1. DESCRIPTION OF LPM MODULES

This section describes in detail the functionality and semantics of each module in LPM.

1.1 MODULE CATEGORIES

The LPM modules fall into five major categories:

CATEGORY	**DESCRIPTION**
GATES	
LPM_CONSTANT	Constant value
LPM_INV, LPM_AND, LPM_OR, LPM_XOR	Basic combinatorial gates
LPM_BUSTRI	Tri-State buffer
LPM_MUX	Multiplexer
LPM_DECODE	Decoder
LPM_CLSHIFT	Combinatorial shifter
ARITHMETIC COMPONENTS	
LPM_COUNTER	Counter
LPM_ADD_SUB	Adder/Subtracter
LPM_COMPARE	Comparator
LPM_MULT	Multiplier
LPM_ABS	Absolute Value
LPM_DIVIDE	Divider
STORAGE COMPONENTS	
LPM_ROM	Read Only Memory
LPM_LATCH	Transparent latch
LPM_FF	D-type or T-type flip-flop
LPM_SHIFTREG	Shift Register
LPM_RAM_DQLPM_RAM_IO,	Random Access Memory
LPM_RAM_DP	Dual-Port Ramdom Access Memory
LPM_FIFO	Single colock First-In-First-Out Memory
LPM_FIFO_DC	Double colocks First-In-First-Out Memory
TABLE PRIMITIVES	
LPM_FSM	Finite state machine
LPM_TTABLE	Truth table
PAD PRIMITIVES	
LPM_INPAD, LPM_OUTPAD, LPM_BIPAD	Input/Output/Bidrectional pads

Note that truth table, finite state machine, RAM and ROM modules require more information than can be contained in the LPM netlist to define their function. These modules use supporting files to describe their function. These supporting files use the standard Intel HEX, Berkeley PLA and KISS formats.

1.1.1 Logic Conventions

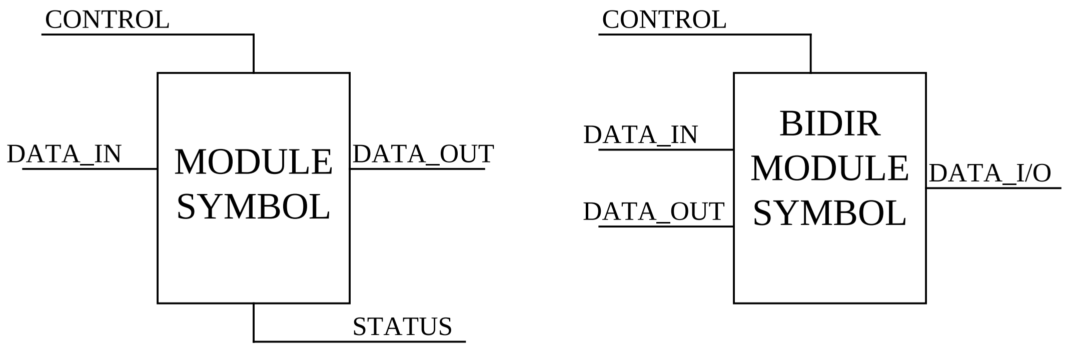
Where logic equations or logic models are used, the following symbols are used for both single bit operations and bit-wise vector operations:

AND	&	
OR	|	
XOR	^	
NOT or INVERT	~	on vectors this is 1's complement
NAND	~&	
NOR	~|	
XNOR	~^	
LEFT SHIFT	<<	vector only
RIGHT SHIFT	>>	vector only
Two's Complement or Unsigned Add	+	
Two's Complement or Unsigned Subtract	–	
Two's Complement or Unsigned Multiply	×	

1.1.2 Drawing Conventions

The drawings of all modules in this document use the following conventions (shown in the figure below):

Data Inputs	Are shown going into the left of the module symbol.
Data Outputs	Are shown coming out of the right side of the module symbol except on bidir module symbols that have bi-directional port where they are on the left side.
Control Inputs	Are shown going into the top of the module symbol.
Status Outputs	Are shown coming out of the bottom of the module symbol.
Data I/Os	Are shown on the right side of the bidir module symbol.
Width & Size	Width refers to LPM_Width and Size refers to LPM_Size

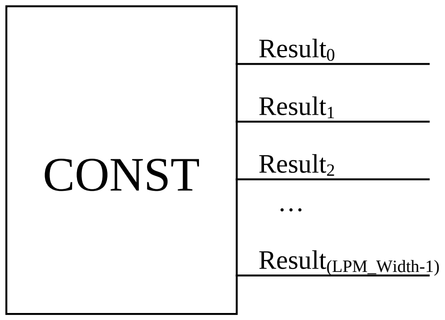


1.1.3 Scan Test Conventions

All of the modules that have scan-test ports (LPM_COUNTER, LPM_LATCH, LPM_FF, LPM_TFF, LPM_FSM) use the same convention. The **TestOut** port always has the same value as the most significant bit of the output (**Q** or **State**). When **TestEnab** is high, the data on **TestIn** is shifted into the least significant bit of the associated register as the contents of the register are shifted towards the most significant bit.

1.2 GATES

1.2.1 LPM_CONSTANT



1.2.1.1 Ports

Port Name	Type	Usage	Description	Comments
Result	O	Required	Value specified by CValue	Output vector LPM_Width wide LPM_Cvalue is truncated or sign extended to LPM_Width bits

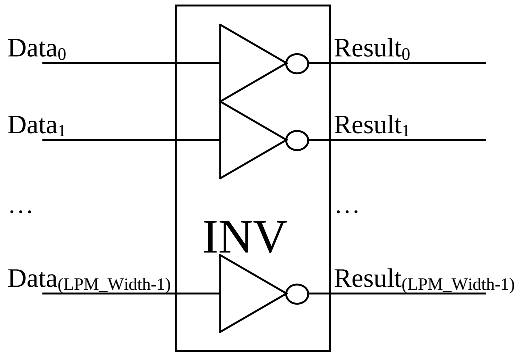
1.2.1.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of output vector
LPM_CValue	Required	LPM Value	Value of constant
LPM_Strength	Optional	WEAK	If present, this indicates a pullup or pulldown strength

1.2.1.3 Function

Result = LPM_Cvalue

1.2.2 LPM_INV



1.2.2.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input	Vector, LPM_Width wide
Result	O	Required	Inverted Result	Vector, LPM_Width wide

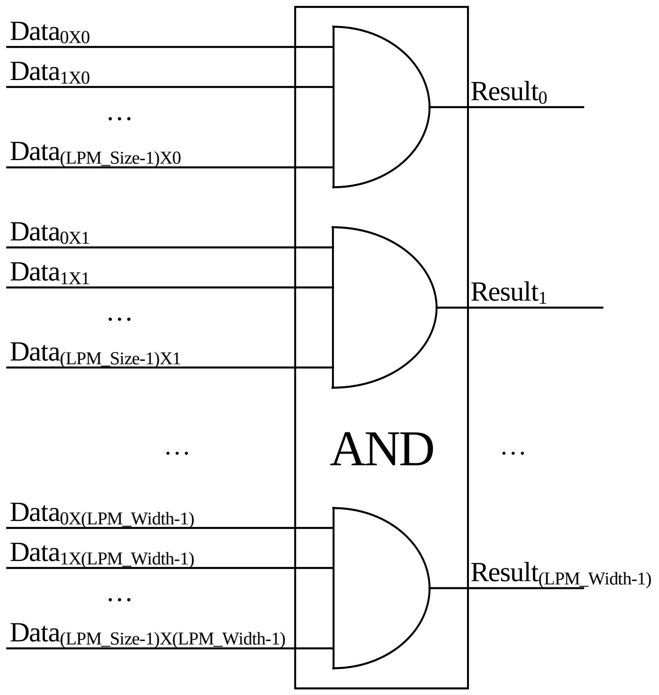
1.2.2.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors

1.2.2.3 Function

Result = ~ Data

1.2.3 LPM_AND



1.2.3.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input	Vector, LPM_Size times LPM_Width wide
Result	O	Required	Result of AND operators	Vector, LPM_Width wide

1.2.3.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of output vector. Number of AND gates.
LPM_Size	Required	LPM Value > 0	Number of inputs to each AND gate. Number of input buses.

1.2.3.3 Function

$$\mathbf{Result_0 = Data_{0X0} \,\&\, Data_{1X0} \,\&\, Data_{2X0} \,\&\, \ldots \,\&\, Data_{LPM\_Size-1X0}}$$

$$\mathbf{Result_1 = Data_{0X1} \,\&\, Data_{1X1} \,\&\, Data_{2X1} \,\&\, \ldots \,\&\, Data_{LPM\_Size-1X1}}$$

$$\mathbf{Result_2 = Data_{0X2} \,\&\, Data_{1X2} \,\&\, Data_{2X2} \,\&\, \ldots \,\&\, Data_{LPM\_Size-1X2}}$$

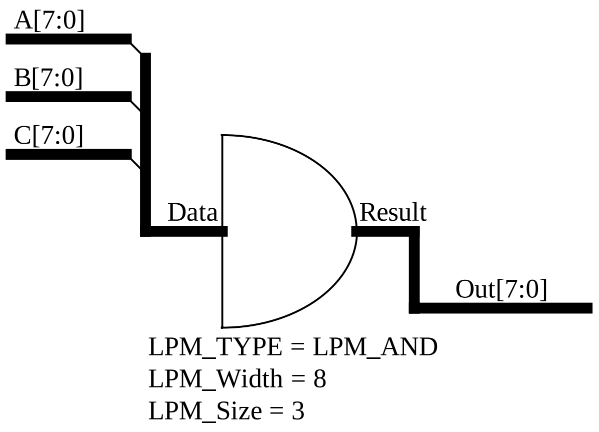
…

$$\mathbf{Result_i = Data_{0Xi} \,\&\, Data_{1Xi} \,\&\, Data_{2Xi} \,\&\, \ldots \,\&\, Data_{LPM\_Size-1Xi}}$$

Where **i** goes from 0 to (LPM_Width - 1).

1.2.3.4 Example

Suppose the designers have three 8-bit buses and they want to AND the corresponding bits of the three buses. This is done using an LPM_AND with an LPM_Width = 8 and an LPM_Size = 3. The LPM_Width of eight indicates that there are eight AND gates, and the LPM_Size of three indicates that each AND gate has three inputs.



This diagram is for illustrative purposes only and is not intended to specify any implementation details.

The function performed by the LPM_AND gate in this case is:

Out[0] = Result_0 = Data_{0X0} & Data_{1X0} & Data_{2X0} = C[0] & B[0] & A[0]

Out[1] = Result_1 = Data_{0X1} & Data_{1X1} & Data_{2X1} = C[1] & B[1] & A[1]

Out[2] = Result_2 = Data_{0X2} & Data_{1X2} & Data_{2X2} = C[2] & B[2] & A[2]

Out[3] = Result_3 = Data_{0X3} & Data_{1X3} & Data_{2X3} = C[3] & B[3] & A[3]

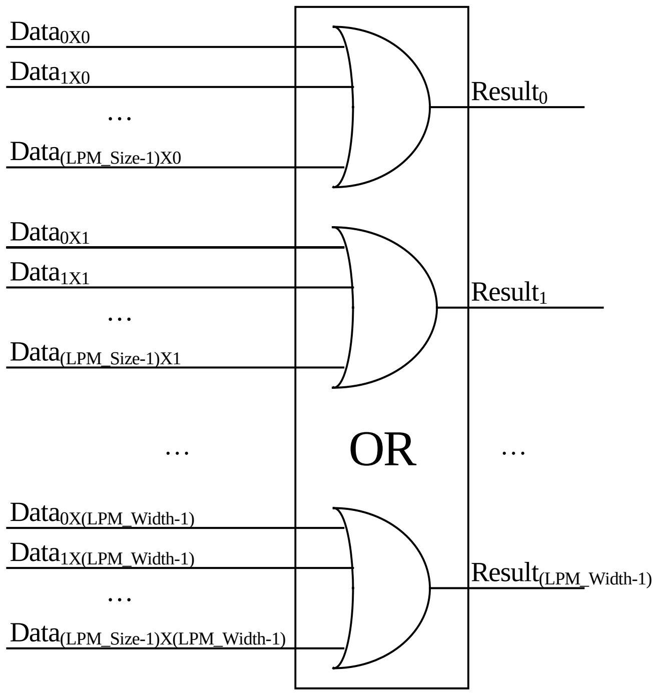
Out[4] = Result_4 = Data_{0X4} & Data_{1X4} & Data_{2X4} = C[4] & B[4] & A[4]

Out[5] = Result_5 = Data_{0X5} & Data_{1X5} & Data_{2X5} = C[5] & B[5] & A[5]

Out[6] = Result_6 = Data_{0X6} & Data_{1X6} & Data_{2X6} = C[6] & B[6] & A[6]

Out[7] = Result_7 = Data_{0X7} & Data_{1X7} & Data_{2X7} = C[7] & B[7] & A[7]

1.2.4 LPM_OR



1.2.4.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input	Vector, LPM_Size times LPM_Width wide
Result	O	Required	Result of OR operators	Vector, LPM_Width wide

1.2.4.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of output vector. Number of OR gates.
LPM_Size	Required	LPM Value > 0	Number of inputs to each OR gate. Number of input buses.

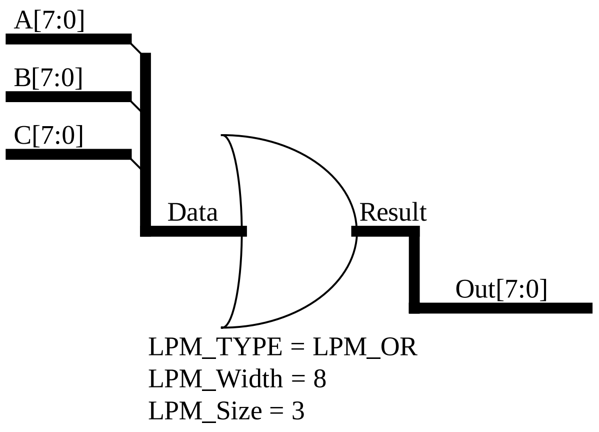
1.2.4.3 Function

$$\mathbf{Result_i = Data_{0Xi} \mid Data_{1Xi} \mid Data_{2Xi} \mid \ldots \mid Data_{LPM\_Size-1Xi}}$$

Where **i** goes from 0 to (LPM_Width - 1).

1.2.4.4 Example

Suppose the designers have three 8-bit buses and they want to OR the corresponding bits of the three buses. This is done using an LPM_OR with an LPM_Width of 8 and an LPM_Size of three. The LPM_Width of eight indicates that there are eight OR gates, and the LPM_Size of three indicates that each OR gate has three inputs.



This diagram is for illustrative purposes only and is not intended to specify any implementation details.

The function performed by the LPM_OR gate in this case is:

Out[0] = Result_0 = Data_{2X0} | Data_{1X0} | Data_{0X0} = A[0] | B[0] | C[0]

Out[1] = Result_1 = Data_{2X1} | Data_{1X1} | Data_{0X1} = A[1] | B[1] | C[1]

Out[2] = Result_2 = Data_{2X2} | Data_{1X2} | Data_{0X2} = A[2] | B[2] | C[2]

Out[3] = Result_3 = Data_{2X3} | Data_{1X3} | Data_{0X3} = A[3] | B[3] | C[3]

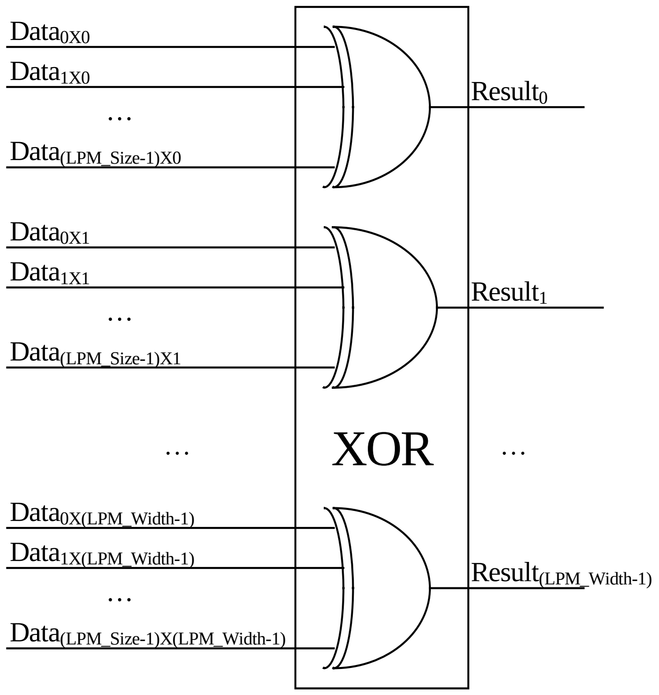
Out[4] = Result_4 = Data_{2X4} | Data_{1X4} | Data_{0X4} = A[4] | B[4] | C[4]

Out[5] = Result_5 = Data_{2X5} | Data_{1X5} | Data_{0X5} = A[5] | B[5] | C[5]

Out[6] = Result_6 = Data_{2X6} | Data_{1X6} | Data_{0X6} = A[6] | B[6] | C[6]

Out[7] = Result_7 = Data_{2X7} | Data_{1X7} | Data_{0X7} = A[7] | B[7] | C[7]

1.2.5 LPM_XOR



1.2.5.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input	Vector, LPM_Size times LPM_Width wide
Result	O	Required	Result of XOR operators	Vector, LPM_Width wide

1.2.5.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of output vector. Number of XOR gates.
LPM_Size	Required	LPM Value > 0	Number of inputs to each XOR gate. Number of input buses.

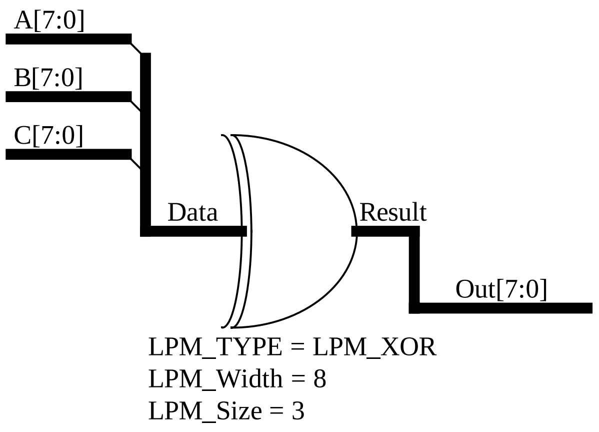
1.2.5.3 Function

$$\mathbf{Result_i = Data_{0Xi} \wedge Data_{1Xi} \wedge Data_{2Xi} \wedge \ldots \wedge Data_{LPM\_Size-1Xi}}$$

Where **i** goes from 0 to (LPM_Width - 1).

1.2.5.4 Example

Suppose the designers have three 8-bit buses and they want to XOR the corresponding bits of the three buses. This is done using an LPM_XOR with an LPM_Width of 8 and an LPM_Size of three. The LPM_Width of eight indicates that there are eight XOR gates, and the LPM_Size of three indicates that each XOR gate has three inputs.



This diagram is for illustrative purposes only and is not intended to specify any implementation details.

The function performed by the LPM_XOR gate in this case is:

Out[0] = Result_0 = Data_{2X0} ^ Data_{1X0} ^ Data_{0X0} = A[0] ^ B[0] ^ C[0]

Out[1] = Result_1 = Data_{2X1} ^ Data_{1X1} ^ Data_{0X1} = A[1] ^ B[1] ^ C[1]

Out[2] = Result_2 = Data_{2X2} ^ Data_{1X2} ^ Data_{0X2} = A[2] ^ B[2] ^ C[2]

Out[3] = Result_3 = Data_{2X3} ^ Data_{1X3} ^ Data_{0X3} = A[3] ^ B[3] ^ C[3]

Out[4] = Result_4 = Data_{2X4} ^ Data_{1X4} ^ Data_{0X4} = A[4] ^ B[4] ^ C[4]

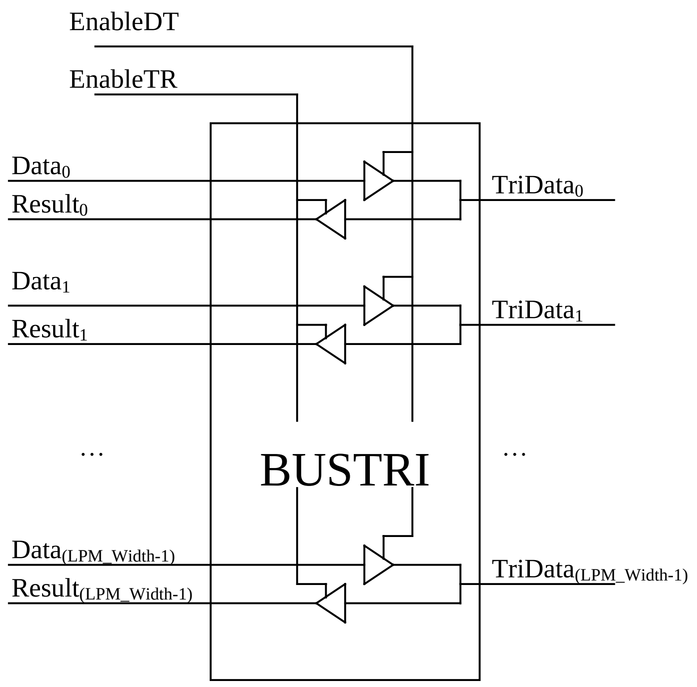
Out[5] = Result_5 = Data_{2X5} ^ Data_{1X5} ^ Data_{0X5} = A[5] ^ B[5] ^ C[5]

Out[6] = Result_6 = Data_{2X6} ^ Data_{1X6} ^ Data_{0X6} = A[6] ^ B[6] ^ C[6]

Out[7] = Result_7 = Data_{2X7} ^ Data_{1X7} ^ Data_{0X7} = A[7] ^ B[7] ^ C[7]

1.2.6 LPM_BUSTRI

Connection to a Tri-State Bus.



1.2.6.1 Ports

Port Name	Type	Usage	Description	Comments
TriData	IO	Required	Bi-directional bus signal	Vector, LPM_Width wide
Data	I	Note 1	Data input to **TriData** bus	Vector, LPM_Width wide. One of **Data** or **Result** must be used.
EnableDT	I	Optional	If HIGH, enables **Data** onto the **TriData** bus.	Default value is Low. Required if **Data** is used.
Result	O	Note 1	Output from **TriData** bus.	Vector, LPM_Width wide. One of **Data** or **Result** must be used.
EnableTR	I	Optional	If HIGH, enable **TriData** onto the **Result** bus.	Default value is Low. Required if **Result** is used

Note 1: Either the **Result** or **Data** port is required. Both may be used.

1.2.6.2 Properties

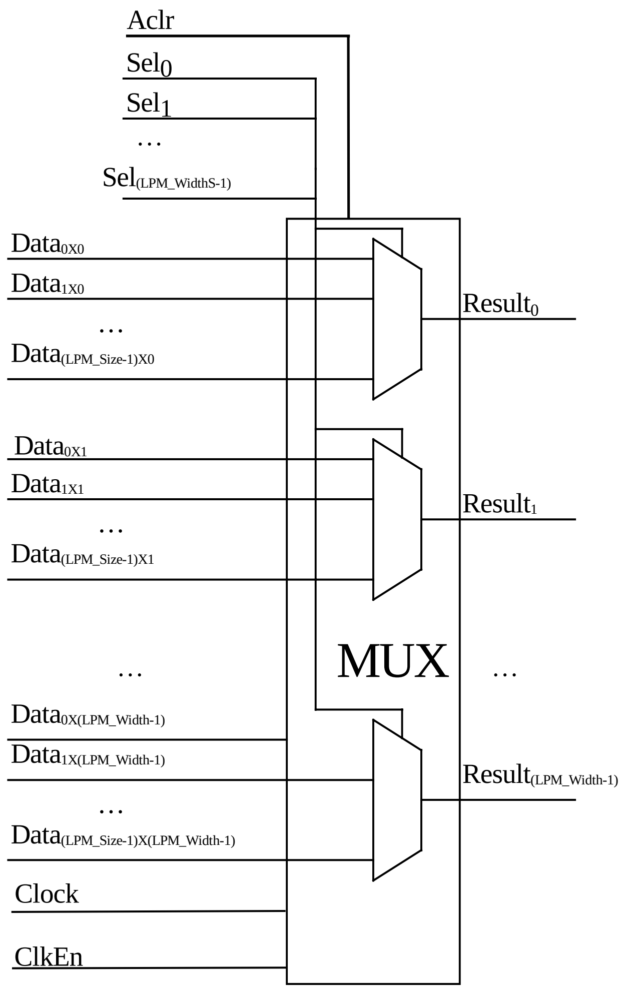
Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors

1.2.6.3 Functions

EnableDT	EnableTR	Data	Result	TriData
L	L	X	Hi-Z	Hi-Z Note 1
L	H	X	VALUE (From **TriData**)	VALUE
H	L	VALUE	Hi-Z	VALUE (From **Data**)
H	H	VALUE	VALUE (From **Data**)	VALUE (From **Data**)

Note 1: When both control ports (EnableDT and EnableTR) are inactive (LOW) the Result port is high impedance, and the TriData port will take its value from the attached net (i.e. it is not driven by the LPM_Bustri).

1.2.7 LPM_MUX



1.2.7.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	**Data** input	Vector, LPM_Size times LPM_Width wide
Result	O	Required	Selected input vector	Vector, LPM_Width wide
Sel	I	Required	Selects one of the input vectors	Vector, LPM_WidthS wide
Clock	I	Optional	Clock for pipelined usage	Note 1
ClkEn	I	Optional	Clock enable for pipelined	Note 2
Aclr	I	Optional	Asynchronous Clear	Note 3

Note 1: The **Clock** port provides for pipelined operation of the LPM_MUX. If a lpm_pipeline other than 0 (default value) is specified, then the clock port must be connected.

Note 2: The **ClkEn** port provides a clock enable for pipelined operation.

Note 3: The pipeline initializes to undefined. The **Aclr** port may be used at any time to reset the pipeline to all 0's asynchronously to the clock.

1.2.7.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of output vector. Number of Multiplexers.
LPM_Size	Required	LPM Value > 0	Number of inputs to each Multiplexer. Number of input buses.
LPM_WidthS	Required	LPM Value >0	WidthS should be the next integer greater than or equal to $\log_2$(LPM_Size) or there will be unselectable input vectors.
LPM_Pipeline	Optional	LPM Value ≥ 0	Default is 0 - non-pipelined

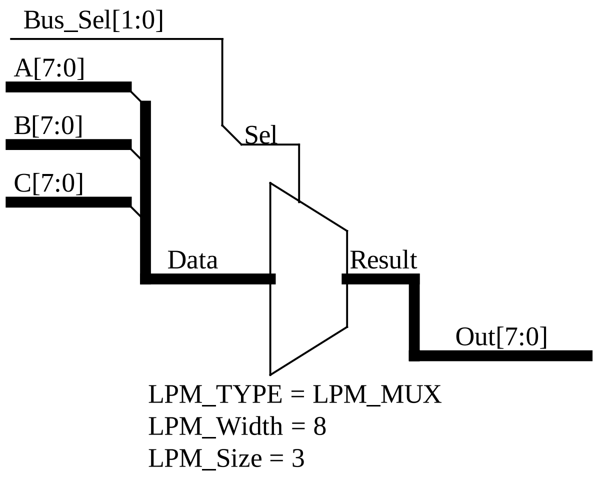
1.2.7.3 Functions

Sel vector	Sel Value	Result
0000…000	0	$\text{Data}_{0\_[\text{LPM\_Width}-1:0]}$
0000…001	1	$\text{Data}_{1\_[\text{LPM\_Width}-1:0]}$
0000…010	2	$\text{Data}_{2\_[\text{LPM\_Width}-1:0]}$
…	…	…
1111…110	LPM_Size-2	$\text{Data}_{\text{LPM\_Size}-2\_[\text{LPM\_Width}-1:0]}$
1111…111	LPM_Size-1	$\text{Data}_{\text{LPM\_Size}-1\_[\text{LPM\_Width}-1:0]}$

This table assumes that LPM_Size is a power of two, but that is not required. If there is no **Data** vector that corresponds to the 'Sel Value'; that is, if **Data**$_{\text{Sel\_Value}}$ is not connected or is greater than LPM_Size, the selection of 'Sel Value' will produce an undefined **Result**.

1.2.7.4 Example

Suppose the designers have three 8-bit buses and they want to select one of the three buses. This is done using an LPM_MUX with an LPM_Width of 8 and an LPM_Size of three. The LPM_Width of eight indicates that there are eight multiplexers, and the LPM_Size of three indicates that each multiplexer has three inputs.



This diagram is for illustrative purposes only and is not intended to specify any implementation details.

Supposing that bus A becomes $\mathbf{Data}_{2Xi}$, bus B becomes $\mathbf{Data}_{1Xi}$, and bus C becomes $\mathbf{Data}_{0Xi}$, the function performed by the LPM_MUX gate in this case is:

Out[0] = $\text{Result}_0 = \text{Data}_{\text{SelX0}}$ = (UNDEFINED if Sel = 3, A[0] if Sel = 2, B[0] if Sel = 1, C[0] if Sel = 0)

Out[1] = $\text{Result}_1 = \text{Data}_{\text{SelX1}}$ = (UNDEFINED if Sel = 3, A[1] if Sel = 2, B[1] if Sel = 1, C[1] if Sel = 0)

Out[2] = $\text{Result}_2 = \text{Data}_{\text{SelX2}}$ = (UNDEFINED if Sel = 3, A[2] if Sel = 2, B[2] if Sel = 1, C[2] if Sel = 0)

Out[3] = $\text{Result}_3 = \text{Data}_{\text{SelX3}}$ = (UNDEFINED if Sel = 3, A[3] if Sel = 2, B[3] if Sel = 1, C[3] if Sel = 0)

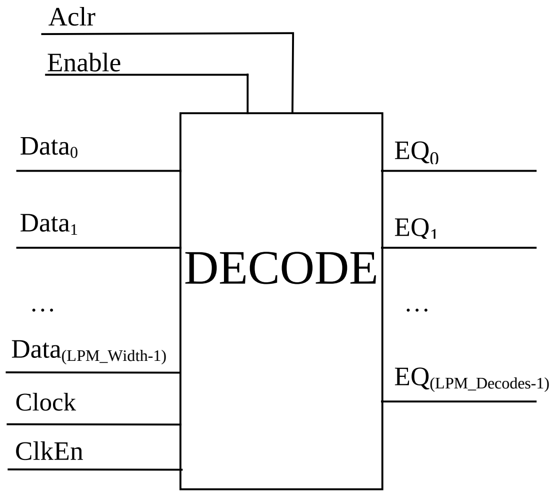
Out[4] = $\text{Result}_4 = \text{Data}_{\text{SelX4}}$ = (UNDEFINED if Sel = 3, A[4] if Sel = 2, B[4] if Sel = 1, C[4] if Sel = 0)

Out[5] = $\text{Result}_5 = \text{Data}_{\text{SelX5}}$ = (UNDEFINED if Sel = 3, A[5] if Sel = 2, B[5] if Sel = 1, C[5] if Sel = 0)

Out[6] = $\text{Result}_6 = \text{Data}_{\text{SelX6}}$ = (UNDEFINED if Sel = 3, A[6] if Sel = 2, B[6] if Sel = 1, C[6] if Sel = 0)

Out[7] = $\text{Result}_7 = \text{Data}_{\text{SelX7}}$ = (UNDEFINED if Sel = 3, A[7] if Sel = 2, B[7] if Sel = 1, C[7] if Sel = 0)

1.2.8 LPM_DECODE



1.2.8.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input. Treated as unsigned binary number.	Vector, LPM_Width wide
Enable	I	Optional	Enable. All outputs low when not active	Default value is Active (High) if absent.

EQ	O	Required	For i = 0 to LPM_Decodes
			if (i = **Data**) $Eq_i = 1$
			else $Eq_i = 0$
				Vector, LPM_Decodes wide.
				If **Data** ≥ LPM_Decodes then all Eq_i are 0

Clock	I	Optional	Clock for pipelined usage	Note 1
ClkEn	I	Optional	Clock enable for pipelined	Note 2
Aclr	I	Optional	Asynchronous Clear	Note 3

Note 1: The **Clock** port provides for pipelined operation of the LPM_DECODE. If a lpm_pipeline other than 0 (default value) is specified, then the clock port must be connected.

Note 2: The **ClkEn** port provides a clock enable for pipelined operation.

Note 3: The pipelined initializes to undefined. The **Aclr** port may be used at any time to reset the pipeline to all 0's asynchronously to the clock.

1.2.8.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input vector
LPM_Decodes	Required	$0 < \text{LPM Value} \leq 2^{\text{LPM\_Width}}$	Number of explicit decodes
LPM_Pipeline	Optional	LPM Value ≥ 0	Default is 0 - non-pipelined

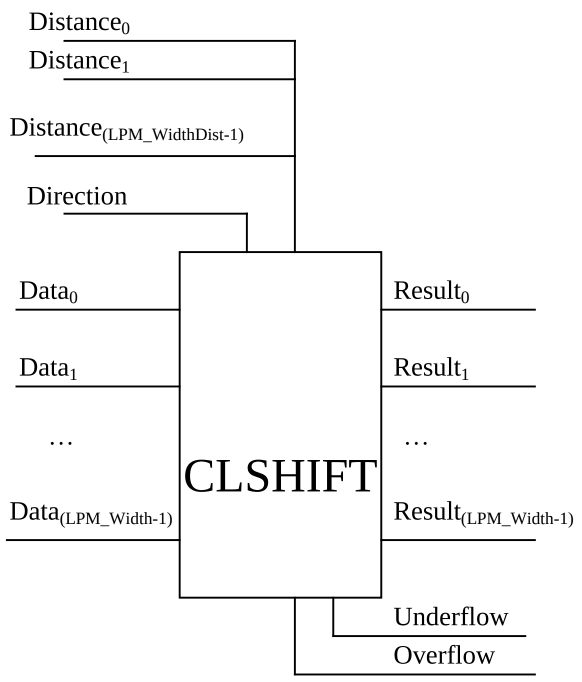
1.2.8.3 Functions

Enable	Data vector	Eq_i that is 1 (high) all other Eq_i = 0 (low)
L	X	NONE
H	0000…000	Eq_0
H	0000…001	Eq_1
…	…	…
H	000…0101	Eq_5
…	…	…
H	Data = LPM_Decodes -1	$Eq_{(LPM\_Decodes-1)}$
H	Data = LPM_Decodes	NONE
H	Data > LPM_Decodes	NONE

Note 1: If **Data** = i and Eq_i is not connected or does not appear in the symbol then all outputs will be low.

1.2.9 LPM_CLSHIFT

Combinatorial Logic shifter. Barrel Shifter.



1.2.9.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data to be shifted.	Vector, LPM_Width wide
Distance	I	Required	Number of positions to shift **Data**.	Vector, LPM_WidthDist wide

Direction	I	Optional	Direction of shift.
			Low = Left (toward MSB)
			High = Right (toward LSB)
				Default value is 0 (Low) = Left (toward the MSB)

Result	O	Required	Shifted **Data**	Vector, LPM_Width wide
Overflow	O	Optional	Logical or Arithmetic Overflow	Note 1
Underflow	O	Optional	Logical or Arithmetic Underflow	Note 1

Note 1: If the LPM_ShiftType is **ROTATE** and Overflow or Underflow are connected, the output of those ports will be undefined.

1.2.9.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input vector
LPM_WidthDist	Optional	LPM Value > 0	Width of the Distance Port Note 1
LPM_ShiftType	Optional	LOGICAL | ROTATE | ARITHMETIC	Default is LOGICAL Note 2

Note 1: LPM_WidthDist specifies the width of the Distance port. The values on the Distance port would normally range from 0 which would mean "no shift" to (LPM_Width-1) which would be the maximum meaningful shift. The typical value assigned to LPM_WidthDist would be "the smallest integer not less than $\log_2$(LPM_Width)" or $\lceil \log_2 \text{LPM\_Width} \rceil$. Any value on the Distance port greater than LPM_Width-1 results in an UNDEFINED output.

Note 2: The sign bit is extended for ARITHMETIC. For a LOGICAL right shift 0's are always shifted into the MSB.

1.2.9.3 Functions

The LPM_CLSHIFT module acts like a barrel-shifter. It is entirely combinational logic.

Overflow occurs when the shifted result exceeds the precision of the **Result** bus. For LOGICAL values, overflow occurs when all ones have been shifted out. For ARITHMETIC value, overflow occurs a significant digit is shifted into or past the sign bit.

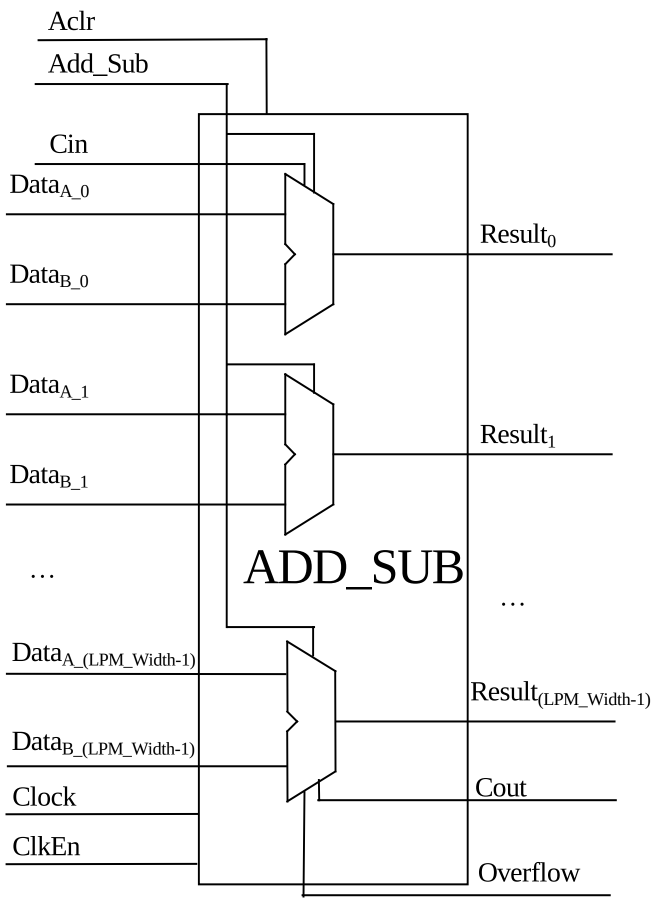
Underflow occurs when the shifted result contains no significant digits.

LPM_ShiftType	Direction	Function
LOGICAL	0 = Left	Result = Data << Distance
LOGICAL	1 = Right	Result = Data >> Distance
ROTATE	0 = Left	$\text{Result}_i = \text{Data}_x$ where x is ((Distance + i) mod LPM_Width)
ROTATE	1 = Right	$\text{Result}_i = \text{Data}_x$ where x is ((Distance - i) mod LPM_Width)
ARITHMETIC	0 = Left	Result = DATA * $2^{\text{LPM\_WidthDist}}$
ARITHMETIC	1 = Right	Result = DATA ÷ $2^{\text{LPM\_WidthDist}}$ (integer divide)

Values larger than (LPM_Width - 1) result in an UNDEFINED output.

1.3 ARITHMETIC COMPONENTS

1.3.1 LPM_ADD_SUB



1.3.1.1 Ports

Port Name	Type	Usage	Description	Comments
DataA	I	Required	Augend/Minuend	Vector, LPM_Width wide
DataB	I	Required	Addend/Subtrahend	Vector, LPM_Width wide

Cin	I	Optional	Carry in to the low order bit
			OP=ADD Low= 0 High= +1
			OP=SUB Low = -1 High= 0
				If not connected, default value is LOW.

Add_Sub	I	Optional	The Operation to be performed =H: OP = ADD =L: OP = SUB	Cannot be used it LPM_Direction property is used. If not connected, defaults value is ADD.
Result	O	Required	**DataA** ± **DataB** ± **Cin**	Vector, LPM_Width wide
Cout	O	Optional	Carry-out (~Borrow-in) of Most Significant Bit (MSB)	Note 1

Overflow	O	Optional	Result exceeds available precision.	Note 2
Clock	I	Optional	Clock for pipelined usage	Note 3
Clken	I	Optional	Clock enable for pipelined	Note 4
Aclr	I	Optional	Asynchronous Clear	Note 5

Note 1: **Cout** has a physical interpretation as the carry-out (~borrow-in) of the most significant bit. Cout is most meaningful for detecting overflow in **unsigned** numbers. See Table 1 for the arithmetic interpretation of **Cout** = 1.

Note 2: **Overflow** has a physical interpretation as the XOR (exclusive or) of the carry into the MSB with the carry out of the MSB. Overflow is only meaningful when the LPM_Representation is **signed**. It indicates that the **Result** has exceeded the available precision. See Table 2 for the arithmetic interpretation of **Overflow** = 1.

Note 3: The **Clock** port provides for pipelined operation of the LPM_ADD_SUB. If a lpm_pipeline other than 0 (default value) is specified, then the clock port must be connected.

Note 4: The **ClkEn** port provides a clock enable for pipelined operation.

Note 5: The pipelined initializes to undefined. The **Aclr** port may be used at any time to reset the pipeline to all 0's asynchronously to the clock

Table 1: Arithmetic interpretation of Cout = 1

	OP = ADD	OP = SUB
Unsigned	(**DataA** + **DataB** + **Cin**) > 2^{LPM_Width}-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.

Table 2: Arithmetic interpretation of Overflow = 1

	OP=ADD	OP=SUB
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) < $-2^{LPM\_Width-1}$

1.3.1.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of **DataA**, **DataB** and **Result**
LPM_Direction	Optional	ADD | SUB	Default is ADD. **Add_Sub** port may not be used if this property is used.
LPM_Representation	Optional	UNSIGNED or SIGNED	Default is SIGNED
LPM_Pipeline	Optional	LPM Value ≥ 0	Default is 0 - non-pipelined

1.3.1.3 Functions

$$\mathbf{Result_i} = \mathbf{DataA_i} \wedge \mathbf{DataB_i} \wedge \mathbf{Cin_i} \wedge (\sim \mathbf{Add\_Sub})$$

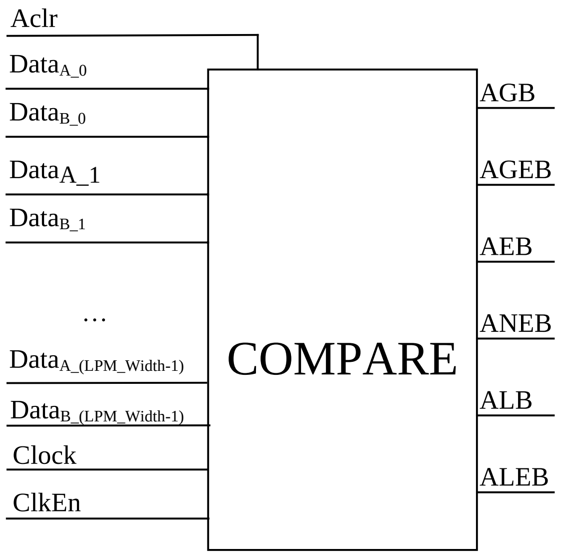
Cout = carry out of the MSB

Overflow = the XOR of the carry into the MSB and **Cout**

When **Cout** is prepended to the **Result**, the result is a vector that always has sufficient precision to represent the result of the operation.

$$\{\mathbf{Cout},\mathbf{Result}\} = \mathbf{DataA} + \mathbf{Cin} \pm \mathbf{DataB}$$

1.3.2 LPM_COMPARE



1.3.2.1 Ports

Port Name	Type	Usage	Description	Comments
DataA	I	Required	**DataB** is compared to this.	Vector, LPM_Width wide
DataB	I	Required	This is compared to **DataA**	Vector, LPM_Width wide
AGB	O	Note 1	High (1) if **DataA** > **DataB**	
AGEB	O	Note 1	High (1) if **DataA** ≥ **DataB**	
AEB	O	Note 1	High (1) if **DataA** = **DataB**	
ANEB	O	Note 1	High (1) if **DataA** ≠ **DataB**	
ALB	O	Note 1	High (1) if **DataA** < **DataB**	
ALEB	O	Note 1	High (1) if **DataA** ≤ **DataB**	
Clock	I	Optional	Clock for pipelined usage	Note 2
ClkEn	I	Optional	Clock enable for pipelined	Note 3
Aclr	I	Optional	Asynchronous Clear	Note 4

Note 1: At least one of the 6 output ports must be connected.

Note 2: The **Clock** port provides for pipelined operation of the LPM_COMPARE. If a lpm_pipeline other than 0 (default value) is specified, then the **Clock** port must be connected.

Note 3: The **ClkEn** port provides a clock enable for pipelined operation.

Note 4: The pipelined initializes to undefined. The **Aclr** port may be used at any time to reset the pipeline to all 0's asynchronously to the clock.

1.3.2.2 Properties

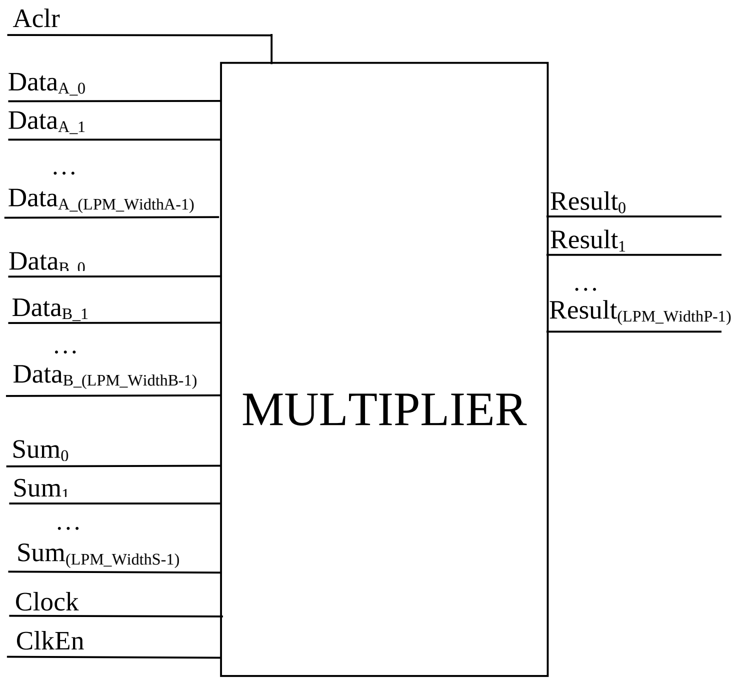
Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of **DataA** and **DataB**
LPM_Representation	Optional	UNSIGNED or SIGNED	Default is UNSIGNED.
LPM_Pipeline	Optional	LPM Value ≥0	Default is 0 - non-pipelined

1.3.2.3 Functions

Signed or unsigned comparison of the value represented by **DataA** versus the value represented by **DataB**. Note that:

AEB = ~ANEB ALB = ~AGEB AGB = ~ALEB

1.3.3 LPM_MULT



1.3.3.1 Ports

Port Name	Type	Usage	Description	Comments
DataA	I	Required	Multiplicand	Vector, LPM_WidthA wide
DataB	I	Required	Multiplier	Vector, LPM_WidthB wide
Sum	I	Optional	Partial Sum	Vector, LPM_WidthS wide. Note 1
Result	O	Required	Product	Vector, LPM_WidthP wide. Note 2
Clock	I	Optional	Clock for pipelined usage	Note 3
ClkEn	I	Optional	Clock enable for pipelined	Note 4
Aclr	I	Optional	Asynchronous Clear	Note 5

Note 1: An extra bit should be reserved in the LPM_WidthS if a carry out is expected from addition of the Product and the Partial Sum. LPM_WidthS should be larger than LPM_WidthA plus LPM_WidthB to guarantee that the carry out will be represented in Result.

Note 2: The product is a vector, LPM_WidthP bits wide. If LPM_WidthP is less than the maximum of either LPM_WidthA plus LPM_WidthB or LPM_WidthS, then only the LPM_WidthP most significant bits are present. See 1.3.3.3.

Note 3: The **Clock** port provides for pipelined operation of the LPM_MULT. If a lpm_pipeline other then 0 (default value) is specified, then the clock port must be connected.

Note 4: The **ClkEn** port provides a clock enable for pipelined operation.

Note 5: The pipelined initializes to undefined. The **Aclr** port may be used at any time to reset the pipeline to all 0's asynchronously to the clock.

1.3.3.2 Properties

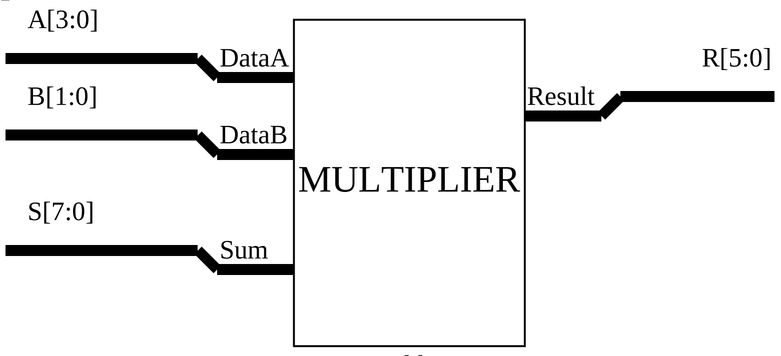
Property	Usage	Value	Comments
LPM_WidthA	Required	LPM Value > 0	Width of **DataA**
LPM_WidthB	Required	LPM Value > 0	Width of **DataB**
LPM_WidthS	Optional	LPM Value > 0	Width of **Sum.** Required if the **Sum** port is used.
LPM_WidthP	Required	LPM Value > 0	Width of **Result.** This represents the LPM_WidthP most significant bits.
LPM_Representation	Optional	UNSIGNED or SIGNED	Default is UNSIGNED.
LPM_Pipeline	Optional	LPM Value ≥ 0	Default is 0 - non-pipelined

1.3.3.3 Function

Result = (**DataA** * **DataB**) + **Sum**

The LSB of the product of **DataA** and **DataB** is aligned with the LSB of **Sum.**

1.3.3.4 Example



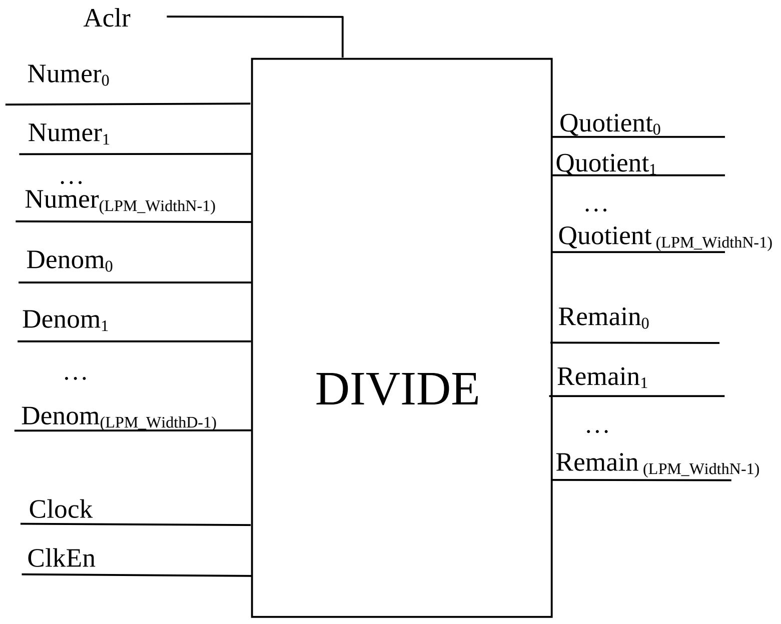
LPM_WidthA = 4
LPM_WidthB = 2
LPM_WidthS = 8
LPM_WidthP = 6
LPM_TYPE = MULTIPLIER

This diagram is for illustrative purposes only and is not intended to specify any implementation details.

					A3	A2	A1	A0
						×	B1	B0
		=	P5	P4	P3	P2	P1	P0
+	S7	S6	S5	S4	S3	S2	S1	S0
=	X7	X6	X5	X4	X3	X2	X1	X0
=	R5	R4	R3	R2	R1	R0		

The partial product is represented by P, and the full product by X. Both are internal only.

1.3.4 LPM_DIVDE


Aclr
Numer$_0$
Numer$_1$
…
Numer$_{(LPM\_WidthN-1)}$
Denom$_0$
Denom$_1$
…
Denom$_{(LPM\_WidthD-1)}$
Clock
ClkEn
DIVIDE
Quotient$_0$
Quotient$_1$
…
Quotient $_{(LPM\_WidthN-1)}$
Remain$_0$
Remain$_1$
…
Remain $_{(LPM\_WidthN-1)}$

1.3.4.1 Ports

Port Name	Type	Usage	Description	Comments
Numer	I	Required	Numerator	Vector, LPM_WidthN wide
Denom	I	Required	Denominator	Vector, LPM_WidthD wide. Note 1
Clock	I	Optional	Clock for pipelined usage	Note 2
ClkEn	I	Optional	Clock enable for pipelined	Note 3
Aclr	I	Optional	Asynchronous Clear	Note 4
Quotient	O	Note 5	Quotient	Vector, LPM_WidthN wide
Remain	O	Note 5	Remainder	Vector, LPM_WidthD wide

Note 1: The **Quotient** and **Remain** will be UNDEFINED if **Denom** value is 0.

Note 2: The **Clock** port provides for pipelined operation of the LPM_DIVIDE. If a lpm_pipeline other than 0 (default value) is specified, then the clock port must be connected.

Note 3: The **ClkEn** port provides a clock enable for pipelined operation.

Note 4: The pipeline initializes to undefined. The **Aclr** port may be used at any time to reset the pipeline to all 0's asynchronously to the clock.

Note 5: At lease one of **Quotient** and **Remain** ports must be used.

1.3.4.2 Properties

Property	Usage	Value	Comments
LPM_WidthN	Required	LPM Value > 0	Width of **Numer**
LPM_WidthD	Required	LPM Value > 0	Width of **Denom**
LPM_NRepresentation	Optional	UNSIGNED or SIGNED	Default is UNSIGNED.
LPM_DRepresentation	Optional	UNSIGNED or SIGNED	Default is UNSIGNED.
LPM_Pipeline	Optional	LPM Value ≥ 0	Default is 0 - non-pipelined

1.3.4.3 Function

Quotient = **Numerator / Denominator**

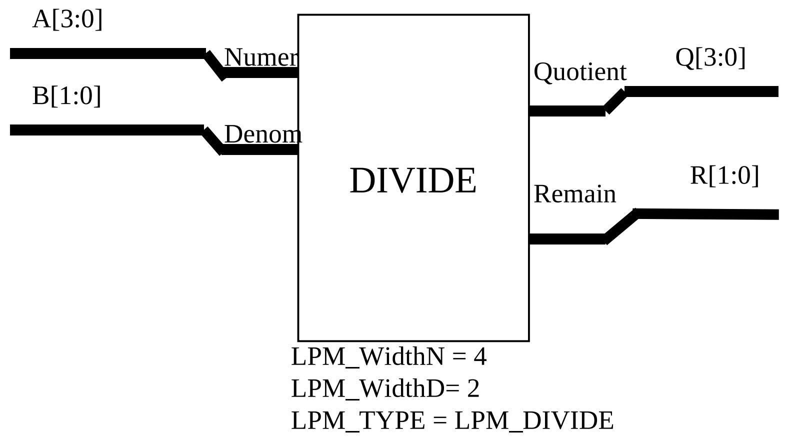
Numerator = Quotient * Denominator + Remainder

1.3.4.4 Examples:

Numerator	Denominator	Quotient	Remainder
+7	+3	+2	+1
-1	+3	-1	+2
-4	+3	-2	+2
-7	+3	-3	+2
+7	-3	-2	+1
+4	-3	-1	+1
-4	-3	+2	+2
-7	-3	+3	+2

In all cases, Remainder is always positive, while Quotient can be negative.

1.3.4.5 Example

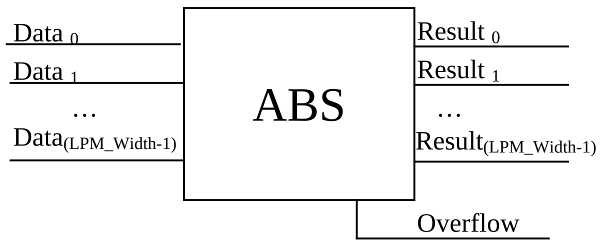


This diagram is for illustrative purposes only and is not intended to specify any implementation details.

	A3	A2	A1	A0
		/	B1	B0
=	Q3	Q2	Q1	Q0
			R1	R0

The quotient is represented by Q, and the remainder is represented by R.

1.3.5 LPM_ABS



1.3.5.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Vector represents SIGNED number	Vector, LPM_Width wide
Result	O	Required	Absolute Value of **Data**	Vector, LPM_Width wide.
Overflow	O	Optional	High (1) if Data = $-2^{LPM\_Width-1}$	Note 1

Note 1: Two's complement allows one more negative number than positive. The overflow port detects that singular instance and goes high to indicate that no positive equivalent exists.

1.3.5.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 1	Width of input and output vectors

1.3.5.3 Function

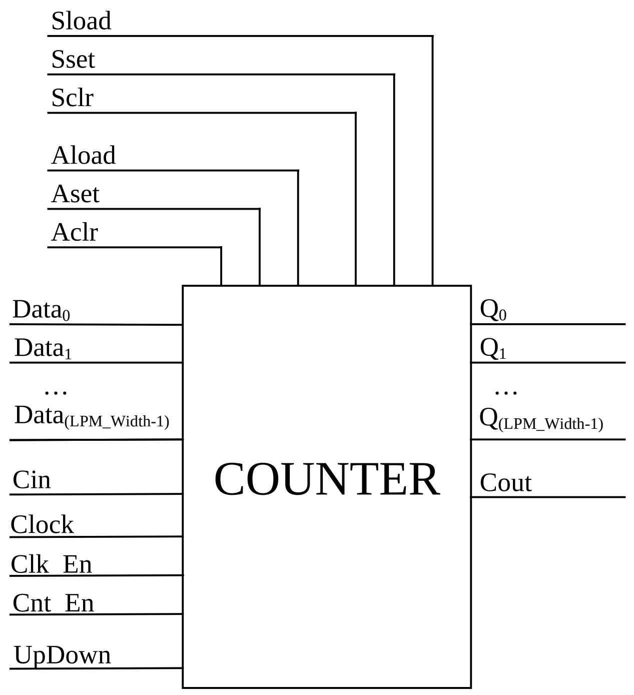
if Data = $-2^{LPM\_Width-1}$, then Overflow = 1, Result = UNDEFINED

else if Data < 0, then Result = (0 – Data)

else Result = Data

Data must always represent a SIGNED number and may be positive or negative. **Result** will always be positive.

1.3.6 LPM_COUNTER



1.3.6.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Optional	Parallel Data load for the counter	Vector, LPM_Width wide Uses Aload and/or Sload
Clock	I	Required	Positive Edge Triggered	
Clk_En	I	Optional	Enable all synchronous activities	Default is enabled (1)
Cnt_En	I	Optional	Disables count when low (0) (without affecting Sload, Sset, Sclr)	Default is enabled (1)
Cin	I	Optional	Carry in	

UpDown	I	Note 1	Controls direction of count
			High = 1 = count up
			Low = 0 = count down
				Default is Up (1)

Cout	O	Optional	Carry out port	Note 3
Q	O	Note 2	Count output.	Vector, LPM_Width wide
Sload	I	Optional	Load the counter with Data on the next clock.	Note 4
Sset	I	Optional	Set counter value to all 1's or to the value of LPM_Svalue, if present	Note 5, Note 6
Sclr	I	Optional	Clear the counter (set to all 0's)	Note 6
Aload	I	Optional	Load the counter with Data.	Note 4

Port Name	Type	Usage	Description	Comments
Aset	I	Optional	Set counter value to all 1's or to the value of LPM_Avalue, if present.	Note 5, Note 6
Aclr	I	Optional	Clear the counter (set to all 0's)	Note 6

Note 1: If the LPM_Direction property is used, then the **UpDown** port cannot be connected. If the LPM_Direction property is not used, then the **UpDown** port is optional.

Note 2: Either **Q** or **Cout** ports must be connected.

Note 3: Since the counter goes through **C** counts where $0 \leq \mathbf{C} <$ Modulus. Modulus is either the value specified by LPM_Modulus if present, or $2^{\text{LPM\_Width}}$. The **Cout** ports are optional and generally will be LPM_Modulus-1 which is the terminal count.

Note 4: If **Aload** and/or **Sload** are used, then the **Data** port must be connected.

Note 5: **Sset** and **Aset** will set the count to the value of LPM_Svalue or LPM_Avalue respectively, if those values are present. If no LPM_Svalue is specified, then **Sset** will set the count to all ones, likewise **Aset**.

Note 6: For outputs such as **Q** and **Cout** on the LPM_COUNTER, Aset, Aclr, Sset and Sclr affect the output **before** polarity is applied.

1.3.6.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors. If no output vectors are specified, then this is the number of bits in the count.
LPM_Modulus	Optional	LPM Value > 0	The maximum count. plus one
LPM_Direction	Optional	UP | DOWN	Note 1
LPM_Avalue	Optional	LPM Value	Loaded when Aset is active (1) Note 2
LPM_Svalue	Optional	LPM Value	Loaded when Sset is active (1) Note 2
LPM_Pvalue	Optional	LPM Value	Loaded at power on. Note 2

Note 1: If the LPM_Direction property is used, then the **UpDown** port cannot be connected. This property allows implementation of a Down counter as the default when the **UpDown** port is not connected.

Note 2: If the value specified is larger than the Modulus, then the behavior of the counter is UNDEFINED. The Modulus is the LPM_Modulus, if present, or else 2^{LPM_Width} .

The LPM_Counter defaults to an unsigned binary counter. The LPM_Hint property can be used to suggest an implementation style other than unsigned binary.

It is suggested, but not required, that all of the following styles be supported in the fitting tool and in simulation.

Unsigned binary	Signed Binary	BCD
standard Gray Code	Johnson	LFSR

1.3.6.3 Functions

Aclr Aset Aload	Sclr Sset Sload	Clock	Cnt_En	Clk_En	Up-Down	Output
H	L	X	X	X	X	Asynchronous value. Note 1
L	H	↑	X	H	X	Synchronous value Note 2
L	H	↑	X	L	X	No change
H	H	X	X	X	X	UNDEFINED
L	L	↑	H	L	X	No change
L	L	↑	H	H	H	Previous output + 1
L	L	↑	H	H	L	Previous output – 1
L	L	↑	H	H	U	Note 3

Note 1: The asynchronous value is determined by which asynchronous port is high: **Aclr**, **Aset** or **Aload**. If **Aclr** and **Aset** are both high, then the output is UNDEFINED. **Aclr** or **Aset** takes priority over **Aload**. Asynchronous controls have priority over synchronous controls. If the LPM_Avalue property is defined, then the **Aset** port, when active, will set the count to the value of the LPM_Avalue..

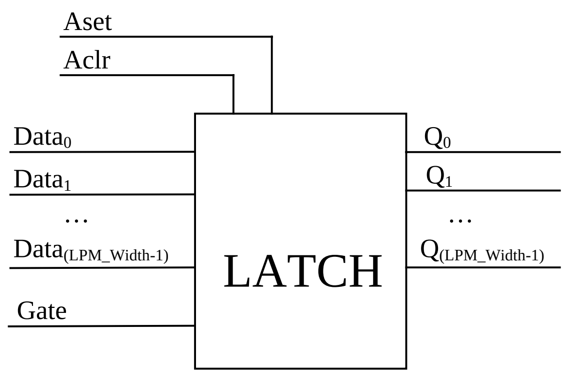
Note 2: The synchronous value is determined by which synchronous port is high: **Sclr**, **Sset** or **Sload**. If more then one synchronous port is high, then **Sclr** takes priority over **Sset** which takes priority over **Sload**. Asynchronous controls have priority over synchronous controls. If the LPM_Svalue property is defined, then the **Sset** port, when active, will set the count to the value of the LPM_Svalue..

Note 3: If the **UpDown** port is not connected, then the LPM_Direction property, if present, will determine the direction of the count. The LPM_Direction property defaults to UP. The **UpDown** and the LPM_Direction property are mutually exclusive; if one is used, then using the other is an ERROR.

1.4 STORAGE COMPONENTS

1.4.1 LPM_LATCH

D-Type Latch.



1.4.1.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Optional	Data Input to D-Type Latches	Vector, LPM_Width wide Note 1
Gate	I	Required	Latch enable input High (1) = flow through Low (0) = latch	
Q	O	Required	Data output from D-type latches	Vector, LPM_Width wide
Aset	I	Optional	Set latch value to all 1's or to the value of LPM_Avalue, if present.	Note 2, Note 3
Aclr	I	Optional	Clear the latch (set to all 0's)	Note 3

Note 1: If the **Data** input is not used, then either **Aset** or **Aclr** must be used.

Note 2: **Aset** will set the count to the value of LPM_Avalue, if that value is present. If no LPM_Avalue is specified, then **Aset** will set the count to all ones.

Note 3: **Aset** and **Aclr** affect the output (**Q_i**) values **before** the application of polarity to the ports.

1.4.1.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors
LPM_Avalue	Optional	LPM Value	Value loaded by **Aset**
LPM_Pvalue	Optional	LPM Value	Value loaded at power-on

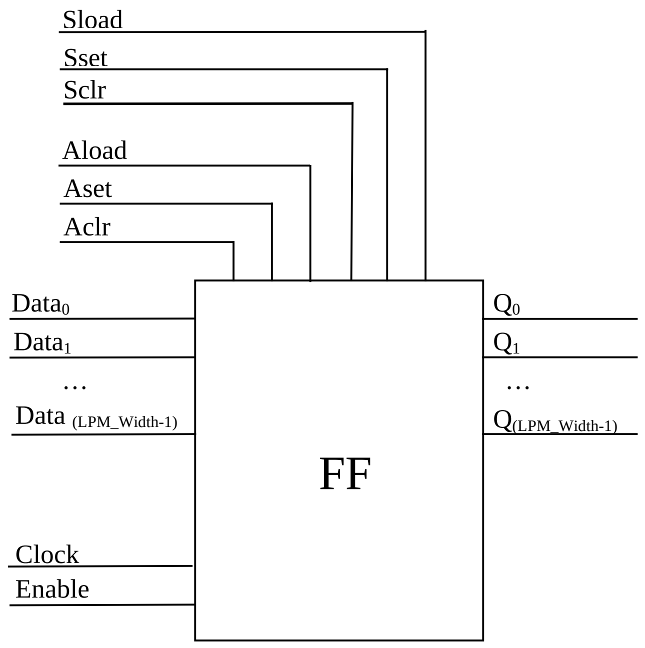
1.4.1.3 Functions

Aclr Aset	Gate	Output
H	X	Asynchronous value. Note 1
L	L	Latch holds current value (latched)
L	H	Latch is transparent (flow-through)

Note 1: The asynchronous value is determined by which asynchronous port is high: **Aclr** or **Aset**. If both asynchronous ports are high, then the output is UNDEFINED. If the LPM_Avalue property is defined, then the **Aset** port, when active, will set the count to the value of the LPM_Avalue.

1.4.2 LPM_FF

Flip-flop: D type or Toggle



1.4.2.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	TFF: Toggle enable DFF: Data input Data input during **Aload** or **Sload**	Vector, LPM_Width wide
Clock	I	Required	Positive Edge Triggered	
Enable	I	Optional	Enable all synchronous activities	Default is enabled (1)
Q	O	Required	Output of Flip-flops	Vector, LPM_Width wide
Sload	I	Note 1	TFF only: Load the Flip-flops with **Data** on the next clock.	Note 2, Note 4
Sset	I	Optional	Set Flip-flops to all 1's or to the value of LPM_Svalue, if present	Note 3, Note 4
Sclr	I	Optional	Clear the Flip-flops (set to all 0's)	Note 4
Aload	I	Note 1	TFF only: Load the Flip-flops with **Data**.	Note 4
Aset	I	Optional	Set Flip-flops to all 1's or to the value of LPM_Avalue, if present.	Note 3, Note 4
Aclr	I	Optional	Clear the Flip-flops (set to all 0's)	Note 4

Note 1: **Aload** and **Sload** are only applicable when LPM _FFType is TFF. If the LPM_FFType is DFF and these ports are connected, it is an ERROR.

Note 2: Synchronous load of LPM_TFF. For load operation **Sload** must be high (1) and **Enable** (the clock enable) must be High or unconnected.

Note 3: **Sset** and **Aset** will set the Flip-flops to the value of LPM_Svalue or LPM_Avalue repectively, if those values are present. If no LPM_Svalue is specified, then **Sset** will set the Flip-flops to all ones, likewise **Aset**.

Note 4: For outputs such as **Q_i** on the LPM_FF, **Aload**, **Aset**, **Aclr**, **Sload**, **Sset** and **Sclr** affect the output **before** polarity is applied.

1.4.2.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors
LPM_Avalue	Optional	LPM Value	Value loaded by **Aset**
LPM_Svalue	Optional	LPM Value	Value loaded by **Sset**
LPM_Pvalue	Optional	LPM Value	Value loaded at power-on
LPM_FFType	Optional	DFF | TFF	Default is DFF

1.4.2.3 Functions

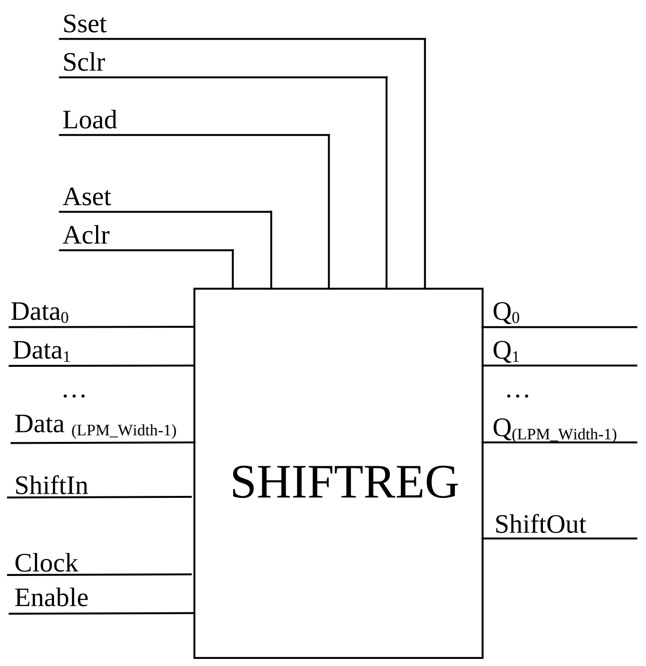
Aclr Aset Aload	Sclr Sset Sload	Clock	Enable	Output
H	X	X	X	Asynchronous value. Note 1
L	H	↑	H	Synchronous value Note 2
L	H	↑	L	No change
L	L	↑	L	No change (clock not enabled)
L	L	↑	H	TFF: FF_i is toggled if **$Data_i$** is high (1). DFF: **Data** is loaded into the register

Note 1: The asynchronous value is determined by which asynchronous port is high: **Aclr**, **Aset** or **Aload**. If **Aclr** and **Aset** are both high, then the output is UNDEFINED. **Aclr** or **Aset** takes priority over **Aload**. Asynchronous controls have priority over synchronous controls. If the LPM_Avalue property is defined, then the **Aset** port, when active, will set the FFs to the value of the LPM_Avalue. **Aload** is not permitted when the LPM_FFType is DFF.

Note 2: The synchronous value is determined by which synchronous port is high: Sclr, Sset or Sload. If more then one synchronous port is high, then Sclr takes priority over Sset which takes priority over Sload. Asynchronous controls have priority over synchronous controls. If the LPM_Svalue property is defined, then the Sset port, when active, will set the FFs to the value of the LPM_Svalue. Sload is not permitted when the LPM_FFType is DFF.

1.4.3 LPM_SHIFTREG

Universal Shift Register



1.4.3.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Note 1	Data for parallel load of shift register	Vector, LPM_Width wide
Clock	I	Required	Clock, positive edge triggered	
Enable	I	Optional	Clock enable input	Default is enabled (High)
ShiftIn	I	Note 1	Input for serial data during shift	
Load	I	Optional	High (1): Load operation Low (0): Shift operation	Default is low (0) - shift operation. Note 2.
Q	O	Note 3	Output for parallel data	Vector, LPM_Width wide
ShiftOut	O	Note 3	Output for serial data during shift	
Aset	I	Note 1	Set register value to all 1's or to the value of LPM_Avalue, if present.	Note 4, Note 5
Aclr	I	Note 1	Clear the register (set to all 0's)	Note 5
Sset	I	Note 1	Set register value to all 1's or to the value of LPM_Svalue, if present	Note 4, Note 5
Sclr	I	Note 1	Clear the register (set to all 0's)	Note 5

Note 1: At least one of **Data**, **Aset**, **Aclr, Sset, Sclr** and/or **ShiftIn** must be used.

Note 2: Synchronous parallel load. For parallel load operation **Load** must be high (1) and **Enable** (the clock enable) must be High or unconnected.

Note 3: Either **ShiftOut** or **Q** or both must be used.

Note 4: **Sset** and **Aset** will set the count to the value of LPM_Svalue or LPM_Avalue respectively, if those values are present. If no LPM_Svalue is specified, then **Sset** will set the count to all ones, likewise **Aset**.

Note 5: **Sset, Sclr, Aset** and **Aclr** affect the output ($\mathbf{Q_i}$) values **before** the application of polarity to the ports.

1.4.3.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors
LPM_Avalue	Optional	LPM Value	Value loaded by **Aset**
LPM_Svalue	Optional	LPM Value	Value loaded by **Sset**
LPM_Pvalue	Optional	LPM Value	Value loaded at power-on
LPM_Direction	Optional	LEFT|RIGHT	Default is LEFT. Note 1.

Note 1: A left shift implies that the data is being shifted into the LSB and out the MSB. The LSB gets the value on the **ShiftIn** port. The **ShiftOut** port is always equal to $\mathbf{Q_{LPM\_Width-1}}$.

1.4.3.3 Functions

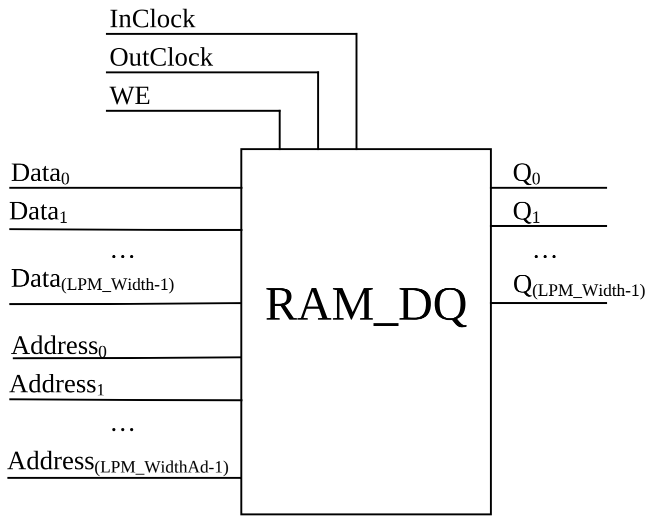
Aclr Aset	Sclr Sset	Clock	Enable	Load	Output
H	X	X	X	X	Asynchronous value. Note 1
L	H	↑	H	X	Synchronous value Note 2
L	H	↑	L	X	No change (clock not enabled)
L	L	↑	L	X	No change (clock not enabled)
L	L	↑	H	L	Parallel load Register from **Data**
L	L	↑	H	H	Q_i is shifted into Q_{i+1} ShiftIn is loaded into Q_0

Note 1: The asynchronous value is determined by which asynchronous port is high: **Aclr** or **Aset.** If **Aclr** and **Aset** are both high, then the output is UNDEFINED. Asynchronous controls have priority over synchronous controls. If the LPM_Avalue property is defined, then the **Aset** port, when active, will set the FFs to the value of the LPM_Avalue.

Note 2: The synchronous value is determined by which synchronous port is high: **Sclr** or **Sset**. If more then one synchronous port is high, then **Sclr** takes priority over **Sset**. Asynchronous controls have priority over synchronous controls. If the LPM_Svalue property is defined, then the **Sset** port, when active, will set the **Q** to the value of the LPM_Svalue.

1.4.4 LPM_RAM_DQ

Memory with separate input and output ports.



1.4.4.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input to memory	Vector, LPM_Width wide
Address	I	Required	Address of memory location	Vector, LPM_WidthAd wide
Q	O	Required	Output of memory	Vector, LPM_Width wide
InClock	I	Optional	Clock for read operation	Note 1
OutClock	I	Optional	Clock for write operation	Note 2
WE	I	Required	Write enable control. Enables write to the memory when high (1).	Note 3

Note 1: If the **InClock** port is used, then the **WE** port acts as an enable for write operations synchronized to the positive going edge of the signal on the **InClock** port. If the **InClock** ports is not used, then the **WE** port acts as an enable for write operations asynchronously.

Note 2: The addressed memory content → **Q** response is synchronous when the **OutClock** port is connected. and asynchronous when it is not connected.

Note 3: If only **WE** is used, the data on the **Address** port should not change while **WE** is active (high, 1). If the data on the **Address** port changes while **WE** is high (1), then all memory locations that are addressed are over-written with **Data**.

1.4.4.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors.
LPM_WidthAd	Required	LPM Value > 0	Width of Address Port. Note 1.
LPM_NumWords	Optional	LPM Value > 0	Number of words stored in Memory. Note 2.
LPM_InData	Optional	REGISTERED | UNREGISTERED	Indicates if **Data** port is registered. Default is REGISTERED
LPM_Address_Control	Optional	REGISTERED | UNREGISTERED	Indicates if **Address** and **WE** ports are registered. Default is REGISTERED
LPM_OutData	Optional	REGISTERED | UNREGISTERED	Indicates if **Q** port is registered. Default is REGISTERED
LPM_File	Optional	File Name	File for RAM initialization.

Note 1: The LPM_WidthAd should be (but is not required to be) equal to: $\lceil \log_2(\text{LPM\_NumWords}) \rceil$. If LPM_WidthAd is too small, some memory locations will not be addressable. If it is too big, then the addresses that are too high will return UNDEFINED.

Note 2: If LPM_NumWords is not used, then it defaults to $2^{\text{LPM\_WidthAd}}$. In general, this value should be (but is not required to be): $2^{\text{LPM\_WidthAd-1}} < \text{LPM\_NumWords} \leq 2^{\text{LPM\_WidthAd}}$.

1.4.4.3 Functions

Random Access Memory. This module can represent asynchronous memory or memory with synchronous inputs and/or outputs.

1.4.4.3.1 Synchronous Memory Operations

Synchronous Write to memoryInClock	WE	Memory Contents
X	L	No change
not ↑	H	No change (requires positive going clock edge)
↑	H	The memory location pointed to by **Address** is loaded with **Data**. Controlled by **WE**.

Synchronous Read from memory

OutClock	Output
not ↑	No Change
↑	The output register is loaded with the contents of the memory location pointed to by **Address**. **Q** outputs the contents of the output register. Note 1

Note 1: WE does not act as a clock enable for the output clock.

1.4.4.3.2 Asynchronous Memory Operations

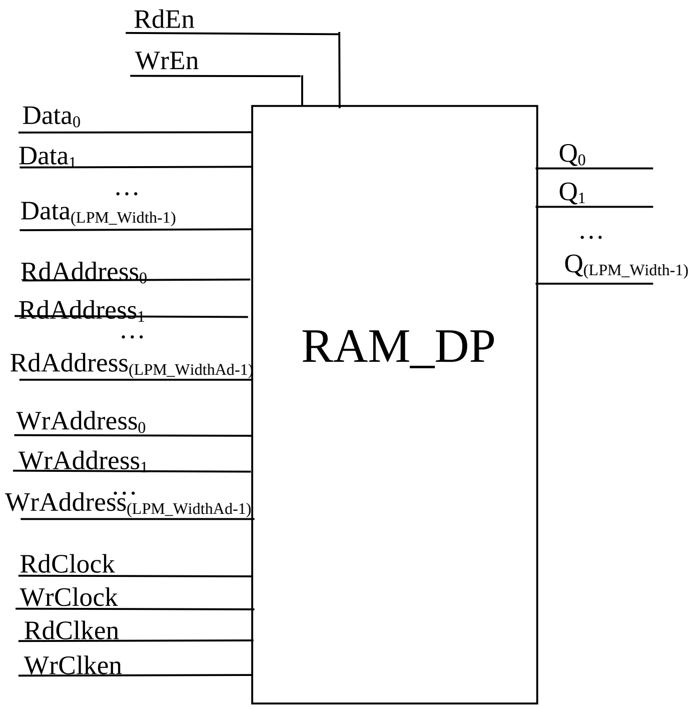
Totally asynchronous memory operations occur when neither **InClock** nor **OutClock** is connected.

WE	Memory Contents
L	No change
H	The memory location pointed to by **Address** is loaded with **Data**. Controlled by **WE**.

The output **Q** is asynchronous and reflects the data in the memory to which **Address** points.

1.4.5 LPM_RAM_DP

Dual-Port Random Access Memory



1.4.5.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input to memory	Vector, LPM_Width wide
RdAddress	I	Required	Read address of memory location	Vector, LPM_WidthAd wide
WrAddress	I	Required	Write address of memory location	Vector, LPM_WidthAd wide
RdClock	I	Optional	Clock for read operation	Note 1
WrClock	I	Optional	Clock for write operation	Note 2
RdClken	I	Optional	Read Clock enable control	Used with all registers clocked by RdClock.
WrClken	I	Optional	Write Clock enable control	Used with all registers clocked by WrClock. Note 3
RdEn	I	Optional	Read enable control	Note 4
WrEn	I	Required	Write enable control	Note 5
Q	O	Required	Output of memory	Vector, LPM_Width wide

Note 1: If the **RdClock** port is used, it acts as the clock for read operation and functions as the clock signal to any registers present on the **RdAddress**, **RdEn** and **Q** ports.

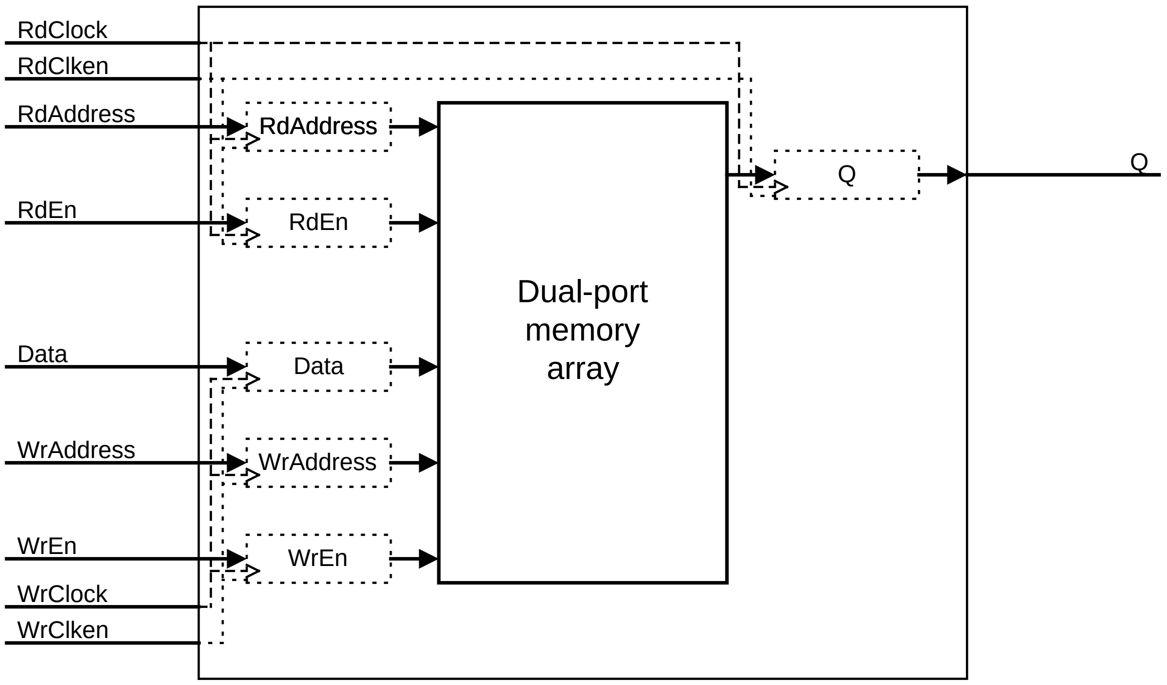
Note 2: If the **WrClock** port is used, it acts as the clock for write operation and functions as the clock signal to any registers present on the **WrAddress**, **WrEn** and **Data** ports. For single-clock synchronous design, user can tie **RdClock** and **WrClock** together.

Note 3: For single-clock synchronous design, user can tie **RdClken** and **WrClken** together

Note 4: If the **RdClock** port is used, then the **RdEn** port acts as an enable for read operations synchronized to the positive going edge of the signal on the **RdClock** port. If the **RdClock** port is not used, then the **RdEn** port acts as an enable for read operations asynchronously.

Note 5: If the **WrEn** port is registered, then writing of the data to the addressed is synchronous to the positive going edge of the signal on **WrClock** when **WrEn** is active. If the **WrEn** port is not registered, then the **WrEn** port acts as an enable for write operations asynchronously.

1.4.5.2 Functional diagram



The functional diagram helps to picture the relational between ports and functions.

1.4.5.3 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors.
LPM_WidthAd	Required	LPM Value > 0	Width of Address Port. Note 1.
LPM_NumWords	Optional	LPM Value > 0	Number of words stored in Memory. Note 2.
LPM_InData	Optional	REGISTERED | UNREGISTERED	Indicates if **Data** port is registered. Default is REGISTERED
LPM_OutData	Optional	REGISTERED | UNREGISTERED	Indicates if **Q** port is registered. Default is REGISTERED
LPM_RdAddress_Control	Optional	REGISTERED | UNREGISTERED	Indicates if **RdAddress** and **RdEn** ports are registered. Default is REGISTERED
LPM_WrAddress_Control	Optional	REGISTERED | UNREGISTERED	Indicates if **WrAddress** and **WrEn** ports are registered. Default is REGISTERED
LPM_File	Optional	File Name	File for RAM initialization.

Note 1: The LPM_WidthAd should be (but is not required to be) equal to: $\lceil \log_2(\text{LPM\_NumWords}) \rceil$. If LPM_WidthAd is too small, some memory locations will not be addressable. If it is too big, then the addresses that are too high will return UNDEFINED.

Note 2: If LPM_NumWords is not used, then it defaults to $2^{\text{LPM\_WidthAd}}$. In general, this value should be (but is not required to be): $2^{\text{LPM\_WidthAd-1}} < \text{LPM\_NumWords} \leq 2^{\text{LPM\_WidthAd}}$.

1.4.5.4 Functions

Random Access Memory. This module can represent asynchronous memory or memory with synchronous inputs and/or outputs.

1.4.5.4.1 Synchronous Memory Operations

Synchronous Write to memory (all inputs registered)

WrClock	WrClken	WrEn	Memory Contents
X	L	L	No change
not ↑	H	H	No change
↑	L	X	No change
↑	H	H	The memory location pointed to by **WrAddress** is loaded with **Data**.

1.4.5.4.2 Synchronous Read from memory

RdClock	RdClken	RdEn	Output
X	L	L	No Change
not ↑	H	H	No Change
↑	L	X	No Change
↑	H	H	**Q** outputs the contents of the memory location.

1.4.5.4.3 Asynchronous Memory Operations

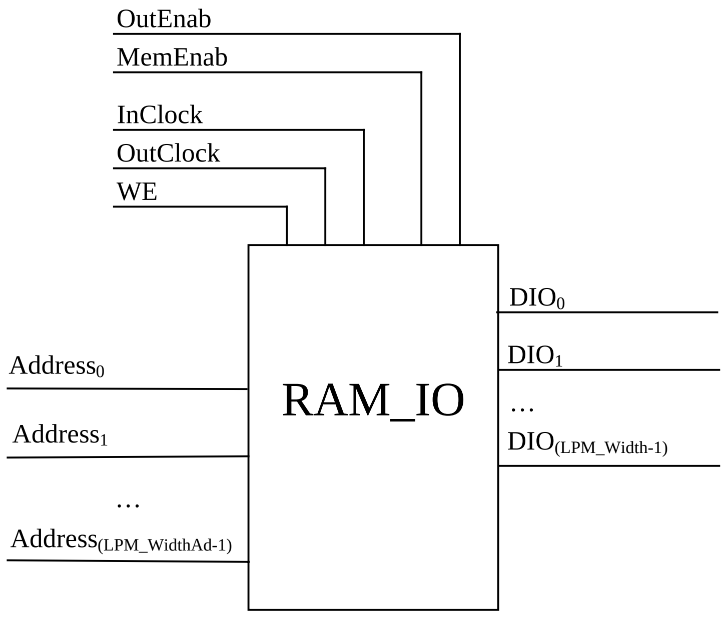
Totally asynchronous memory operations occur when neither **RdClock** nor **WrClock** is connected.

WrEn	Memory Contents
L	No change
H	The memory location pointed to by **WrAddress** is loaded with **Data**. Controlled by **WrEn**.

The output **Q** is asynchronous and reflects the data in the memory to which **RdAddress** points.

1.4.6 LPM_RAM_IO

Memory with a single I/O port.



1.4.6.1 Ports

Port Name	Type	Usage	Description	Comments
Address	I	Required	Address of memory location	Vector, LPM_WidthAd wide
InClock	I	Note 1	Synchronous load of memory	
OutClock	I	Note 2	Synchronous **Q** outputs from memory.	
MemEnab	I	Optional	Memory Output Tristate Enable	Note 3
OutEnab	I	Optional	High (1): DIO ← Memory[Address] Low (0): Memory[Address] ← DIO	Note 4
WE	I	Required	Write enable control. Enables write to the memory when high (1).	Note 5
DIO	I/O	Required	bi-directional Data port	Vector, LPM_Width wide

Note 1: If the **InClock** port is used, then the **WE** port acts as an enable for write operations synchronized to the positive going edge of the signal on the **InClock** port. If the **InClock** ports is not used, then the **WE** port acts as an enable for write operations asynchronously.

Note 2: The addressed memory content → **Q** response is synchronous when the **OutClock** port is connected. and asynchronous when it is not connected.

Note 3: When low, the memory is inactive and the outputs are Hi-Z. This also disables the ability to write to memory.

Note 4: Same as **~WE.** Only one of **OutEnab** or **WE** should be used.

Note 5: Same as **~OutEnab**. Only one of **WE** or **OutEnab** should be used. If no clock ports are used, when **WE** is active (high, 1) the data on the **Address** port should not change. If the data on the **Address** port changes while **WE** is high (1), then all memory locations that are addressed are over-written with **Data**.

1.4.6.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors.
LPM_WidthAd	Required	LPM Value > 0	Width of Address Port. Note 1.
LPM_NumWords	Optional	LPM Value > 0	Number of words stored in Memory. Note 2.
LPM_InData	Optional	REGISTERED | UNREGISTERED	Indicates if **Data** port is registered. Default is REGISTERED
LPM_Address_Control	Optional	REGISTERED | UNREGISTERED	Indicates if **Address, MemEnab** and **WE** ports are registered. Default is REGISTERED
LPM_OutData	Optional	REGISTERED | UNREGISTERED	Indicates if **Q** port is registered. Default is REGISTERED
LPM_File	Optional	File Name	File for RAM initialization.

Note 1: The LPM_WidthAd should be (but is not required to be) equal to: $\lceil \log_2(\text{LPM\_NumWords}) \rceil$. If LPM_WidthAd is too small, some memory locations will not be addressable. If is too big, then the addresses that are too high will return UNDEFINED.

Note 2: If LPM_NumWords is not used, then it defaults to $2^{\text{LPM\_WidthAd}}$. In general, this value should be (but is not required to be): $2^{\text{LPM\_WidthAd-1}} <$ LPM_NumWords $\leq 2^{\text{LPM\_WidthAd}}$.

1.4.6.3 Functions

Random Access Memory. This module can represent asynchronous memory or memory with synchronous inputs and/or outputs**.**

1.4.6.3.1 Synchronous Memory Operations

Synchronous Write to memory

InClock	Mem-Enab	WE or ~OutEnab	Memory contents
X	L	X	Hi-Z (memory not enabled)
↑	H	L	No change (no write enable)
not ↑	H	H	No change (requires positive going clock edge)
↑	H	H	Memory[**Address**] ← **DIO** controlled by **WE**

Synchronous Read from memory

OutClock	Mem-Enab	Out-Enab or ~WE	Output (Note 1)
X	L	X	Hi-Z (memory not enabled)
↑	H	L	DIO acts as an input to the LPM_RAM_IO.
↑	H	H	The output register is loaded with the contents of the memory location pointed to by **Address**. **DIO** outputs the contents of the output register. Note 1
not ↑	H	H	No change. **DIO** is held constant until next clock. Data will change on next **OutClock**.

Note 1: WE does not act as a clock enable for the output clock.

1.4.6.3.2 Asynchronous Memory Operations

Totally asynchronous memory operations occur when neither **InClock** nor **OutClock** is connected.

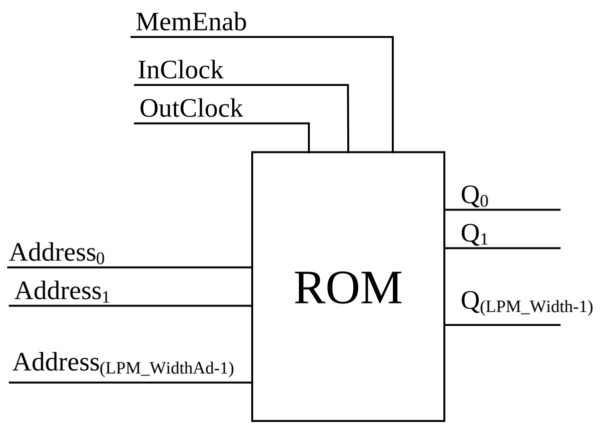
Mem-Enab	WE or ~OutEnab	Memory Contents (Note 1)
L	X	Hi-Z (memory not enabled)
H	L	No change (No Write Enable)
H	H	The memory location pointed to by Address is loaded with Data on **DIO**. Note 2

Note 1: When neither **InClock** nor **OutClock** is connected, the output **DIO** is asynchronous and reflects the data in the memory to which **Address** points when **DIO** is acting as an output.

Note 2: The data on the **Address** port should not change while **WE** is high (**OutEnab** is low). If the data on the **Address** port changes while **WE** is high (**OutEnab** is low), then all memory locations that are addressed are over-written with **DIO**.

1.4.7 LPM_ROM

Read only memory



1.4.7.1 Ports

Port Name	Type	Usage	Description	Comments
Address	I	Required	Address of memory location	Vector, LPM_WidthAd wide
InClock	I	Note 1	Synchronous **Address**	
OutClock	I	Note 2	Synchronous **Q** outputs from memory.	
MemEnab	I	Optional	Memory enable control.	Low: **Q** output is Hi-Z High: **Q** is Memory[**Address**]
Q	O	Required	Output of memory	Vector, LPM_Width wide

Note 1: The **Address** is synchronous(registered) when the **InClock** port is connected. and asynchronous(registered) when it is not connected

Note 2: The addressed memory content → **Q** response is synchronous when the **OutClock** port is connected. and asynchronous when it is not connected.

1.4.7.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output vectors.
LPM_WidthAd	Required	LPM Value > 0	Width of Address Port. Note 1.
LPM_NumWords	Optional	LPM Value > 0	Number of words stored in Memory. Note 2.
LPM_Address_Control	Optional	REGISTERED | UNREGISTERED	Indicates if **Addres** port is registered. Default is REGISTERED
LPM_OutData	Optional	REGISTERED | UNREGISTERED	Indicates if **Q** is registered. Default is REGISTERED
LPM_File	Required	File Name	File for ROM initialization.

Note 1: The LPM_WidthAd should be (but is not required to be) equal to: $\lceil \log_2(\text{LPM\_NumWords}) \rceil$. If LPM_WidthAd is too small, some memory locations will not be addressable. If is too big, then the addresses that are too high will return UNDEFINED.

Note 2: If LPM_NumWords is not used, then it defaults to $2^{\text{LPM\_WidthAd}}$. In general, this value should be (but is not required to be): $2^{\text{LPM\_WidthAd-1}} < \text{LPM\_NumWords} \leq 2^{\text{LPM\_WidthAd}}$.

1.4.7.3 Functions

Read Only Memory. This module can represent asynchronous memory or memory with synchronous outputs.

1.4.7.3.1 Synchronous Memory Operations

Synchronous memory

OutClock	MemEnab	Output
X	L	**Q** output is Hi-Z
not ↑	H	No Change in output
↑	H	The output register is loaded with the contents of the memory location pointed to by **Address**. **Q** outputs the contents of the output register.

1.4.7.3.2 Asynchronous Memory Operations

Totally asynchronous memory operations occur when none of **InClock** nor **OutClock** is connected.

MemEnab	Memory Contents
L	**Q** output is Hi-Z
H	The memory location pointed to by **Address** is read.

The output **Q** is asynchronous and reflects the data in the memory to which **Address** points.

1.4.7.4 ROM Contents

The format for the file containing the ROM contents is contained in section 11.4, HEX OBJECT FILE SPECIFICATION. A summary and examples is included here for reference only.

1.4.7.4.1 Glossary

Hex-byte: an 8-bit byte represented by a pair of hex-digits.

Hex-digit: a symbol representing values from 0 to 15 (4-bits) using the digits 0-9 and the letters A-F (or a-f).

Byte Count: A pair of hex-digits indicating the number of data or address hex-bytes in the current record. The following fields are not included in the Byte-count: address record, data indicator record or checksum.

Address Bytes: A pair of hex-bytes representing the address offset (with respect to the current Extended Address) of the first word in the data portion of the record. The value of the address bytes is added to the current Extended Address to form the real address of the first data word. An Extended Address is defined as zero until it is redefined by an Extended Address record.

Extended Address: The concatenation of the Data in an Extended Address Record with a hex '0'. See the example below.

Sum_Check: A hex-byte representing the sum of the bytes in the record, exclusive of the Sum_Check. The bytes are taken one hex-byte (two hex-digits) at a time and summed as unsigned integers. White space characters are ignored for the Sum_Check calculation. The two's complement of the sum is computed and the low-order hex-byte is retained. The hex-bytes that are included in the sum are: Byte_Count, Address, Record_Type ('00' or '02'), and the Data. The Sum_Check is not included in the calculation. (The term Sum_Check is used to avoid conflict with the EDIF definition Check_Sum.)

For example, if the record is : 07 0000 02 00 01 02 03 03 03 01 EA the bytes sum as 07+00+00+02+00+01+02+03+03+03+01=16. The 32-bit two's complement of 00000016 is FFFFFFEA. The least-significant byte has the hex value EA.

Interpretation of the data values in the Hex Object File depends on the value of the Width property. The data values correspond to the words in the LPM_ROM, with the word at address zero appearing first in the data list. The word at address one follows the word at address zero, etc. Each word in the Hex Object File is padded on the left by zero bits so that the word consumes an even multiple of 8 bits (a hex-byte). If the 'Width' is an even multiple of 8 bits, then no padding is needed. For example, suppose an LPM_ROM is 3-bits wide. To represent the data values 0 through 7 in such an LPM_ROM you would use the hex-bytes: 00 01 02 03 04 05 06 07.

The goal is readability: the number "1" would be represented as a hex-byte "01" and padding bits will be ignored so that subsequent words will also be readable. In a 10-bit wide LPM_ROM, two hex-byte pairs are used to represent each word. The first hex-byte value contains the two most-significant bits (bits 10 and 9). The second hex-byte value contains the eight least-significant bits (bits 7, 6, 5, 4, 3, 2, 1, and 0).

Example 1:

Width	Value	hex-byte
6	0	00
6	7	07
6	50	32
10	7	00 07
10	27	00 1B
10	273	01 11
10	725	02 D5
10	1023	03 FF

Example 2:
Hex file for an LPM_ROM (Width = 10, Numwords = 32)

line #	**Contents**
1	:02 0000 02 00 01 FB
2	:08 0000 00 00 07 00 1B 01 11 02 D5 ED
3	:04 0008 00 03 FF 00 01 F1
4	:00000001FF

Interpretation:

Line 4 contains ":00000001FF" which is the "End of File" indicator.

The first field of each of lines 1, 2, and 3 is the byte count (this excludes the first three fields and the last field, that is: byte count = # fields - 4).

The second field is an address offset from the current Extended Address. If none is specified, then the current Extended Address is used.

The third field indicates whether the following hex-bytes are a new Extended Address (02) or data (00).

The fields between the third and last fields contain either an address or data.

If the fields contain an address, then it is multiplied by 16 to form a new Extended Address. Note that the second field is required to be '0000' when an address is specified.

If the fields contain data, then hex-bytes are taken in groups to form the data words. If the 'Width' is between 1 and 8 then one hex-byte is read. If the 'Width is between 9 and 16, then two hex-bytes are taken. Between 17 and 24, three hex-bytes are taken, etc.

The last field is a checksum computed by summing all of the bytes (including the first three fields), truncating the result to the least significant hex-byte and taking the two's complement.

Corresponding contents of LPM_ROM:

Address (Hex)	**+0**	**+1**	**+2**	**+3**	**+4**	**+5**	**+6**	**+7**
0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
16 (10)	7	1B	111	2D5	0	0	0	0
24 (18)	3FF	1	0	0	0	0	0	0

Example 3:
An LPM_ROM with Width=2 and Numwords=7 (7 words of 2-bits each)

```
:07    0000      00      00      01      02      03      03      03      01      EA
:00    0000      01      FF
```

Corresponding contents of LPM_ROM:

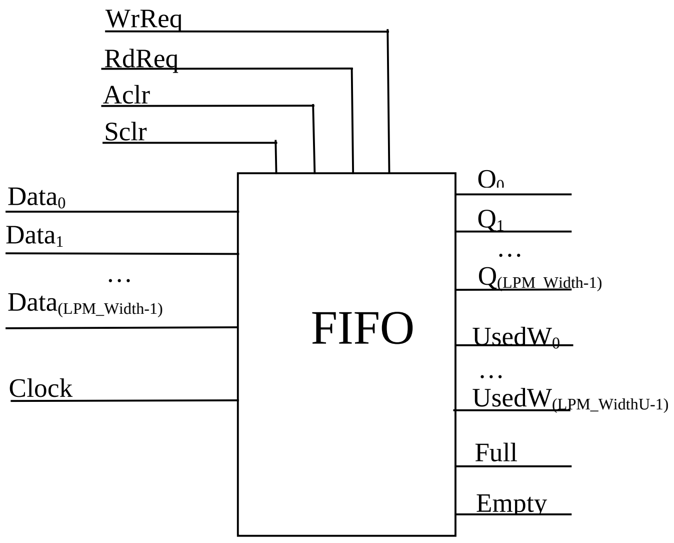
Address (Hex)	+0	+1	+2	+3	+4	+5	+6+7
0	0	1	2	3	3	3	1

1.4.7.4.2 additional comments on Hex File Format

There is optional white space between the all of the fields. The white space does not affect the checksum.

1.4.8 LPM_FIFO

Single-Clock First-In-First-Out Memory.



1.4.8.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input to memory stack	Vector, LPM_Width wide
Clock	I	Required	Clock to memory	
Aclr	I	Optional	Asynchronous Clear of memory pointer	Empties the FIFO asynchronously
Sclr	I	Optional	Synchronous Clear of memory pointer	Empties the FIFO synchronously
RdReq	I	Required	Read request control	Disabled if **Empty** = 1
WrReq	I	Required	Write request control	Disabled if **Full** = 1
Full	O	Optional	Full flag when memory is full	
Empty	O	Optional	Empty flag when memory is empty	
UsedW	O	Optional	Number of words used in the memory	Vector, Note 1
Q	O	Required	Output of memory	Vector, LPM_Width wide

Note 1: The width of UsedW should be equal to a round up integer value of $\lceil \log_2(\text{LPM\_NumWords}) \rceil$.

1.4.8.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output data vectors.
LPM_WidthU	Note 1	LPM Value > 0	Width of UsedW
LPM_NumWords	Required	LPM Value > 0	Number of words of the memory. Note 2.
LPM_ShowAhead	Optional	ON | OFF	Date will be available immediately for read. Note 3. Default is OFF

Note 1: Required if **UsedW** is used.

Note 2: LPM_NumWords is the size of the memory.

Note 3: LPM_ShowAhead allows user to read the data immediately after data is the memory without asserting **RdReq** explicitly. **RdReq** effectively acts as a read acknowledge. LPM_ShowAhead does not affect the read pointer.

1.4.8.3 Functions

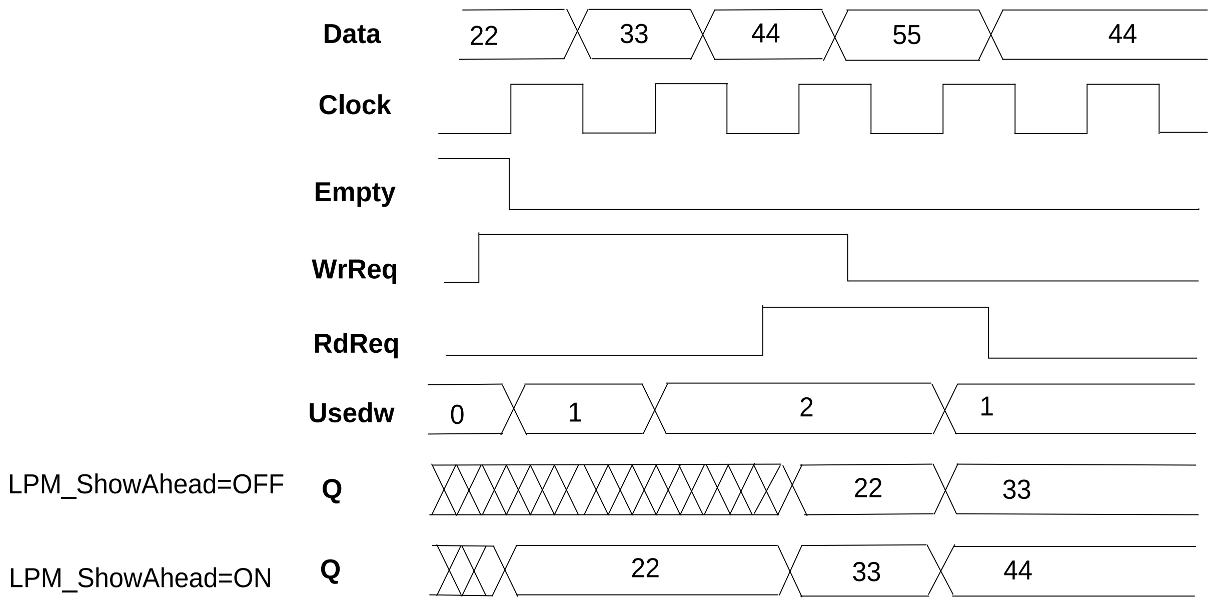
First-In-First-Out Memory. This module can represent memory with synchronous inputs and outputs.

1.4.8.3.1 Synchronous Memory Operations

Clock	RdReq	WrReq	Memory Contents
X	L	L	No change
not ↑	X	X	No change (requires positive going clock edge)
↑	L	H	Write **Data** to memory.
↑	H	L	Read memory and update **Q**
↑	H	H	Write **Data** to memory and read memory to **Q.** Note 1.

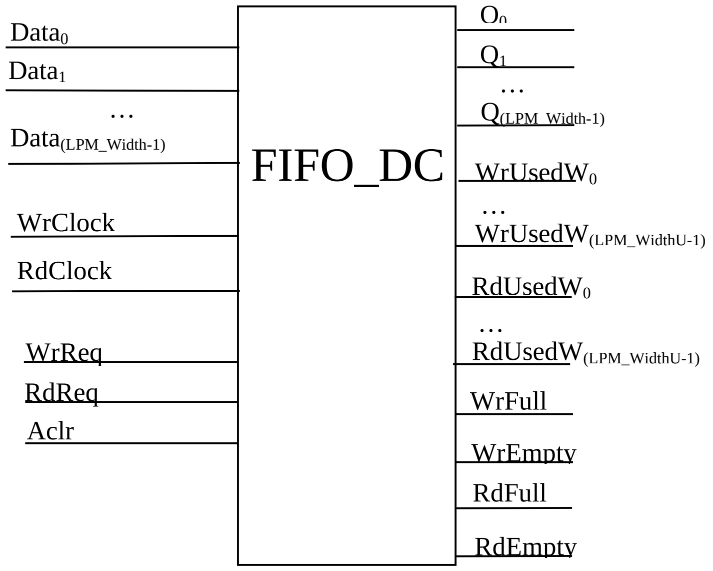
Note 1: When FIFO is full then **WrReq** will be ignored and **RdReq** is executed. When FIFO is empty then **RdReq** will be ignored and WrReq will be executed.

1.4.8.4 Example



1.4.9 LPM_FIFO_DC

Dual-Clock First-In-First-Out Memory.



1.4.9.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input to memory stack	Vector, LPM_Width wide
RdClock	I	Required	Clock for memory read	
WrClock	I	Required	Clock for memtory write	
Aclr	I	Optional	Asynchronous Clear of memory pointer	Empties the FIFO
RdReq	I	Required	Read request control	Disalbed if **RdEmpty** = 1
WrReq	I	Required	Write request control	Disabled if **WrFull** = 1
RdFull	O	Optional	Full flag when memory is full	Synchronous with RdClock
WrFull	O	Optional	Full flag when memory is full	Synchronous with WrClock
RdEmpty	O	Optional	Empty flag when memory is empty	Synchronous with RdClock
WrEmpty	O	Optional	Empty flag when memory is empty	Synchronous with WrClock
RdUsedW	O	Optional	Number of words in the FIFO	Vector, Note 1
WrUsedW	O	Optional	Number of words in the FIFO	Vector, Note 1
Q	O	Required	Output of memory stack	Vector, LPM_Width wide

Note 1. The width of RdUsedW and WrUsedW should be equal to a round up integer value of $\lceil \log_2(LPM\_NumWords) \rceil$.

1.4.9.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input and output data vectors.
LPM_WidthU	Note 1	LPM Value > 0	Width of WrUsedW and RdUsedW
LPM_NumWords	Required	LPM Value > 0	Number of words of the memory. Note 2.
LPM_ShowAhead	Optional	ON | OFF	Data will be available immediately for read. Note 3. Default is OFF

Note 1: Required if WrUsedW or RdUsedW port is used.

Note 2: LPM_NumWords is the size of the memory.

Note 3: LPM_ShowAhead allows user to read the data immediately after data is the written to memory without asserting **RdReq** explicitly. **RdReq** effectively acts as a read acknowledge. LPM_ShowAhead does not affect the read pointer.

1.4.9.3 Functions

First-In-First-Out Memory. This module can represent memory with synchronous inputs and outputs.

1.4.9.3.1 Synchronous Memory Operations

RdClock	WrClock	RdReq	WrReq	Memory Contents
X	X	L	L	No change
not ↑	not ↑	X	X	No change (requires positive going clock edge)
↑	↑	L	H	Write **Data** to memory.
↑	↑	H	L	Read memory and update **Q.**
↑	↑	H	H	Write **Data** to memory and read memory to **Q.** Note 1.

Note 4: When FIFO is full then **WrReq** will be ignored and **RdReq** is executed. When FIFO is empty then **RdReq** will be ignored and **WrReq** will be executed.

1.5 TABLE PRIMITIVES

1.5.1 TABLE FORMATS

The full syntax of Truth Table files is defined in section 11.5. A summary is included here for reference only.

Logical Functions can be described by how they behave rather than how they are implemented. In LPM this manner of description is restricted to the two table-based modules LPM_TTABLE and LPM_FSM. The former describes a stateless behavior while the latter describes the behavior of systems with a memory of their behavior (i.e., with a state). The table-based digital functions LPM_TTABLE and LPM_FSM define their function by describing their outputs as a function of their inputs in the form of a table. A table in this format describes a function as a sum of products. The inputs to the product terms are the true and inverted inputs to the module. A typical product term of a table driven module with four inputs ($Data_{3:0}$) is:

$$\mathbf{Result = \sim Data_0 \; \& \; Data_1 \; \& \; Data_3}$$

Tables in LPM are represented in the Berkeley PLA format with an input plane and an output plane. The table entry corresponding to the above equation is:

1-10 1

The first four characters represent the input terms. Notice that the notation depends upon positional information so that an entry is needed even if the input is not used in the product term. Each position in the input plane corresponds to an input variable: a 1 implies the corresponding input literal is used in the product term, a 0 implies the complemented input literal appears in the product term, and ‘-’ implies the input literal does not appear in the product term. A sum of products term is expressed as two of more lines in the table. Each line in the table thus expresses a function for one or more of the outputs but the total function is expressed as the sum of all the lines in the table.

Boolean sums of terms can be described in various ways. One method is to describe the input conditions under which the outputs are 1 and imply that the outputs are 0 for all other sets of inputs (i.e., the ON-set can be provided). This may represent a more complex function than is needed because some of the outputs are not used under certain input conditions (i.e., they are “don’t cares”). A more precise way of defining the Boolean function then, is to provide both the ON-set and the DC-set (Don’t Care set). Then the OFF-set (the conditions under which the outputs must be low) is the complement of the union of the On- and DC-sets. LPM follows the Berkeley format in allowing the following different representations for truth tables:

1. By providing the ON-set, the OFF-set is implied as the complement of the ON-set and the DC-set is empty. (LPM_TruthType property = F)
2. By providing the ON-set and DC-set, the OFF-set is implied as the complement of the union of the ON-set and the DC-set. If any product term belongs to both the ON-set and the DC-set, then it is considered a Don’t Care and may be removed from the ON-set during the fitting process. (LPM_TruthType = FD)

3. By providing the ON-set and OFF-set the DC-set is implied as the complement of the union of the ON-set and OFF-set. It is an error for any product term to belong to both the ON-set and OFF-set. (LPM_TruthType = FR)
4. By providing the ON-set, the DC-set and the OFF-set the truth table is fully specified. (LPM_TruthType = FDR)

The definition of the output section of a Boolean function expressed as a table depends upon which of the four descriptions is used. The output formats are:

F = LPM_TruthType. For each output, a 1 means that this product term belongs to the ON-set, a 0 means that this product term has no meaning for the value of this function (i.e., the output value may be set by some other function). This type corresponds to an actual PLA where only the ON-set is actually implemented.

FD = LPM_TruthType. For each output, a 1 means that this product term belongs to the ON-set, a 0 means that this product term has no meaning for the value of this function, and a '-' implies that this product term belongs in the DC-set.

FR = LPM_TruthType. For each output, a 1 means that this product term belongs to the ON-set, a 0 means that this product term belongs to the OFF-set, and a '-' means that this product term has no meaning for the value of this function.

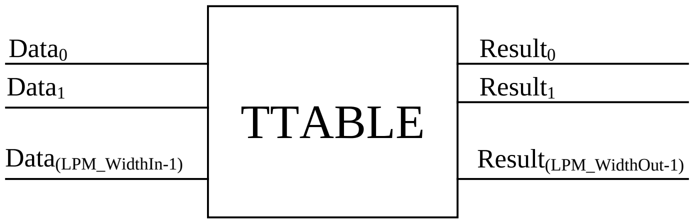
FDR = LPM_TruthType. For each output, a 1 means that this product term belongs to the ON-set, a 0 means that this product term belongs to the OFF-set, a '-' implies means that this product term belongs to the DC-set, and a '~' implies that this product term has no meaning for the value of this function.

Note 1: regardless of the type, a '~' implies the product term has no meaning for the value of the function.

Note 2: If at all possible, the fitter should be given the DC-set (either implicitly or explicitly) in order to improve the results of the fitting.

1.5.2 LPM_TTABLE

Truth Table



1.5.2.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input	Vector, LPM_WidthIn wide
Result	O	Required	Result of Logic Function	Vector, LPM_WidthOut wide

1.5.2.2 Properties

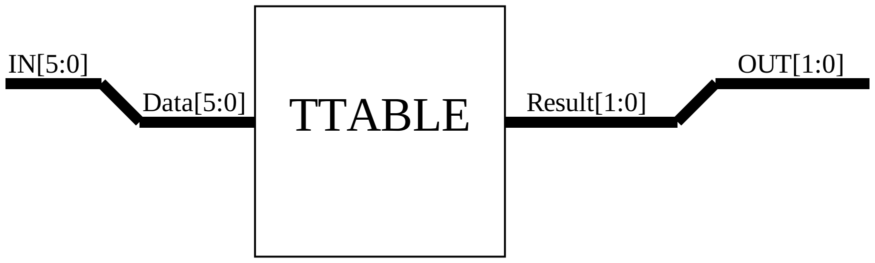
Property	Usage	Value	Comments
LPM_WidthIn	Required	LPM Value > 0	Width of input vector
LPM_WidthOut	Required	LPM Value > 0	Width of output vector
LPM_File	Required	String File Name	Name of file containing Truth Table file.
LPM_TruthType	Optional	F | FD | FR | FDR	Default is FD

1.5.2.3 Function

$$\mathbf{Result} = f(\mathbf{Data})$$

Where f is the function defined in the Truth Table file.

1.5.2.4 Example



LPM_WidthIn = 6
LPM_WidthOut = 2
LPM_File = TT1.txt
LPM_TruthType = FD
LPM_TYPE = TTABLE

This diagram is for illustrative purposes only and is not intended to specify any implementation details.

The file TT1.txt contains:

.i	6		# No. of inputs					
.o	2		# No. of outputs					
0	0	-	-	1	0		0	1
0	0	-	1	0	0		1	0
0	1	1	-	-	0		1	1

This corresponds to:

$\mathbf{Result_0 = {\sim}Data_0 \;\&\; Data_1 \;\&\; {\sim}Data_4 \;\&\; {\sim}Data_5 \mid {\sim}Data_0 \;\&\; Data_3 \;\&\; Data_4 \;\&\; {\sim}Data_5}$

$\mathbf{Result_1 = {\sim}Data_0 \;\&\; {\sim}Data_1 \;\&\; Data_2 \;\&\; {\sim}Data_4 \;\&\; {\sim}Data_5 \mid {\sim}Data_0 \;\&\; Data_3 \;\&\; Data_4 \;\&\; {\sim}Data_5}$

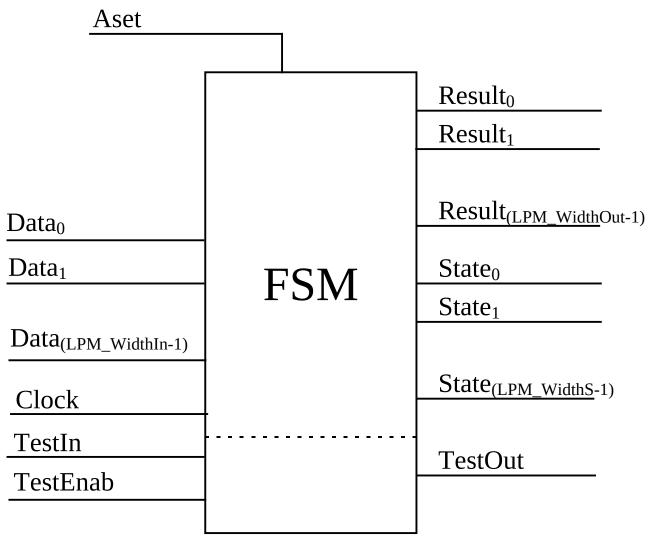
or

$\mathbf{OUT_0 = {\sim}IN_0 \;\&\; IN_1 \;\&\; {\sim}IN_4 \;\&\; {\sim}IN_5 \mid {\sim}IN_0 \;\&\; IN_3 \;\&\; IN_4 \;\&\; {\sim}IN_5}$

$\mathbf{OUT_1 = {\sim}IN_0 \;\&\; {\sim}IN_1 \;\&\; IN_2 \;\&\; {\sim}IN_4 \;\&\; {\sim}IN_5 \mid {\sim}IN_0 \;\&\; IN_3 \;\&\; IN_4 \;\&\; {\sim}IN_5}$

1.5.3 LPM_FSM

Finite State Machine



1.5.3.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data input	Vector, LPM_WidthIn wide
Clock	I	Required	Clock for State transitions	Positive edge triggered
State	O	Optional	Current State Vector	Vector, LPM_WidthS wide. Note 1
Result	O	Required	Result of Logic Function	Vector, LPM_WidthOut wide. Note 2
Aset	I	Optional	Asynchronous set control	Note 3
TestEnab	I	Note 4	Test clock enable input	
TestIn	I	Note 4	Serial test data input	
TestOut	O	Note 4	Serial test data output	TestOut = State$_{LPM\_WidthS-1}$

Note 1: The state vector is always present inside the FSM. It may be brought out if needed elsewhere in the design by using the **State** port.

Note 2: The **Result** vectors are asynchronous. The outputs may be purely a function of the internal state vector (a Moore machine) or may be a function of both the internal state vector and the **Data** inputs (a Mealy machine).

Note 3: **Aset** will set the count to the value of LPM_Avalue, if that value is present. If no LPM_Avalue is specified, then **Aset** will set the count to all ones. **Aset** affects the outputs (**Result** and **State**) values **before** the application of polarity to the ports.

Note 4: Either all of the Test ports must be connected or none of them.

1.5.3.2 Properties

Property	Usage	Value	Comments
LPM_WidthIn	Required	LPM Value > 0	Width of input vector
LPM_WidthOut	Required	LPM Value > 0	Width of output vector
LPM_WidthS	Optional	LPM Value > 0	Width of the State vector
LPM_File	Required	String File Name	Name of file containing Truth Table file.
LPM_Pvalue	Optional	LPM Value	Power-up value of State Vector
LPM_Avalue	Optional	LPM Value	Value of State Vector when Aset is asserted.
LPM_TruthType	Optional	F | FD | FR | FDR	Default is FD

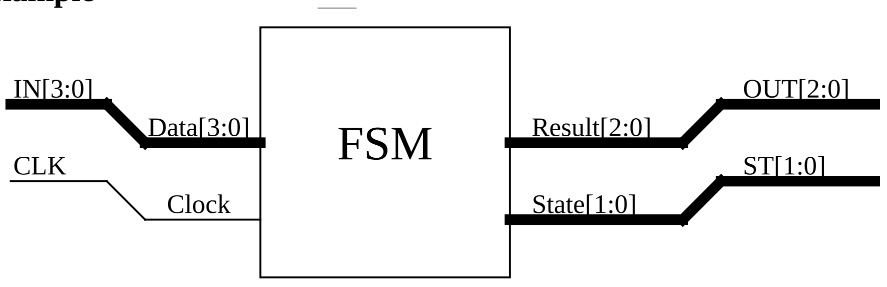
1.5.3.3 Functions

$$\textbf{Result} = f(\textbf{State})$$ Moore machine

$$\textbf{Result} = f(\textbf{State, Data})$$ Mealy machine

$$\textbf{State}_{T+1} = f(\textbf{State}_T\textbf{, Data})$$

1.5.3.4 Example

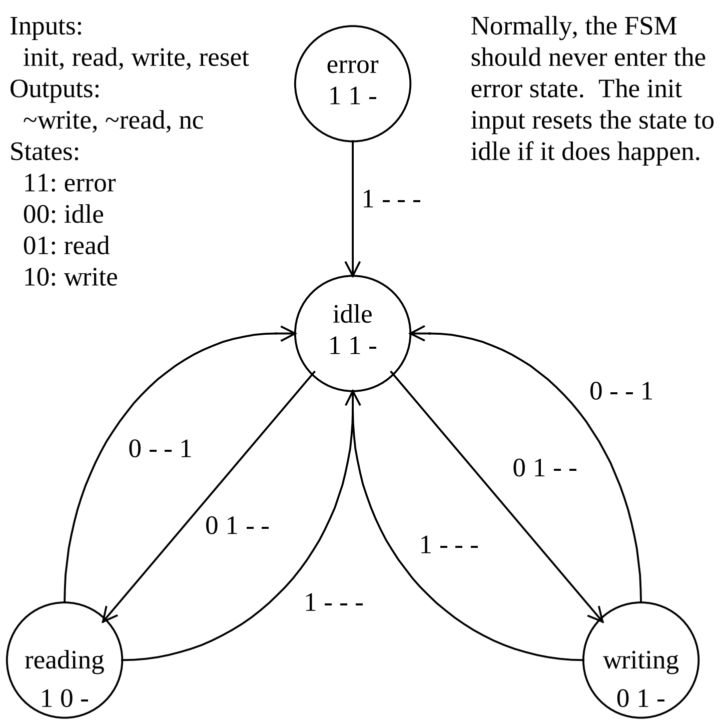


LPM_WidthIn = 4
LPM_WidthOut = 3
LPM_WidthS = 2
LPM_File = FSM1.txt
LPM_TruthType = FD
LPM_Pvalue = 0
LPM_Avalue = 0
LPM_TYPE = FSM

This diagram is for illustrative purposes only and is not intended to specify any implementation details.

The file FSM1.txt contains:

.start	Kiss		
.I 4			
.o 3			
.p 5			
1 - - -	dc	idle	1 1 -
0 1 - -	idle	reading	1 0 -
0 0 1 -	idle	writing	0 1 -
0 - - 1	reading	idle	1 1 -
0 - - 1	writing	idle	1 1 -
.code	dc	- -	
.code	idle	0 0	
.code	reading	0 1	
.code	writing	1 0	



This can be better understood by considering the ninth line form the FSM1.txt file:

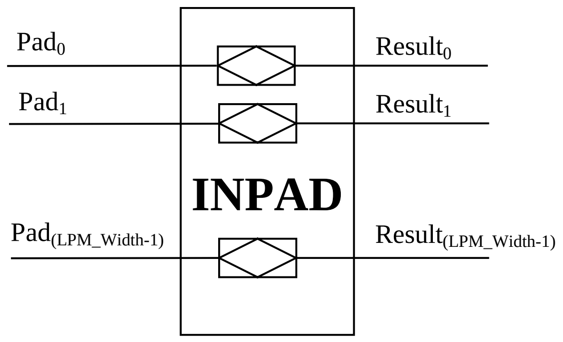
0 - - 1	reading	idle	1 1 -
Data (input)	From state	To state	**Result** (Output)

Although all LPM_FSMs have a valid state encoding, the fitter is free to re-encode the states. Care must be taken in re-encoding if the state is brought outside the LPM_FSM.

1.6 PAD PRIMITIVES

1.6.1 LPM_INPAD

Input Pad



1.6.1.1 Ports

Port Name	Type	Usage	Description	Comments
Pad	I	Optional	External Data input	Vector, LPM_Width wide
Result	O	Required	Data from Pads	Vector, LPM_Width wide

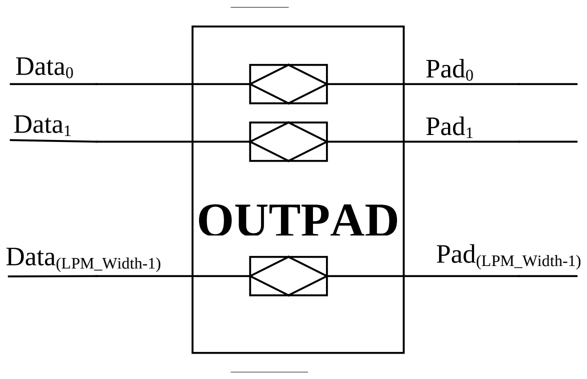
1.6.1.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of input vector

1.6.1.3 Function

Result = Pad

1.6.2 LPM_OUTPAD
Output Pad



1.6.2.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data for output from pads	Vector, LPM_Width wide
Pad	O	Optional	Pads to output data	Vector, LPM_Width wide

1.6.2.2 Properties

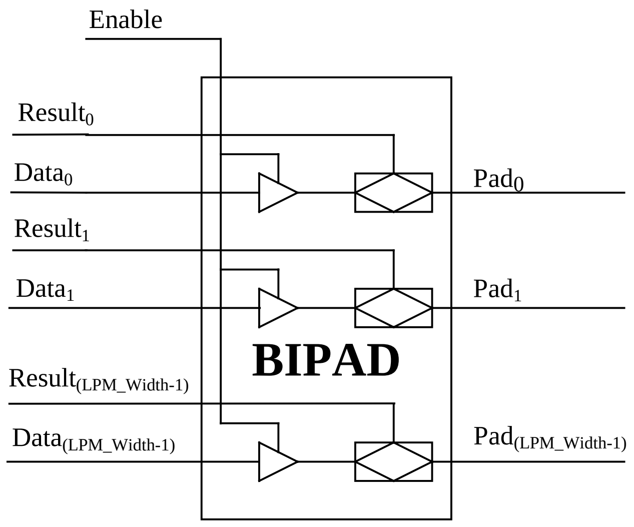
Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of output vector

1.6.2.3 Function

Pad = Data

1.6.3 LPM_BIPAD

Bi-directional input/output Pad



1.6.3.1 Ports

Port Name	Type	Usage	Description	Comments
Data	I	Required	Data for output from pads	Vector, LPM_Width wide
Enable	I	Required	Tristate Enable to the pad	
Result	O	Optional	Data input from the pad	Vector, LPM_Width wide
Pad	I/O	Optional	Pad for input/output	Vector, LPM_Width wide

1.6.3.2 Properties

Property	Usage	Value	Comments
LPM_Width	Required	LPM Value > 0	Width of output vector

1.6.3.3 Function

If Enable = 1, then Pad = Data

else if Enable = 0, then Result = Pad

If the Result port is not connected, then this module acts as a Tristate output port .