- lpm_and.....4
- lpm_bustri.....5
- lpm_clshift.....6
- lpm_constant.....7
- lpm_decode.....8
- lpm_inv.....9
- lpm_mux.....10
- busmux.....11
- mux12
- lpm_or13
- lpm_xor.....14

lpm_and

Parameterized AND Gate



Ports

Name	Type	Required	Description
data[][]	Input	Yes	Data input to the AND gate(s). This port consists of LPM_SIZE buses, each LPM_WIDTH wide.
result[]	Output	Yes	Each result[] bit is the AND of all data[LPM_SIZE-1..0][] inputs. This port is LPM_WIDTH wide.

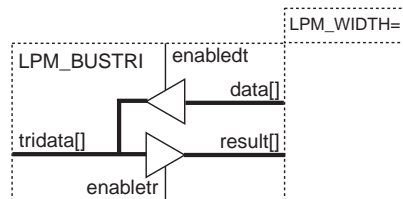
Parameters

Name	Required	Value	Description
LPM_SIZE	Yes	Integer > 0	Number of inputs to each AND gate.
LPM_WIDTH	Yes	Integer > 0	Width of the result[] port. Number of AND gates.

lpm_bustri

Parameterized Tri-State Buffer

The `lpm_bustri` function can be used to create both unidirectional and bidirectional tri-state bus controllers.



Ports

Name	Type	Required	Description
<code>data[]</code>	Input	Yes	Data input to the <code>tridata[]</code> bus. This port is <code>LPM_WIDTH</code> wide.
<code>enabletr</code>	Input	No	If high, enables <code>tridata[]</code> onto the <code>result[]</code> bus. Required if <code>result[]</code> is present (default = 0).
<code>enabledt</code>	Input	Yes	If high, enables <code>data[]</code> onto the <code>tridata[]</code> bus.
<code>result[]</code>	Output	No	This port is <code>LPM_WIDTH</code> wide.
<code>tridata[]</code>	Bidirectional	Yes	Bidirectional bus signal. This port is <code>LPM_WIDTH</code> wide.

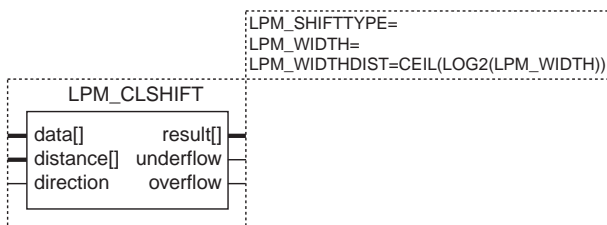
Parameters

Name	Required	Value	Description
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>data[]</code> , <code>result[]</code> , and <code>tridata[]</code> ports.

lpm_clshift

Parameterized Combinatorial Logic Shifter or Barrel Shifter

The `lpm_clshift` function performs logical, rotational, or arithmetic combinatorial shifting. The direction and distance of the shifting are user-controllable.



Ports

Name	Type	Required	Description
<code>data[]</code>	Input	Yes	Data to be shifted. This port is <code>LPM_WIDTH</code> wide.
<code>distance[]</code>	Input	Yes	Number of positions to shift <code>data[]</code> in the direction specified by the <code>direction</code> port. This port is <code>LPM_WIDTHDIST</code> wide.
<code>direction</code>	Input	No	Direction of shift. Low = left (toward the most significant bit (MSB)), high = right (toward the least significant bit (LSB)). Default value is 0 (low) = left (toward the MSB).
<code>result[]</code>	Output	Yes	Shifted data. This port is <code>LPM_WIDTH</code> wide.
<code>underflow</code>	Output	No	Logical or arithmetic underflow. If "ROTATE" is specified as the <code>LPM_SHIFTTYPE</code> parameter value and <code>overflow</code> and <code>underflow</code> are connected, the output of <code>overflow</code> and <code>underflow</code> will be undefined logic levels.
<code>overflow</code>	Output	No	Logical or arithmetic overflow. If "ROTATE" is specified as the <code>LPM_SHIFTTYPE</code> parameter value and <code>overflow</code> and <code>underflow</code> are connected, the output of <code>overflow</code> and <code>underflow</code> will be undefined logic levels.

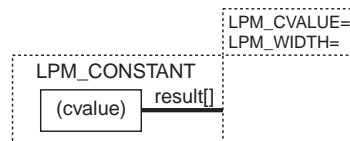
Parameters

Name	Required	Value	Description
<code>LPM_SHIFTTYPE</code>	No	"LOGICAL" "ROTATE" "ARITHMETIC"	The sign bit is extended for "ARITHMETIC" right shifts. For a "LOGICAL" right shift, 0s are always shifted into the MSB or LSB. The default value is "LOGICAL".
<code>LPM_WIDTH</code>	Yes	Integer > 1	Width of the <code>data[]</code> and <code>result[]</code> ports.
<code>LPM_WIDTHDIST</code>	Yes	Integer > 0	Width of the <code>distance[]</code> port. The <code>distance[]</code> port values normally range from 0, which is "no shift," to $(LPM_WIDTH - 1)$, which is the maximum meaningful shift. The typical value assigned to <code>LPM_WIDTHDIST</code> is "the smallest integer not less than $\log_2(LPM_WIDTH)$ " or $\text{ceil}(\log_2(LPM_WIDTH))$. Any <code>distance[]</code> port value greater than $(LPM_WIDTH - 1)$ results in an undefined logic level.

lpm_constant

Parameterized Constant Generator

The `lpm_constant` function applies a constant to a bus. This function is useful for comparisons and arithmetic functions that operate on a constant value.



Ports

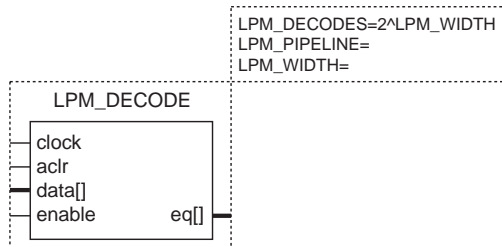
Name	Type	Required	Description
<code>result[]</code>	Output	Yes	Value specified by the argument to <code>LPM_CVALUE</code> . This port is <code>LPM_WIDTH</code> wide. The <code>LPM_CVALUE</code> parameter is truncated or zero-padded to <code>LPM_WIDTH</code> bits.

Parameters

Name	Required	Value	Description
<code>LPM_CVALUE</code>	Yes	Integer ≥ 0	Constant value to be driven out on the <code>result[]</code> port. If <code>LPM_CVALUE</code> cannot be represented in <code>LPM_WIDTH</code> bits, the <code>result[]</code> port drives the value <code>LPM_CVALUE mod 2^{LPM_WIDTH}</code> .
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>result[]</code> port.

lpm_decode

Parameterized Decoder



Ports

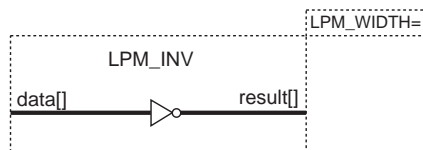
Name	Type	Required	Description
clock	Input	No	Clock for pipelined usage. The <code>clock</code> port provides pipelined operation for <code>lpm_decode</code> . For <code>LPM_PIPELINE</code> values other than 0 (default value), the <code>clock</code> port must be connected.
aclr	Input	No	Asynchronous clear for pipelined usage. The pipeline initializes to undefined. The <code>aclr</code> port can be used at any time to reset the pipeline to all 0s, asynchronously to the clock.
data[]	Input	Yes	Data input. Treated as an unsigned binary-encoded number. This port is <code>LPM_WIDTH</code> wide.
enable	Input	No	Enable. All outputs are low when this port is inactive. If absent, the default value is active (high).
eq[]	Output	Yes	Output of the decoder. This port is <code>LPM_DECODES</code> wide. If <code>data[]</code> \geq <code>LPM_DECODES</code> , all <code>eq[]</code> outputs are 0.

Parameters

Name	Required	Value	Description
LPM_DECODES	Yes	$2^{\text{LPM_WIDTH}} \geq \text{Integer} > 0$	Number of explicit decoder outputs. The default value is $2^{\text{LPM_WIDTH}}$.
LPM_PIPELINE	No	Integer ≥ 0	Specifies the number of clock cycles of latency associated with the <code>eq[]</code> output. A value of zero (0) indicates that no latency exists, and that a purely combinatorial function will be instantiated. The default value is 0 (non-pipelined).
LPM_WIDTH	Yes	Integer > 0	Width of the input value to be decoded.

lpm_inv

Parameterized Inverter



Ports

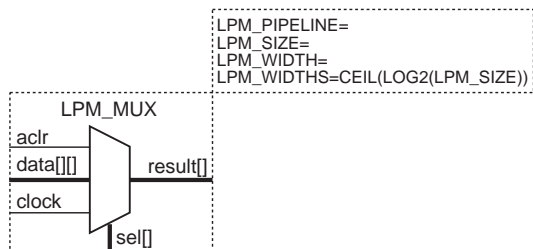
Name	Type	Required	Description
data[]	Input	Yes	Data input to the <code>lpm_inv</code> function. This port is <code>LPM_WIDTH</code> wide.
result[]	Output	Yes	Inverted result. This port is <code>LPM_WIDTH</code> wide.

Parameters

Name	Required	Value	Description
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>data[]</code> and <code>result[]</code> ports.

Ipm_mux

Parameterized Multiplexer



Ports

Name	Type	Required	Description
aclr	Input	No	Asynchronous clear for pipelined usage. The pipeline initializes to undefined. The aclr port can be used at any time to reset the pipeline to all 0s, asynchronously to the clock.
data[[]]	Input	Yes	Data input. This port consists of LPM_SIZE buses, each LPM_WIDTH wide.
clock	Input	Yes	Clock for pipelined usage. The clock port provides pipelined operation for lpm_mux. For LPM_PIPELINE values other than 0 (default value), the clock port must be connected.
sel[]	Input	Yes	Selects one of the input buses. This port is LPM_WIDTHS wide.
result[]	Output	Yes	Selected input port. This port is LPM_WIDTH wide.

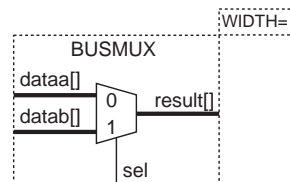
Parameters

Name	Required	Value	Description
LPM_PIPELINE	No	Integer ≥ 0	Specifies the number of clock cycles of latency associated with the result[] output. A value of zero (0) indicates that no latency exists, and that a purely combinatorial function will be instantiated. The default value is 0 (non-pipelined).
LPM_SIZE	Yes	$2^{LPM_WIDTHS} \geq \text{Integer} > 1$	Number of inputs to each multiplexer. Number of input buses.
LPM_WIDTH	Yes	Integer > 0	Width of the data[[]] and result[] ports.
LPM_WIDTHS	Yes	Integer > 0	Width of the sel[] port. The default value is $\text{ceil}(\log_2(\text{LPM_SIZE}))$.

busmux

Parameterized Multiplexer

The `busmux` function is an Altera-provided function derived from `lpm_mux` and is intended to simplify the use of `lpm_mux` in Graphic Design Files (.gdf). The `busmux` function is an instance of `lpm_mux` with `LPM_SIZE` set to 2.



Ports

Name	Type	Required	Description
dataa[]	Input	Yes	Data input to the <code>busmux</code> . This port is <code>WIDTH</code> wide.
datab[]	Input	Yes	Data input to the <code>busmux</code> . This port is <code>WIDTH</code> wide.
sel	Input	Yes	Selects one of the ports.
result[]	Output	Yes	The selected input port. This port is <code>WIDTH</code> wide.

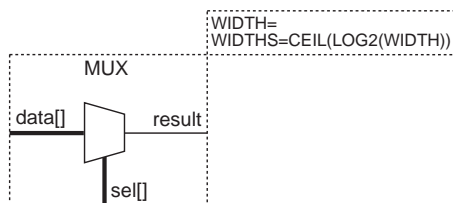
Parameters

Name	Required	Value	Description
WIDTH	Yes	Integer > 0	Width of the <code>dataa[]</code> , <code>datab[]</code> , and <code>result[]</code> ports.

mux

Parameterized Multiplexer

The `mux` function is an Altera-provided function derived from `lpm_mux` and is intended to simplify the use of `lpm_mux` in GDFs. The `mux` function is an instance of `lpm_mux` with `LPM_WIDTH` set to 1.



Ports

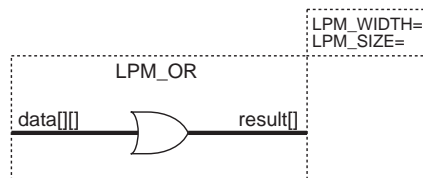
Name	Type	Required	Description
<code>data[]</code>	Input	Yes	Data input to the <code>mux</code> . This port is <code>WIDTH</code> wide.
<code>sel[]</code>	Input	Yes	Selects one of the ports. This port is <code>WIDTHS</code> wide.
<code>result</code>	Output	Yes	The selected input port. This port is 1 bit wide.

Parameters

Name	Required	Value	Description
<code>WIDTH</code>	Yes	Integer > 0	Width of the <code>data[]</code> port.
<code>WIDTHS</code>	Yes	Integer > 0	Width of the <code>sel[]</code> port. The default value is $\text{ceil}(\log_2(\text{WIDTH}))$.

lpm_or

Parameterized OR Gate



Ports

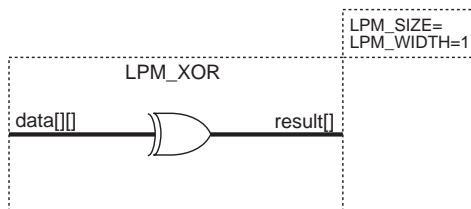
Name	Type	Required	Description
data[][]	Input	Yes	Data input to the OR gate(s). This port consists of <code>LPM_SIZE</code> buses, each <code>LPM_WIDTH</code> wide.
result[]	Output	Yes	Result of OR operators. This port is <code>LPM_WIDTH</code> wide.

Parameters

Name	Required	Value	Description
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>data[][]</code> and <code>result[]</code> ports. Number of OR gates.
<code>LPM_SIZE</code>	Yes	Integer > 0	Number of inputs to each OR gate. Number of input buses.

lpm_xor

Parameterized XOR Gate



Ports

Name	Type	Required	Description
<code>data[[]]</code>	Input	Yes	Data input to the XOR gate(s). This port consists of <code>LPM_SIZE</code> buses, each <code>LPM_WIDTH</code> wide.
<code>result[]</code>	Output	Yes	Each <code>result[]</code> bit is the XOR of <code>LPM_SIZE</code> bits. This port is <code>LPM_WIDTH</code> wide.

Parameters

Name	Required	Value	Description
<code>LPM_SIZE</code>	Yes	Integer > 0	Number of inputs to each XOR gate. Number of input buses.
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>data[[]]</code> and <code>result[]</code> ports. Number of XOR gates.

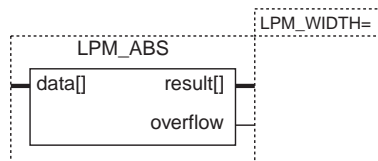


December 1996

lpm_abs.....	16
lpm_add_sub.....	17
lpm_compare.....	19
lpm_counter.....	21
lpm_mult.....	23

lpm_abs

Parameterized Absolute Value



Ports

Name	Type	Required	Description
data[]	Input	Yes	Signed number. This port is LPM_WIDTH wide.
result[]	Output	Yes	Absolute value of data[]. This port is LPM_WIDTH wide.
overflow	Output	No	High if $data[] = -2^{(LPM_WIDTH - 1)}$. Two's complement allows one more negative number than positive. The <code>overflow</code> port detects that singular instance and goes high to indicate that no positive equivalent exists.

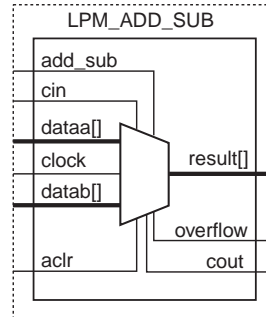
Parameters

Name	Required	Value	Description
LPM_WIDTH	Yes	Integer ≥ 0	Width of the data[] and result[] ports.

lpm_add_sub

Parameterized Adder/Subtractor

LPM_DIRECTION=
LPM_PIPELINE=
LPM_REPRESENTATION=
LPM_WIDTH=
ONE_INPUT_IS_CONSTANT=



Ports

Name	Type	Required	Description
add_sub	Input	No	If the input is high, the operation = $dataa[] + datab[]$. If the input is low, the operation = $dataa[] - datab[]$. This port cannot be used if the LPM_DIRECTION parameter is used. If both the add_sub port and the LPM_DIRECTION parameter are omitted, the operation defaults to "ADD".
cin	Input	No	Carry-in to the low-order bit. If the operation is "ADD", Low = 0 and High = +1. If the operation is "SUB", Low = -1 and High = 0. If omitted, the default is 0 (i.e., no affect on "ADD" or "SUB" operations).
dataa[]	Input	Yes	Addend/Minuend. This port is LPM_WIDTH wide.
clock	Input	No	Clock for pipelined usage. The clock port provides pipelined operation of lpm_add_sub. For LPM_PIPELINE values other than 0 (default value), the clock port must be connected.
datab[]	Input	Yes	Addend/Subtrahend. This port is LPM_WIDTH wide.
aclr	Input	No	Asynchronous clear for pipelined usage. The pipeline initializes to undefined. The aclr port can be used at any time to reset the pipeline to all 0s, asynchronously to clock.
result[]	Output	Yes	$dataa[] + or - datab[] + or - cin$. This port is LPM_WIDTH wide.
overflow	Output	No	Result exceeds available precision. If overflow is used, cout cannot be used. The overflow signal has a physical interpretation as the XOR of the carry into the MSB with the carry out of the MSB. The overflow signal is only meaningful when the LPM_REPRESENTATION parameter is set to "SIGNED". <i>Note (1)</i>
cout	Output	No	Carry-out (borrow-in) of the MSB. If overflow is used, cout cannot be used. The cout signal has a physical interpretation as the carry-out (borrow-in) of the MSB and is most meaningful for detecting overflow in "UNSIGNED" operations. <i>Note (2)</i>

Parameters

Name	Required	Value	Description
LPM_DIRECTION	No	"ADD" "SUB" "DEFAULT"	The add_sub port cannot be used if the LPM_DIRECTION parameter is has a value other than "DEFAULT". The default value is "DEFAULT".
LPM_PIPELINE	No	Integer ≥ 0	Specifies the number of clock cycles of latency associated with the result[] output. A value of zero (0) indicates that no latency exists, and that a purely combinatorial function will be instantiated. The default value is 0 (non-pipelined).
LPM_REPRESENTATION	No	"SIGNED" "UNSIGNED"	Type of addition performed. The default value is "SIGNED".
LPM_WIDTH	Yes	Integer > 0	Width of the dataa[], datab[], and result[] ports.
ONE_INPUT_IS_CONSTANT	No	"YES" "NO"	Provides greater optimization if an input is constant. The default value is "NO".

Notes:

- (1) The following table describes the overflow port during "ADD" and "SUB" operations.

Value	ADD Operation	SUB Operation
"UNSIGNED"	Not meaningful.	Not meaningful.
"SIGNED"	$(dataa + datab + cin) > 2^{(LPM_WIDTH - 1) - 1}$ or $(dataa + datab + cin) > 2^{(LPM_WIDTH - 1)}$	$(dataa - datab - cin) > 2^{(LPM_WIDTH - 1) - 1}$ or $(dataa - datab - cin) > 2^{(LPM_WIDTH - 1)}$

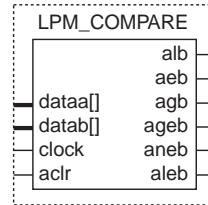
- (2) The following table describes the cout port during "ADD" and "SUB" operations.

Value	ADD Operation	SUB Operation
"UNSIGNED"	$dataa + datab + cin > 2^{(LPM_WIDTH - 1) - 1}$	Normal subtract. However, if cout = 0, then $(dataa - datab - cin) < 0$.
"SIGNED"	Normal result of adding two negative numbers, or possible overflow.	Normal result when subtracting a positive number from a negative number, or possible overflow.

lpm_compare

Parameterized Comparator

CHAIN_SIZE=
LPM_PIPELINE=
LPM_REPRESENTATION=
LPM_WIDTH=
ONE_INPUT_IS_CONSTANT=



Ports

Name	Type	Required	Description
dataa[]	Input	Yes	The datab[] signal is compared to this value. This port is LPM_WIDTH wide.
datab[]	Input	Yes	Value to be compared to dataa[]. This port is LPM_WIDTH wide.
clock	Input	No	Clock for pipelined usage. The clock port provides pipelined operation for lpm_compare. For LPM_PIPELINE values other than 0 (default value), the clock port must be connected.
aclr	Input	No	Asynchronous clear for pipelined usage. The pipeline initializes to undefined. The aclr port can be used at any time to reset the pipeline to all 0s, asynchronously to clock.
alb	Output	No	High (1) if dataa[] < datab[]. One of the alb, aeb, agb, ageb, aleb, or aneb outputs must be present.
aeb	Output	No	High (1) if dataa[] = datab[]. One of the alb, aeb, agb, ageb, aleb, or aneb outputs must be present.
agb	Output	No	High (1) if dataa[] > datab[]. One of the alb, aeb, agb, ageb, aleb, or aneb outputs must be present.
ageb	Output	No	High (1) if dataa[] ≥ datab[]. One of the alb, aeb, agb, ageb, aleb, or aneb outputs must be present.
aneb	Output	No	High (1) if dataa[] ≠ datab[]. One of the alb, aeb, agb, ageb, aleb, or aneb outputs must be present.
aleb	Output	No	High (1) if dataa[] ≤ datab[]. One of the alb, aeb, agb, ageb, aleb, or aneb outputs must be present.

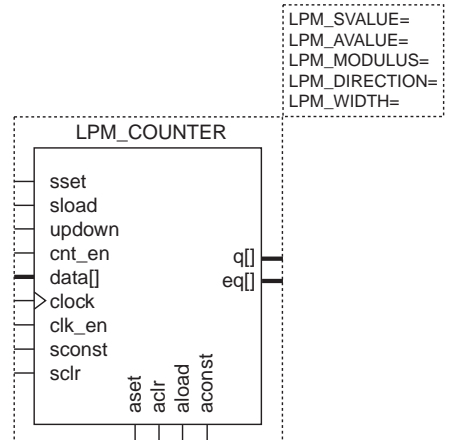
Parameters

Name	Required	Value	Description
CHAIN_SIZE	No	Integer > 0	Maximum allowable length of carry chains or cascade chains in FLEX 10K and FLEX 8000 devices. The default value is 8. In MAX+PLUS II, this value overrides the value of the <i>Max. Auto Length</i> option for the Carry Chain or Cascade Chain logic option(s) in the global project logic synthesis style, which is specified with the Global Project Logic Synthesis dialog box (Assign menu). For other device families, varying the CHAIN_SIZE parameter will provide different speed/size combinations—setting smaller values for CHAIN_SIZE generally results in faster and larger comparators and vice versa. For more information, contact Altera Applications.
LPM_PIPELINE	No	Integer ≥ 0	Specifies the number of clock cycles of latency associated with the al _b , ae _b , ag _b , age _b , ane _b , and ale _b outputs. A value of zero (0) indicates that no latency exists, and that a purely combinatorial function will be instantiated. Default value is 0 (non-pipelined).
LPM_REPRESENTATION	No	"SIGNED" "UNSIGNED"	Type of comparison performed. The default value is "UNSIGNED".
LPM_WIDTH	Yes	Integer > 0	Width of the data _a [] and data _b [] ports.
ONE_INPUT_IS_CONSTANT	No	"YES" "NO"	Provides greater optimization if an input is constant. The default value is "NO".

lpm_counter

Parameterized Counter

The `lpm_counter` function is a full-featured counter with loading, up/down control, clock, and count enabling and clearing.



Ports (Part 1 of 2)

Name	Type	Required	Description
<code>sset</code>	Input	No	Synchronous set input. Sets the counter on the next active clock edge. Default = 0. Sets <code>q[]</code> outputs to all 1s, or to the value specified by <code>LPM_SVALUE</code> . If both <code>sset</code> and <code>sclr</code> are used and both are asserted, <code>sclr</code> is dominant. For outputs such as <code>q[]</code> and <code>eq[]</code> , <code>sset</code> affects the output before polarity is applied.
<code>sload</code>	Input	No	Synchronous load input. Loads the counter with data on the next active clock edge. Default = 0. If <code>sload</code> is used, <code>data[]</code> must be connected.
<code>updown</code>	Input	No	Controls the direction of the count. High (1) = count up. Low (0) = count down. Default = up (1). If the <code>LPM_DIRECTION</code> parameter is used, the <code>updown</code> port cannot be connected. If <code>LPM_DIRECTION</code> is not used, the <code>updown</code> port is optional.
<code>cnt_en</code>	Input	No	Count enable input. Disables count when low (0) without affecting <code>sload</code> , <code>sset</code> , or <code>sclr</code> . Default = 1.
<code>data[]</code>	Input	No	Parallel data input to the counter. This port is <code>LPM_WIDTH</code> wide. Uses <code>aload</code> and/or <code>sload</code> .
<code>clock</code>	Input	Yes	Positive-edge-triggered clock.
<code>clk_en</code>	Input	No	Clock enable input. Enables all synchronous activities. Default = 1.
<code>sconst</code>	Input	No	This port is provided only for backwards-compatibility in MAX+PLUS II pre-version 6.0 designs. Altera does not recommend using this port for new designs.
<code>sclr</code>	Input	No	Synchronous clear input. Clears the counter on the next active clock edge. Default = 0. If both <code>sset</code> and <code>sclr</code> are used and both are asserted, <code>sclr</code> is dominant. For outputs such as <code>q[]</code> and <code>eq[]</code> , <code>sclr</code> affects the output before polarity is applied.

Ports (Part 2 of 2)

Name	Type	Required	Description
aset	Input	No	Asynchronous set input. Default = 0. Sets $q[]$ outputs to all 1s, or to the value specified by LPM_AVALUE. If both <i>aset</i> and <i>aclr</i> are used and both are asserted, <i>aclr</i> is dominant. For outputs such as $q[]$ and $eq[]$, <i>aset</i> affects the output before polarity is applied.
aclr	Input	No	Asynchronous clear input. Default = 0. If both <i>aset</i> and <i>aclr</i> are used and both are asserted, <i>aclr</i> is dominant. For outputs such as $q[]$ and $eq[]$, <i>aclr</i> affects the output before polarity is applied.
aload	Input	No	Asynchronous load input. Asynchronously loads the counter with the value on the data input. Default = 0. If <i>aload</i> is used, <i>data[]</i> must be used.
aconst	Input	No	This port is provided only for backwards-compatibility in MAX+PLUS II pre-version 6.0 designs. Altera does not recommend using this port in new designs.
$eq[]$	Output	No	Counter decode output. Active high when the counter reaches the specified count value. Either the $q[]$ port or $eq[]$ port must be connected. Up to <i>c</i> eq ports can be used ($c \leq 15$). Only the 16 lowest count values are decoded. When the count value is <i>c</i> , the eqc output is set high (1). For example, when the count is 0, $eq0 = 1$, when the count is 1, $eq1 = 1$, and when the count is 15, $eq15 = 1$. Decoded outputs for count values of 16 or greater require external decoding. The eqc outputs are asynchronous.
$q[]$	Output	No	Data output from the counter. This port is LPM_WIDTH wide. Either $q[]$ or at least one of the $eq[]$ ports must be connected.

Parameters

Name	Required	Value	Description
LPM_SVALUE	No	Integer ≥ 0	Constant value that is loaded on the rising edge of clock when <i>sset</i> is high. Must be used if <i>sset</i> is used.
LPM_AVALUE	No	Integer ≥ 0	Constant value that is loaded when <i>aset</i> is high. This parameter must be used if <i>aset</i> is used. If the value specified is larger than the <i><modulus></i> , the behavior of the counter is an undefined logic level. The <i><modulus></i> is LPM_MODULUS, if present, or 2^{LPM_WIDTH} .
LPM_MODULUS	No	Integer > 1	The maximum count, plus one. Number of unique states in the counter's cycle. If the load value is larger than the LPM_MODULUS parameter, the behavior of the counter is not specified. The default value is 2^{LPM_WIDTH} .
LPM_DIRECTION	No	"UP" "DOWN" "DEFAULT"	If the LPM_DIRECTION parameter is used, the <i>updown</i> port cannot be connected. When the <i>updown</i> port is not connected, the count direction is "UP". In all other cases, the default value is "DEFAULT".
LPM_WIDTH	Yes	Integer > 0	Width of the input and output ports. If no output ports are specified, the value is the number of bits in the count.

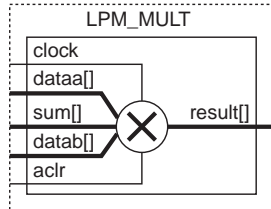
lpm_mult

Parameterized Multiplier

The `lpm_mult` function allows two signed or unsigned numbers to be multiplied. In addition, the result of the multiplication can be added to a third number.

```

INPUT_A_IS_CONSTANT=
INPUT_B_IS_CONSTANT=
LPM_PIPELINE=
LPM_REPRESENTATION=
LPM_WIDTHA=
LPM_WIDTHB=
LPM_WIDTHHP=(LPM_WIDTHA+LPM_WIDTHB)
LPM_WIDTHS=LPM_WIDTHA
  
```



Ports

Name	Type	Required	Description
<code>clock</code>	Input	No	Clock for pipelined usage. The <code>clock</code> port provides pipelined operation for <code>lpm_mult</code> . For <code>LPM_PIPELINE</code> values other than 0 (default value), the <code>clock</code> port must be connected.
<code>dataa[]</code>	Input	Yes	Multiplicand. This port is <code>LPM_WIDTHA</code> wide.
<code>sum[]</code>	Input	No	Partial sum. This port is <code>LPM_WIDTHS</code> wide.
<code>datab[]</code>	Input	Yes	Multiplier. This port is <code>LPM_WIDTHB</code> wide.
<code>aclr</code>	Input	No	Asynchronous clear for pipelined usage. The pipeline initializes to undefined. The <code>aclr</code> port can be used at any time to reset the pipeline to all 0s, asynchronously to <code>clock</code> .
<code>result[]</code>	Output	Yes	This port is <code>LPM_WIDTHHP</code> wide. If $LPM_WIDTHHP < \max(LPM_WIDTHA + LPM_WIDTHB, LPM_WIDTHS)$ or (LPM_WIDTHS) , only the <code>LPM_WIDTHHP</code> MSBs are present.

Parameters

Name	Required	Value	Description
INPUT_A_IS_CONSTANT	No	"YES" "NO"	If <code>dataa[]</code> is connected to a constant value, setting the value of <code>INPUT_A_IS_CONSTANT</code> to "YES" optimizes the multiplier for resource usage and speed. The default value is "NO".
INPUT_B_IS_CONSTANT	No	"YES" "NO"	If <code>datab[]</code> is connected to a constant value, setting the value of <code>INPUT_B_IS_CONSTANT</code> to "YES" optimizes the multiplier for resource usage and speed. The default value is "NO".
LPM_PIPELINE	No	Integer ≥ 0	Specifies the number of clock cycles of latency associated with the <code>result[]</code> output. The default value of zero (0) indicates that no latency exists, and that a purely combinatorial function will be instantiated. The default value is 0 (non-pipelined).
LPM_REPRESENTATION	No	"SIGNED" "UNSIGNED"	Type of multiplication performed. The default value is "UNSIGNED".
LPM_WIDTHA	Yes	Integer > 0	Width of the <code>dataa[]</code> port.
LPM_WIDTHB	Yes	Integer > 0	Width of the <code>datab[]</code> port.
LPM_WIDTHP	Yes	Integer > 0	Width of the <code>result[]</code> port. The default is <code>LPM_WIDTHA+LPM_WIDTHB</code> .
LPM_WIDTHS	No	Integer > 0	Width of the <code>sum[]</code> port. If the <code>sum[]</code> port is not used, <code>LPM_WIDTHS</code> must be set to a value between 1 and <code>LPM_WIDTHP</code> , inclusive. The default value is <code>LPM_WIDTHA</code> .
LATENCY, <i>Note (1)</i>	No	Integer ≥ 0	Same as <code>LPM_PIPELINE</code> .

Note:

- (1) The `LATENCY` parameter is provided only for backwards-compatibility with MAX+PLUS II pre-version 7.0 designs. For all new designs, you should use the `LPM_PIPELINE` parameter instead.



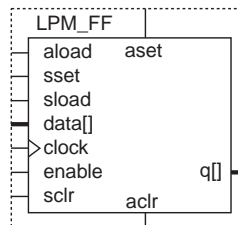
December 1996

lpm_ff.....	26
lpm_latch.....	28
lpm_ram_dq.....	29
lpm_ram_io.....	31
lpm_rom.....	33
lpm_shiftreg.....	34

Ipm_ff

Parameterized D or T Flipflop

LPM_AVALUE=
LPM_FFTYPE=
LPM_SVALUE=
LPM_WIDTH=



Ports

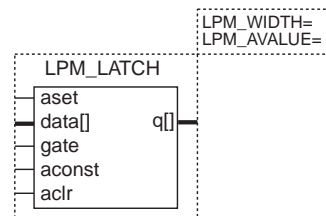
Name	Type	Required	Description
aload	Input	No	Asynchronous load input. Asynchronously loads the flipflop with the value on the data[] input. Default = 0. If aload is used, data[] must be used.
sset	Input	No	Synchronous set input. Sets the q[] outputs to the value specified by LPM_SVALUE, if that value is present, or sets the q[] outputs to all 1s. If both sset and sclr are used and both are asserted, sclr is dominant. The sset input affects the output q[] values before polarity is applied to the ports.
sload	Input	No	Synchronous load input. Loads the flipflop with the value on the data[] input on the next active clock edge. Default = 0. If sload is used, data[] must be used. For load operation, sload must be high (1) and enable must be high (1) or unconnected.
data[]	Input	No	T flipflop: toggle enable; D flipflop: data input. This port is LPM_WIDTH wide. If the data[] port is not used, at least one of the aset, aclr, sset, or sclr ports must be used.
clock	Input	Yes	Positive-edge-triggered clock.
enable	Input	No	Clock enable input. Default = 1.
sclr	Input	No	Synchronous clear input. If both sset and sclr are used and both are asserted, sclr is dominant. The sclr input affects the output q[] values before polarity is applied to the ports.
aset	Input	No	Asynchronous set input. Sets q[] outputs to the value specified by LPM_AVALUE, if that value is present, or sets the q[] outputs to all 1s.
aclr	Input	No	Asynchronous clear input. If both aset and aclr are used and both are asserted, aclr is dominant. The aclr input affects the output q[] values before polarity is applied to the ports.
q[]	Output	Yes	Data output from D flipflops. This port is LPM_WIDTH wide.

Parameters

Name	Required	Value	Description
LPM_AVALUE	Yes	Integer ≥ 0	Constant value that is loaded when <code>aset</code> is high. The default value is all 1s.
LPM_FFTYPE	No	"DFF" "TFF"	Type of flipflop. The default value is "DFF".
LPM_SVALUE	No	Integer ≥ 0	Constant value that is loaded on the rising edge of <code>clock</code> when <code>sset</code> is high. The default value is all 1s.
LPM_WIDTH	Yes	Integer > 0	Width of the <code>data[]</code> and <code>q[]</code> ports.

lpm_latch

Parameterized Latch



Ports

Name	Type	Required	Description
aset	Input	No	Asynchronous set input. Default = 0. Sets <code>q[]</code> outputs to the value specified by <code>LPM_AVALUE</code> , if that value is present. If no <code>LPM_AVALUE</code> is specified, <code>aset</code> will set the count to all 1s. If both <code>aset</code> and <code>aclr</code> are used and both are asserted, <code>aclr</code> is dominant. The <code>aset</code> and <code>aclr</code> inputs affect the output <code>q[]</code> values before polarity is applied to the ports.
data[]	Input	No	Data input to the D-type latch. This port is <code>LPM_WIDTH</code> wide. If the <code>data[]</code> port is not used, either <code>aset</code> or <code>aclr</code> must be used.
gate	Input	Yes	Latch enable input. High = flow-through, low = latch.
aconst	Input	No	This port is provided only for backwards-compatibility in MAX+PLUS II pre-version 6.0 designs. Altera does not recommend using this port in new designs.
aclr	Input	No	Asynchronous clear input. Default = 0. Sets the latch to all 0s. If both <code>aset</code> and <code>aclr</code> are used and both are asserted, <code>aclr</code> is dominant. The <code>aset</code> and <code>aclr</code> inputs affect the output <code>q[]</code> values before the polarity is applied to the ports.
q[]	Output	Yes	Data output from the latches. This port is <code>LPM_WIDTH</code> wide.

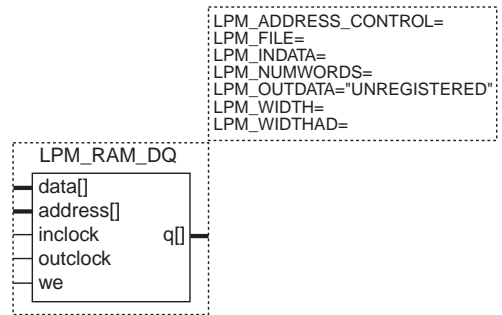
Parameters

Name	Required	Value	Description
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>data[]</code> and <code>q[]</code> ports.
<code>LPM_AVALUE</code>	No	Integer ≥ 0	Constant value that is loaded when <code>aset</code> is high. The default value of <code>LPM_AVALUE</code> is all 1s.

lpm_ram_dq

Parameterized Random Access Memory with Separate Input and Output Ports

The `lpm_ram_dq` function can be used as either synchronous or asynchronous random access memory. The `lpm_ram_dq` function has separate input and output data buses.



Ports

Name	Type	Required	Description
<code>data[]</code>	Input	Yes	Data input to memory. This port is <code>LPM_WIDTH</code> wide.
<code>address[]</code>	Input	Yes	Address input to the memory. This port is <code>LPM_WIDTHHAD</code> wide.
<code>inclock</code>	Input	No	Synchronizes memory loading. If the <code>inclock</code> port is used, the <code>we</code> port acts as an enable for write operations synchronized to the rising edge of the <code>inclock</code> input. If the <code>inclock</code> port is not used, the <code>we</code> port acts as an enable for asynchronous write operations.
<code>outclock</code>	Input	No	Synchronizes <code>q[]</code> outputs from memory. The addressed memory content → <code>q[]</code> response is synchronous when the <code>outclock</code> port is connected, and asynchronous when it is not connected.
<code>we</code>	Input	Yes	Write enable input. When high, enables write operation to the memory. Required if <code>inclock</code> is not present. If only <code>we</code> is used, the data on the <code>address[]</code> port should not change while <code>we</code> is high. If the data on the <code>address[]</code> port changes while <code>we</code> is high, all memory locations that are addressed are overwritten with <code>data[]</code> .
<code>q[]</code>	Output	Yes	Data output from the memory. This port is <code>LPM_WIDTH</code> wide.

Storage Functions

Parameters

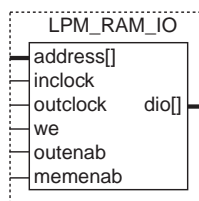
Name	Required	Value	Description
LPM_ADDRESS_CONTROL	No	"REGISTERED" "UNREGISTERED"	Controls whether the <code>address[]</code> and <code>we</code> ports are registered. The default value is "REGISTERED".
LPM_FILE	No	"<filename>"	Name of the Memory Initialization File (.mif) or Hexadecimal File (.hex) containing RAM initialization data ("<filename>"). If omitted, contents default to all 0s.
LPM_INDATA	No	"REGISTERED" "UNREGISTERED"	Controls whether the <code>data[]</code> port is registered. The default value is "REGISTERED".
LPM_NUMWORDS	No	Integer > 0	Number of words stored in memory. In general, this value should be (but is not required to be): $2^{LPM_WIDTHAD - 1} < LPM_NUMWORDS \leq 2^{LPM_WIDTHAD}$. The default value is $2^{LPM_WIDTHAD}$.
LPM_OUTDATA	No	"REGISTERED" "UNREGISTERED"	Controls whether the <code>q[]</code> and internal <code>eq</code> ports are registered. The default value is "REGISTERED".
LPM_WIDTH	Yes	Integer > 0	Width of the <code>data[]</code> and <code>q[]</code> ports.
LPM_WIDTHAD	Yes	Integer > 0	Width of the <code>address[]</code> port. LPM_WIDTHAD should be (but is not required to be): $\log_2(LPM_NUMWORDS)$. If LPM_WIDTHAD is too small, some memory locations will not be addressable. If LPM_WIDTHAD is too large, the addresses that are too high will return undefined logic levels.

lpm_ram_io

Parameterized Random Access Memory with a Single I/O Port

The `lpm_ram_io` function can be used as either synchronous or asynchronous random access memory. The `lpm_ram_io` function has a bidirectional bus.

```
LPM_ADDRESS_CONTROL=
LPM_FILE=
LPM_INDATA=
LPM_NUMWORDS=
LPM_OUTDATA="UNREGISTERED"
LPM_WIDTH=
LPM_WIDTHHAD=
```



Ports

Name	Type	Required	Description
<code>address[]</code>	Input	Yes	Address input to the memory. This port is <code>LPM_WIDTHHAD</code> wide. If <code>memenab</code> is used, it should be inactive when <code>address[]</code> is changing.
<code>inclock</code>	Input	No	Synchronizes memory loading. If the <code>inclock</code> port is used, the <code>we</code> port acts as an enable for write operations synchronized to the rising edge of the <code>inclock</code> input. If the <code>inclock</code> port is not used, the <code>we</code> port acts as an enable for asynchronous write operations.
<code>outclock</code>	Input	No	Synchronizes <code>dio[]</code> from memory. The addressed memory content $\rightarrow q[]$ response is synchronous when the <code>outclock</code> port is connected, and asynchronous when it is not connected.
<code>we</code>	Input	Yes	Write enable input. Either <code>we</code> or <code>outenab</code> should be used. When high, enables write operations to the memory. If no clock ports are used, the data on the <code>address[]</code> port should not change when <code>we</code> is high (1). Required if <code>clock</code> is not present. If <code>we</code> is absent, the default value is enabled.
<code>outenab</code>	Input	No	Output enable input. High (1): <code>dio[]</code> from memory <code>address[]</code> . Low (0): Memory <code>address[]</code> from <code>dio[]</code> . Either <code>memenab</code> or <code>outenab</code> must be present.
<code>memenab</code>	Input	No	Memory output tri-state enable. Either <code>memenab</code> or <code>outenab</code> must be connected. If <code>memenab</code> is present, it should be inactive when <code>address[]</code> is changing.
<code>dio[]</code>	Bidirectional	Yes	Data port for the memory. This port is <code>LPM_WIDTH</code> wide.

Storage Functions

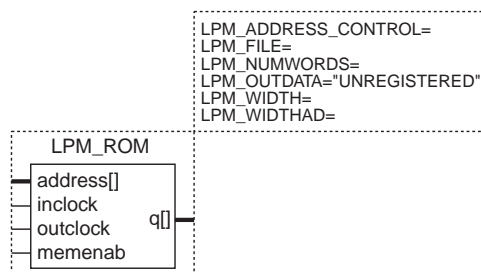
Parameters

Name	Required	Value	Description
LPM_ADDRESS_CONTROL	No	"REGISTERED" "UNREGISTERED"	Controls whether the address[], memenab, and we ports are registered. The default value is "REGISTERED".
LPM_FILE	No	"<filename>"	Name of the MIF or Hex File containing ROM initialization data ("<filename>"). If omitted, contents default to all 0s.
LPM_INDATA	No	"REGISTERED" "UNREGISTERED"	Controls whether the internal data port is registered. The default value is "REGISTERED".
LPM_NUMWORDS	No	Integer > 0	Number of words stored in memory. In general, this value should be (but is not required to be): $2^{LPM_WIDTHAD - 1} < LPM_NUMWORDS \leq 2^{LPM_WIDTHAD}$. The default value is $2^{LPM_WIDTHAD}$.
LPM_OUTDATA	No	"REGISTERED" "UNREGISTERED"	Controls whether the dio[] port is registered. The default value is "REGISTERED".
LPM_WIDTH	Yes	Integer > 0	Width of dio[] and internal data and q ports.
LPM_WIDTHAD	Yes	Integer > 0	Width of the address[] port. LPM_WIDTHAD should be (but is not required to be) equal to $\log_2(LPM_NUMWORDS)$. If LPM_WIDTHAD is too small, some memory locations will not be addressable. If it is too large, the addresses that are too high will return undefined logic levels.

lpm_rom

Parameterized Read-Only Memory

The `lpm_rom` function can be used as either synchronous or asynchronous read-only memory.



Ports

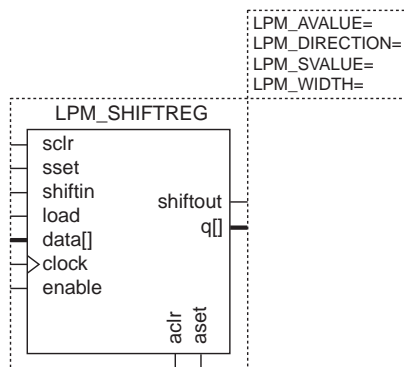
Name	Type	Required	Description
address[]	Input	Yes	Address input to the memory. This port is <code>LPM_WIDTHHAD</code> wide.
inclock	Input	No	Clock for input registers. The <code>address[]</code> port is synchronous (registered) when the <code>inclock</code> port is connected, and is asynchronous (unregistered) when the <code>inclock</code> port is not connected.
outclock	Input	No	Clock for output registers. The addressed memory content \rightarrow <code>q[]</code> response is synchronous when the <code>outclock</code> port is connected, and is asynchronous when it is not connected.
memenab	Input	No	Memory enable input. High = data output on <code>q[]</code> , Low = high-impedance outputs.
q[]	Output	Yes	Memory output. This port is <code>LPM_WIDTH</code> wide.

Parameters

Name	Required	Value	Description
<code>LPM_ADDRESS_CONTROL</code>	No	"REGISTERED" "UNREGISTERED"	Controls whether the <code>address[]</code> port is registered. The default value is "REGISTERED".
<code>LPM_FILE</code>	Yes	"<filename>"	Name of the MIF or Hex File containing ROM initialization data ("<filename>").
<code>LPM_NUMWORDS</code>	No	Integer > 0	In general, this value should be (but is not required to be) $2^{LPM_WIDTHHAD-1} < LPM_NUMWORDS \leq 2^{LPM_WIDTHHAD}$. The default value is $2^{LPM_WIDTHHAD}$.
<code>LPM_OUTDATA</code>	No	"REGISTERED" "UNREGISTERED"	Controls whether the <code>q[]</code> port is registered. The default value is "REGISTERED".
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>q[]</code> port.
<code>LPM_WIDTHHAD</code>	Yes	Integer > 0	Width of the <code>address[]</code> port. <code>LPM_WIDTHHAD</code> should be (but is not required to be) equal to $\log_2(LPM_NUMWORDS)$. If <code>LPM_WIDTHHAD</code> is too small, some memory locations will not be addressable. If it is too large, the addresses that are too high will return undefined logic levels.

lpm_shiftreg

Parameterized Shift Register



Ports

Name	Type	Required	Description
sclr	Input	No	Synchronous clear input. If both <code>sset</code> and <code>sclr</code> are used and both are asserted, <code>sclr</code> is dominant. The <code>sclr</code> input affects the output <code>q[]</code> values before polarity is applied to the ports.
sset	Input	No	Synchronous set input. Sets <code>q[]</code> outputs to the value specified by <code>LPM_SVALUE</code> , if that value is present, or sets the <code>q[]</code> outputs to all 1s. If both <code>sset</code> and <code>sclr</code> are used and both are asserted, <code>sclr</code> is dominant. The <code>sset</code> input affects the output <code>q[]</code> values before polarity is applied to the ports.
shiftin	Input	No	Serial shift data input. At least one of the <code>data[]</code> , <code>aset</code> , <code>aclr</code> , <code>sset</code> , <code>sclr</code> , and/or <code>shiftin</code> ports must be used.
load	Input	No	Synchronous parallel load. High (1): load operation; low (0): shift operation. Default is low (0) shift operation. For parallel load operation, <code>load</code> must be high (1) and <code>enable</code> must be high or unconnected.
data[]	Input	No	Data input to the shift register. This port is <code>LPM_WIDTH</code> wide. At least one of the <code>data[]</code> , <code>aset</code> , <code>aclr</code> , <code>sset</code> , <code>sclr</code> and/or <code>shiftin</code> ports must be used.
clock	Input	Yes	Positive-edge-triggered clock. Default = 1.
enable	Input	No	Clock enable input. The shift options also use the <code>enable</code> input for the clock enable. For serial operation, both <code>shiftin</code> and <code>enable</code> must be high.
aclr	Input	No	Asynchronous clear input. If both <code>aset</code> and <code>aclr</code> are used and both are asserted, <code>aclr</code> is dominant. The <code>aclr</code> input affects the output <code>q[]</code> values before polarity is applied to the ports.
aset	Input	No	Asynchronous set input. Sets <code>q[]</code> outputs to the value specified by <code>LPM_AVALUE</code> , if that value is present, or sets the <code>q[]</code> outputs to all 1s. If both <code>aset</code> and <code>aclr</code> are used and both are asserted, <code>aclr</code> is dominant. The <code>aset</code> input affects the output <code>q[]</code> values before polarity is applied to the ports.
shiftout	Output	No	Serial shift data output. This port is <code>LPM_WIDTH</code> wide. Either <code>q[]</code> or <code>shiftout</code> or both must be used.
q[]	Output	No	Data output from the shift register. This port is <code>LPM_WIDTH</code> wide. Either <code>q[]</code> or <code>shiftout</code> or both must be used.

Parameters

Name	Required	Value	Description
LPM_AVALUE	No	Integer > 0	Constant value that is loaded when <code>aset</code> is high.
LPM_DIRECTION	No	"LEFT" "RIGHT"	Direction of the shift register. The default value is "LEFT".
LPM_SVALUE	No	Integer ≥ 0	Constant value that is loaded on the rising edge of <code>clock</code> when <code>sset</code> is high. The default value is all 1s.
LPM_WIDTH	Yes	Integer > 0	Width of the <code>data[]</code> and <code>q[]</code> ports.



Notes:



Custom Parameterized Functions

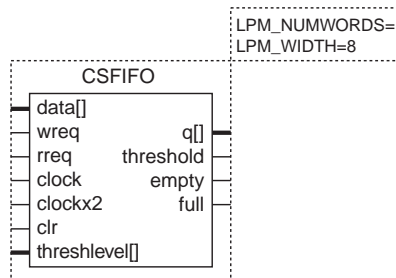
December 1996

csfifo.....	38
csdpram.....	39

csfifo

Cycle-Shared FIFO

The `csfifo` function is a custom function that provides a cycle-shared FIFO with both dynamic and static user threshold level controls. It also offers empty and full flag outputs.



Ports

Name	Type	Required	Description
data[]	Input	Yes	Data input to the <code>csfifo</code> . This port is <code>LPM_WIDTH</code> wide.
wreq	Input	Yes	Write request.
rreq	Input	Yes	Read request.
clock	Input	Yes	Positive-edge-triggered clock.
clockx2	Input	Yes	Positive-edge-triggered clock.
clr	Input	No	Resets <code>csfifo</code> to empty.
threshlevel[]	Input	No	Level (number of words) that the <code>threshold</code> output signal asserts.
q[]	Output	Yes	Data output from <code>csfifo</code> . This port is <code>LPM_WIDTH</code> wide.
threshold	Output	No	Indicates that <code>csfifo</code> contains greater than the <code>threshlevel[]</code> number of words.
empty	Output	No	Indicates that <code>csfifo</code> is empty.
full	Output	No	Indicates that <code>csfifo</code> is full.

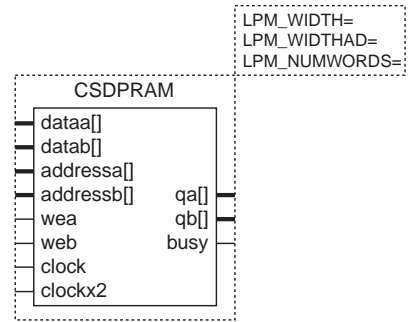
Parameters

Name	Required	Value	Description
LPM_NUMWORDS	No	Integer > 0	Number of words stored in memory. In general, this value should be (but is not required to be): $2^{LPM_WIDTH-1} < LPM_NUMWORDS \leq 2^{LPM_WIDTHAD}$. The default value is 2^{LPM_WIDTH} .
LPM_WIDTH	Yes	Integer > 0	Width of the <code>data[]</code> and <code>q[]</code> ports. The default value is 8.

csdpram

Parameterized Cycle-Shared Dual-Port RAM

The `csdpram` function is a custom function that has two address and two data ports that are cycle-shared. This function also provides a busy flag to indicate that the address on both ports is pointing to the same location and that a port will have priority.



Ports

Name	Type	Required	Description
<code>dataa[]</code>	Input	Yes	Data input to the memory. This port is <code>LPM_WIDTH</code> wide.
<code>datab[]</code>	Input	Yes	Data input to the memory. This port is <code>LPM_WIDTH</code> wide.
<code>addressa[]</code>	Input	Yes	Address input to the memory. This port is <code>LPM_WIDTHHAD</code> wide.
<code>addressb[]</code>	Input	Yes	Address input to the memory. This port is <code>LPM_WIDTHHAD</code> wide.
<code>wea</code>	Input	Yes	Write enable input.
<code>web</code>	Input	Yes	Write enable input.
<code>clock</code>	Input	Yes	Positive-edge-triggered clock.
<code>clockx2</code>	Input	Yes	Positive-edge-triggered clock.
<code>qa[]</code>	Output	Yes	Data output from the memory. This port is <code>LPM_WIDTH</code> wide.
<code>qb[]</code>	Output	Yes	Data output from the memory. This port is <code>LPM_WIDTH</code> wide.
<code>busy</code>	Output	No	Indicates that <code>addressa[] = addressb[]</code> and that <code>dataa[]</code> is writing data.

Parameters

Name	Required	Value	Description
<code>LPM_WIDTH</code>	Yes	Integer > 0	Width of the <code>dataa[]</code> , <code>datab[]</code> , <code>qa[]</code> , and <code>qb[]</code> ports.
<code>LPM_WIDTHHAD</code>	Yes	Integer > 0	Width of the <code>addressa[]</code> and <code>addressb[]</code> ports. <code>LPM_WIDTHHAD</code> should be (but is not required to be) equal to $\log_2(\text{LPM_NUMWORDS})$. If <code>LPM_WIDTHHAD</code> is too small, some memory locations will not be addressable. If it is too large, the addresses that are too high will return undefined logic levels.
<code>LPM_NUMWORDS</code>	No	Integer > 0	Number of words stored in memory. In general, this value should be (but is not required to be): $2^{\text{LPM_WIDTHHAD} - 1} < \text{LPM_NUMWORDS} \leq 2^{\text{LPM_WIDTHHAD}}$. The default value is $2^{\text{LPM_WIDTHHAD}}$.



Notes:



December 1996

Altera Regional Offices

NORTHERN CALIFORNIA (CORPORATE HEADQUARTERS)

Altera Corporation
2610 Orchard Parkway
San Jose, CA 95134-2020
TEL: (408) 894-7000
FAX: (408) 433-3943
(408) 894-7755

Altera Corporation
2290 N. First Street, Suite 212
San Jose, CA 95131
TEL: (408) 894-7900
FAX: (408) 894-7979

SOUTHERN CALIFORNIA

Altera Corporation
15375 Barranca Parkway, Suite B-201
Irvine, CA 92618
TEL: (714) 450-0262
FAX: (714) 450-0263

Altera Corporation
Olympic Plaza Executive Center
11500 West Olympic Boulevard, Suite 400
Los Angeles, CA 90064
TEL: (310) 312-4507
FAX: (310) 312-4508

Altera Corporation
5355 Mira Sorrento Place, Suite 100
San Diego, CA 92121
TEL: (619) 597-7518
FAX: (619) 597-7418

ARIZONA

Altera Corporation
2390 E. Camelback Road, Suite 300
Phoenix, Arizona 85016
TEL: (602) 553-1090
FAX: (602) 553-1198

COLORADO

Altera Corporation
Denver Technology Center
7900 East Union Avenue, #1100
Denver, CO 80237
TEL: (303) 694-5352
FAX: (303) 694-5351

Altera Corporation
14142 Denver West Parkway, #200
Golden, CO 80401
TEL: (303) 216-0167
FAX: (303) 277-0429

GEORGIA

Altera Corporation
3675 Crestwood Parkway, Suite 400
Duluth, GA 30136
TEL: (770) 935-6070
FAX: (770) 935-6073

ILLINOIS

Altera Corporation
475 N. Martingale Road, Suite 420
Schaumburg, IL 60173
TEL: (847) 240-0313
FAX: (847) 240-0266

MARYLAND

Altera Corporation
9891 Broken Land Parkway, Suite 300
Columbia, MD 21046
TEL: (410) 312-5708
FAX: (410) 309-0720

MASSACHUSETTS

Altera Corporation
238 Littleton Road, Suite 207
Westford, MA 01886
TEL: (508) 392-1100
FAX: (508) 392-1157

MINNESOTA

Altera Corporation
2850 Metro Drive, Suite 250
Bloomington, MN 55425
TEL: (612) 851-7861
FAX: (612) 858-7258

NEW JERSEY

Altera Corporation
575 State Highway 28
Raritan, NJ 08869
TEL: (908) 526-9400
FAX: (908) 526-5471

NORTH CAROLINA

Altera Corporation
5511 Capital Center Drive, Suite 110
Raleigh, NC 27606
TEL: (919) 852-1004
FAX: (919) 852-0809

OHIO

Altera Corporation
7784 Reynolds Road
Mentor, OH 44060
TEL: (216) 946-8211
FAX: (216) 946-9411

Altera Regional Offices

(Continued)

OREGON

Altera Corporation
11000 S.W. Stratus Street, Suite 330-17
Beaverton, OR 97008
TEL: (503) 643-8447
FAX: (503) 644-2345

TEXAS

Altera Corporation
5080 Spectrum Drive, Suite 812W
Dallas, TX 75248
TEL: (972) 701-2330
FAX: (972) 701-2331

TEXAS (continued)

Altera Corporation
9430 Research Boulevard
Echelon IV, Suite 400
Austin, TX 78759
TEL: (512) 343-4542
FAX: (512) 418-1186

CANADA

Altera Corporation
300 March Road, Suite 601B
Kanata, Ontario K2K 2E2
Canada
TEL: (613) 599-3141
FAX: (613) 599-3143

Altera International Sales Offices

UNITED STATES (CORPORATE HEADQUARTERS)

Altera Corporation
2610 Orchard Parkway
San Jose, CA 95134-2020
USA
TEL: (408) 894-7000
FAX: (408) 433-3943
(408) 894-7755

UNITED KINGDOM (EUROPEAN HEADQUARTERS)

Altera UK Limited
Holmers Farm Way
High Wycombe
Buckinghamshire
HP12 4XF
United Kingdom
TEL: (44) 1 494 602 000
FAX: (44) 1 494 602 001

NORTHERN EUROPE

Altera UK Limited
Holmers Farm Way
High Wycombe
Buckinghamshire
HP12 4XF
United Kingdom
TEL: (44) 1 494 602 020
FAX: (44) 1 494 602 021

BELGIUM

Altera Belgium
Katwilgweg 7B
2050 Antwerpen
Belgium
TEL: (32) 3 254 0420
FAX: (32) 3 254 0320

FRANCE

Altera France S.A.R.L.
Le Mermoz
13 Avenue Morane Saulnier
78140 Velizy
France
TEL: (33) 1 34 63 07 50
FAX: (33) 1 34 63 07 51

GERMANY

Altera GmbH
Max-Planck-Str. 5
D-85716 Unterschleissheim
Germany
TEL: (49) 89 3218 250
FAX: (49) 89 3218 2579

HONG KONG

Altera Hong Kong
Suite 1008, Tower 1
China Hong Kong City
33 Canton Road, Tsimshatsui
Kowloon, Hong Kong
TEL: (852) 2377-0218
FAX: (852) 2377-2811

ITALY

Altera Italia S.R.L.
Corso Lombardia 75
Autoporto Pescarito
10099 San Mauro, Torinese (Torino)
Italy
TEL: (39) 11 223 8588
FAX: (39) 11 223 8589

JAPAN

Altera Japan, Ltd.
Shinjuku Mitsui Building
2-1-1, Nishi-Shinjuku
Shinjuku-ku, Tokyo 162-04
Japan
TEL: (81) 3 3340 9480
FAX: (81) 3 3340 9487

KOREA

Altera Korea
Youndang Building, Suite 501
144-23 Samsung-dong, Kangnam-Ku
Seoul, Korea 135-090
TEL: (82) 2 538-6895
FAX: (82) 2 538-6896

SWEDEN

Altera AB
Sjoangsvagen 15
192 72 Sollentuna
Sweden
TEL: (46) 8 626 60 91
FAX: (46) 8 626 60 99

Altera, MAX, MAX+PLUS, MAX+PLUS II, and FLEX are trademarks and/or service marks of Altera Corporation in the United States and other countries. Altera Corporation acknowledges the trademarks of other organizations for their respective products or services mentioned in this document, specifically: Verilog and Cadence are a registered trademark of Cadence Design Systems, Inc. Data I/O is a registered trademark of Data I/O Corporation. Exemplar Logic is a registered trademark of Exemplar Logic, Inc. Mentor Graphics is a registered trademark of Mentor Graphics Corporation. MINC is a registered trademark of MINC Incorporated. SPARCstation is a trademark of SPARC International, Inc. and is licensed exclusively to Sun Microsystems, Inc. Sun Workstation is a registered trademark, and Sun is a trademark of Sun Microsystems, Inc. Synopsys is a registered trademark of Synopsys, Inc. VeriBest is a registered trademark of VeriBest Inc. Viewlogic is a registered trademark of Viewlogic Systems, Inc.

Altera reserves the right to make changes, without notice, in the devices or the device specifications identified in this document. Altera advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty. Testing and other quality control techniques are used to the extent Altera deems such testing necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed. In the absence of written agreement to the contrary, Altera assumes no liability for Altera applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does Altera warrant or represent any patent right, copyright, or other intellectual property right of Altera covering or relating to any combination, machine, or process in which such semiconductor devices might be or are used.

Altera's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Altera Corporation. As used herein:

1. Life support devices or systems are devices or systems that (a) are intended for surgical implant into the body or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Products mentioned in this document are covered by one or more of the following U.S. patents: 5,574,893; 5,572,717; 5,572,148; 5,572,067; 5,570,040; 5,567,177; 5,563,592; 5,557,217; 5,555,214; 5,550,842; 5,550,782; 5,548,552; 5,548,228; 5,543,732; 5,543,730; 5,541,530; 5,537,295; 5,537,057; 5,525,917; 5,523,247; 5,517,186; 5,498,975; 5,495,182; 5,493,519; 5,477,474; 5,463,328; 5,444,394; 5,438,295; 5,436,575; 5,436,574; 5,434,514; 5,432,467; 5,414,312; 5,399,922; 5,384,499; 5,376,844; 5,375,086; 5,371,422; 5,369,314; 5,359,243; 5,359,242; 5,353,248; 5,352,940; 5,350,954; 5,349,255; 5,341,308; 5,341,048; 5,341,044; 5,329,487; 5,317,212; 5,317,210; 5,315,172; 5,309,046; 5,301,416; 5,294,975; 5,285,153; 5,280,203; 5,274,581; 5,272,368; 5,268,598; 5,260,611; 5,260,610; 5,258,668; 5,247,478; 5,247,477; 5,243,233; 5,241,224; 5,237,219; 5,220,533; 5,220,214; 5,200,920; 5,187,392; 5,166,604; 5,162,680; 5,144,167; 5,138,576; 5,128,565; 5,121,006; 5,111,423; 5,097,208; 5,091,661; 5,066,873; 5,045,772; 4,969,121; 4,930,107; 4,930,098; 4,930,097; 4,912,342; 4,903,223; 4,899,070; 4,899,067; 4,871,930; 4,864,161; 4,831,573; 4,785,423; 4,774,421; 4,713,792; 4,677,318; 4,617,479; 4,609,986; 4,020,469; and certain foreign patents.

