LatticeECP2/M Memory Usage Guide

June 2013 Technical Note TN1104

Introduction

This technical note discusses memory usage for the LatticeECP2™ and LatticeECP2M™ device families. It is intended to be used by design engineers as a guide for integrating the EBR- (Embedded Block RAM) and PFU-based memories in this device family using the ispLEVER® design tool.

The architecture of these devices provides resources for FPGA on-chip memory applications. The sysMEM™ EBR complements the distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM, FIFO and ROM memories can be constructed using the EBR. LUTs and PFU can implement Distributed Single-Port RAM, Dual-Port RAM and ROM.

The capabilities of the EBR RAM and PFU RAM are referred to as primitives and are described later in this document. Designers can utilize the memory primitives in two ways via the IPexpress™ tool in the ispLEVER software. The IPexpress GUI allows users to specify the memory type and size required. IPexpress takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.

The remainder of this document discusses the use of IPexpress, memory modules and memory primitives.

Memories in LatticeECP2/M Devices

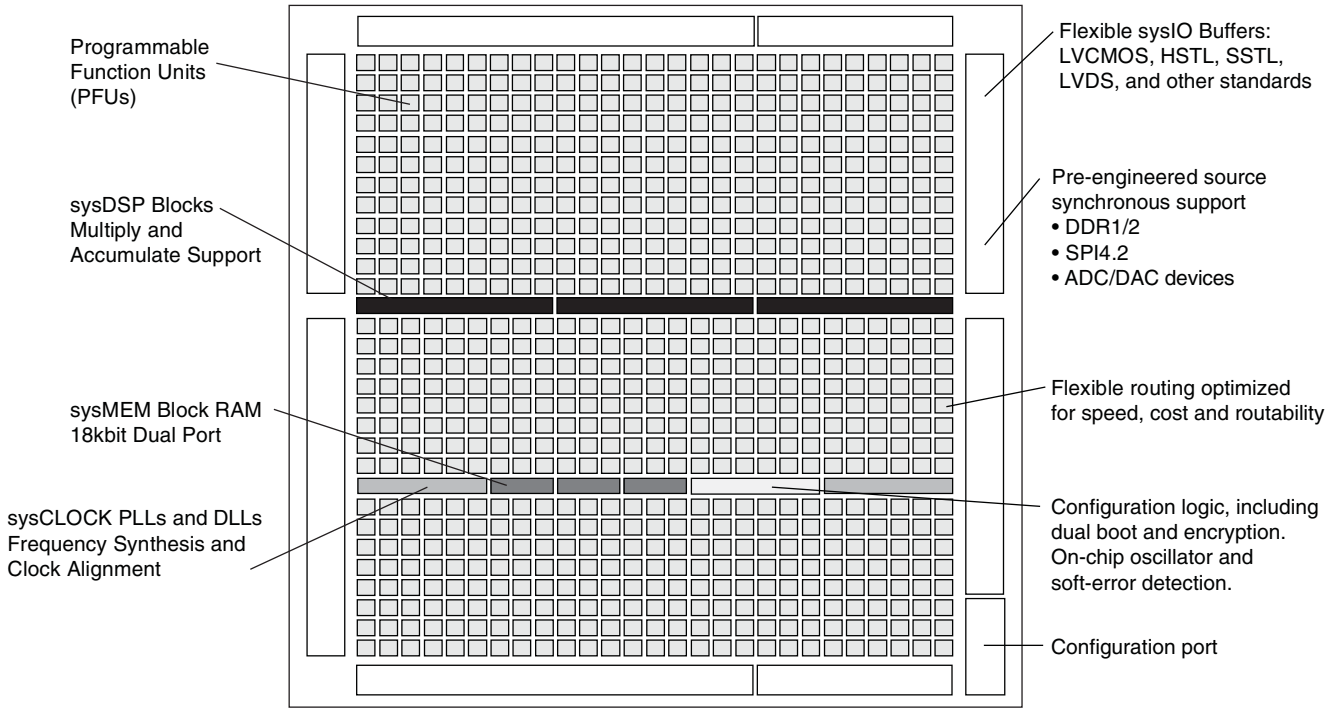
There are two kinds of logic blocks, the Programmable Functional Unit (PFU) and Programmable Functional Unit without RAM (PFF). The PFU contains the building blocks for logic, arithmetic, RAM, ROM and register functions. The PFF block contains building blocks for logic, arithmetic and ROM functions. Both PFU and PFF blocks are optimized for flexibility allowing complex designs to be implemented quickly and efficiently. Logic Blocks are arranged in a two-dimensional array. Only one type of block is used per row.

The LatticeECP2 family of devices contains up to two rows of sysMEM EBR blocks and the LatticeECP2M family of devices contains up to seven rows of sysMEM EBR blocks. sysMEM EBRs are large, dedicated 18K fast memory blocks. Each sysMEM block can be configured in a variety of depths and widths of RAM or ROM. In addition, LatticeECP2/M devices contain up to two rows of sysDSP™ blocks. Each sysDSP block has multipliers and accumulators, which are the building blocks for complex signal processing capabilities

Table 11-1. LatticeECP2/M LUT and Memory Densities

LUTs	6K	12K	20K		35K		50K		70K		100K
Device	**ECP2-6**	**ECP2-12**	**ECP2-20**	**ECP2M-20**	**ECP2-35**	**ECP2M-35**	**ECP2-50**	**ECP2M-50**	**ECP2-70**	**ECP2M-70**	**ECP2M-100**
LUTs (K)	6	12	21	19	32	34	48	48	68	67	95
Distributed RAM (Kbits)	12	24	42	41	65	71	96	101	136	145	202
EBR SRAM Blocks	3	12	15	66	18	114	21	225	60	246	288
EBR SRAM (Kbits)	55	221	276	1217	332	2101	387	4147	1106	4534	5308

Figure 11-1. Simplified Block Diagram, LatticeECP2-6 Device (Top Level)



Utilizing IPexpress

Designers can utilize IPexpress to easily specify a variety of memories in their designs. These modules are constructed using one or more memory primitives along with general purpose routing and LUTs, as required. The available primitives are:

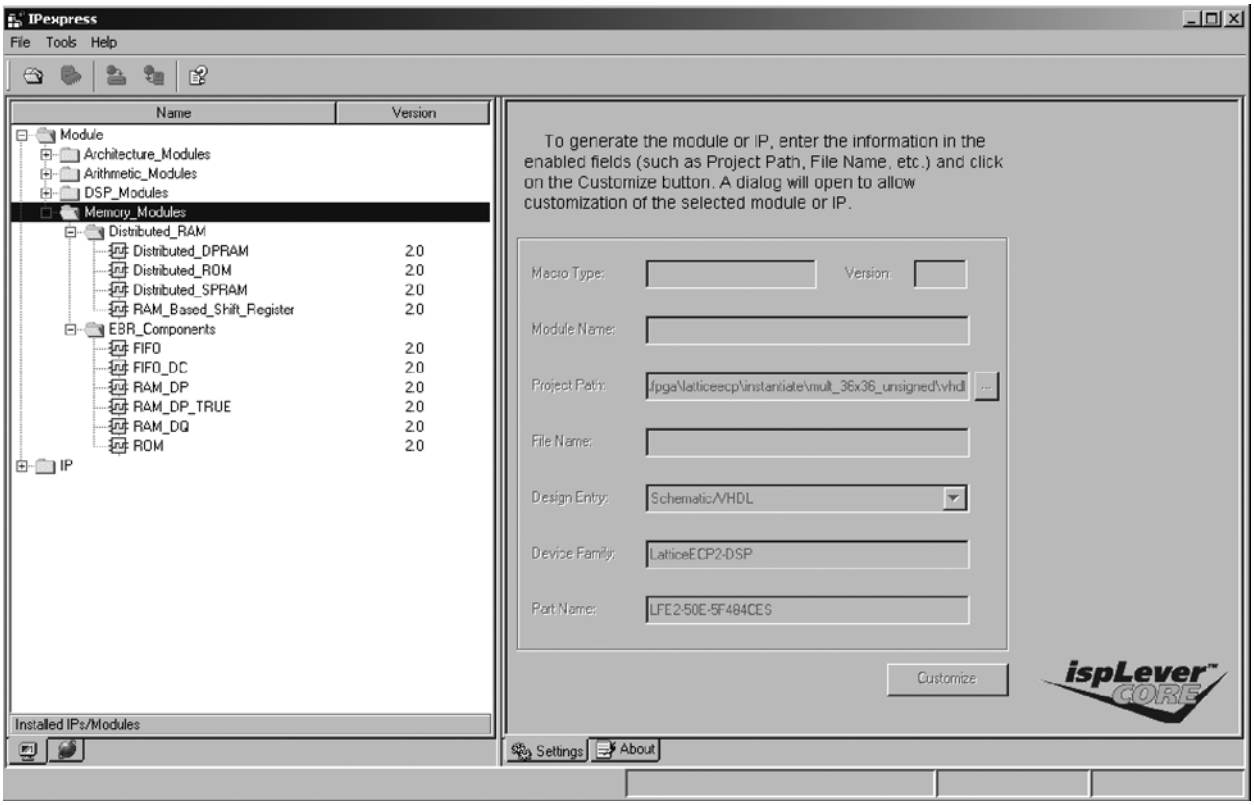
- Single Port RAM (RAM_DQ) – EBR-based
- Dual PORT RAM (RAM_DP_TRUE) – EBR-based
- Pseudo Dual Port RAM (RAM_DP) – EBR-based
- Read Only Memory (ROM) – EBR-Based
- First In First Out Memory (Dual Clock) (FIFO_DC) – EBR-based
- Distributed Single Port RAM (Distributed_SPRAM) – PFU-based
- Distributed Dual Port RAM (Distributed_DPRAM) – PFU-based
- Distributed ROM (Distributed_ROM) – PFU/PFF-based
- Distributed Shift Register (RAM_Based_Shift_Register) – PFU-based (see IPexpress Help for details)

IPexpress Flow

For generating any of these memories, create (or open) a project for the LatticeECP2/M devices.

From the Project Navigator, select **Tools > IPexpress** or click on the button in the toolbar when LatticeECP2/M devices are targeted in the project. This opens the IPexpress main window as shown in Figure 11-2.

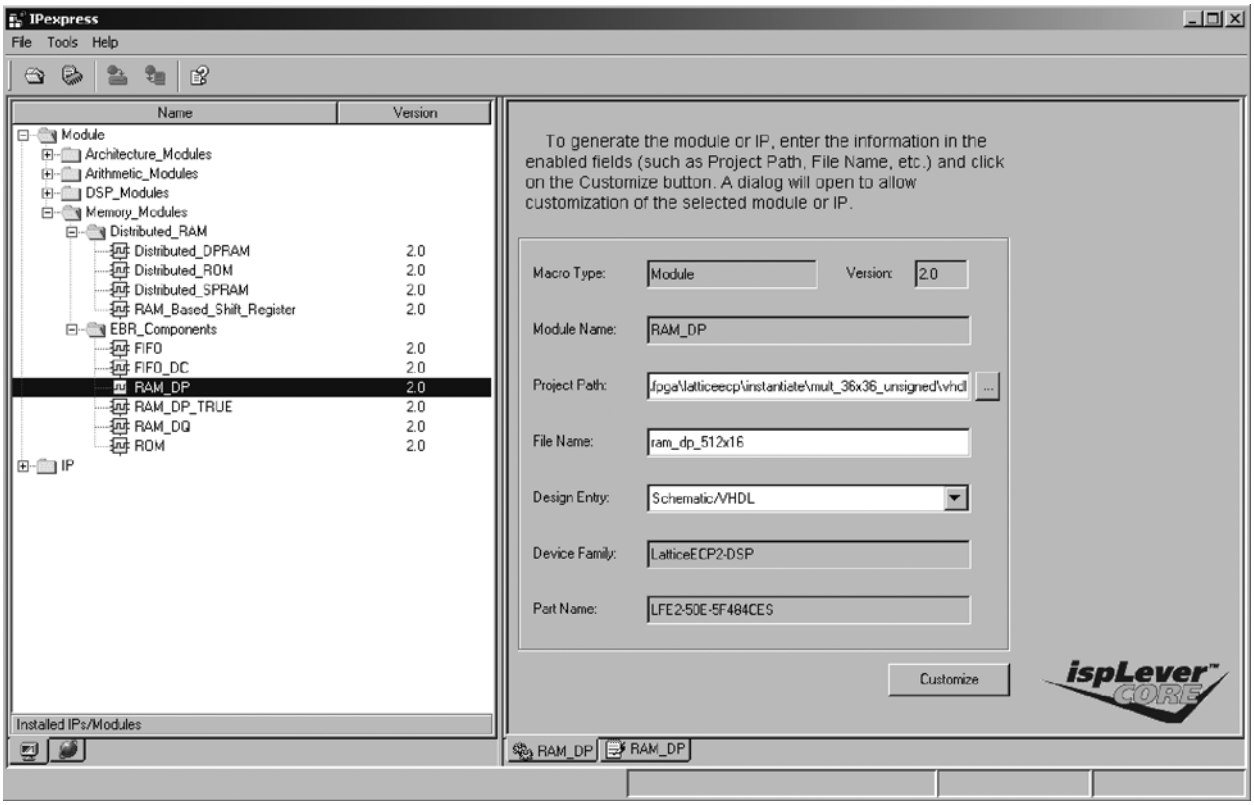
Figure 11-2. IPexpress – Main Window



The left pane of this window includes the Module Tree. The EBR-based Memory Modules are under the **EBR_Components** and the PFU-based Distributed Memory Modules are under **Storage_Components**, as shown in Figure 11-2.

As an example, let us consider generating an EBR-based Pseudo Dual Port RAM of size 512x16. Select **RAM_DP** under **EBR_Components**. The right pane changes as shown in Figure 11-3.

Figure 11-3. Example Generating Pseudo Dual Port RAM (RAM_DP) Using IPexpress



In the right pane, options like the **Device Family**, **Macro Type**, **Category**, and **Module_Name** are device and selected module dependent. These cannot be changed in IPexpress.

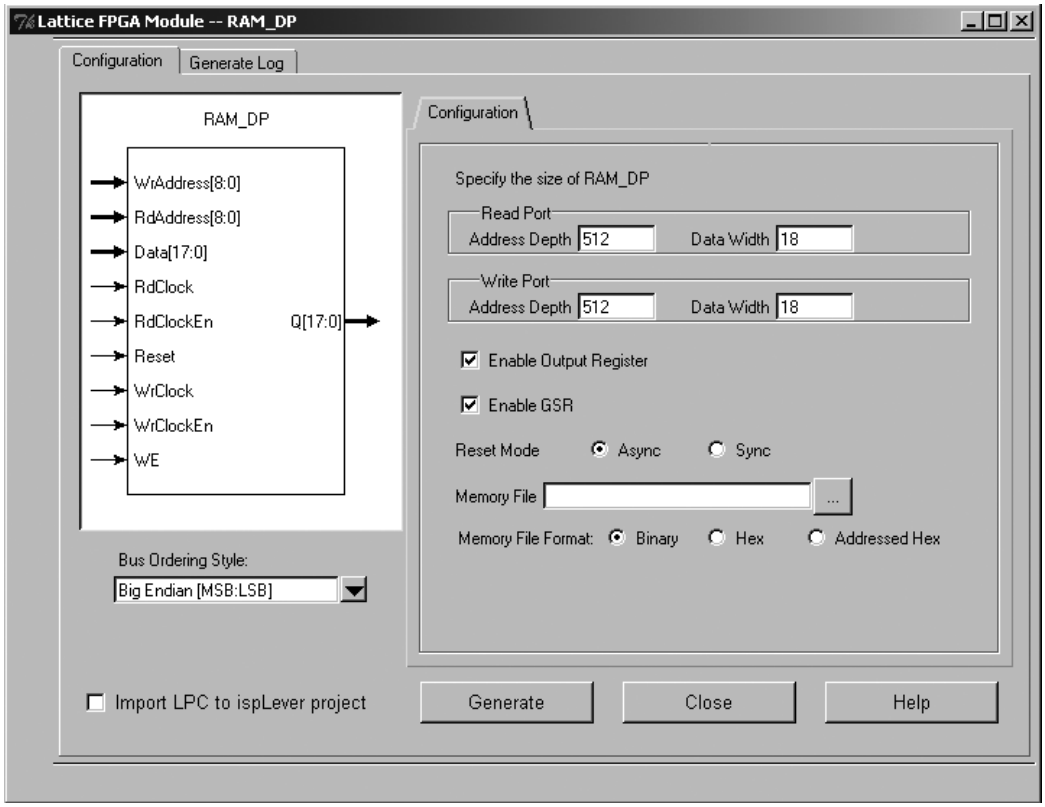
Users can change the directory where the generated module files will be placed by clicking the **Browse** button in the **Project Path**.

The **Module Name** text box allows users to specify an entity name for the module they are about to generate. Users must provide this entity name.

Design entry, Verilog or VHDL, by default, is the same as the project type. If the project is a VHDL project, the selected design entry option will be “Schematic/ VHDL”, and “Schematic/ Verilog-HDL” if the project type is Verilog-HDL.

The **Device** pull-down menu allows users to select different devices within the same family, LatticeECP2/M in this example. By clicking the **Customize** button, another window opens where users can customize the RAM (Figure 11-4).

Figure 11-4. Example Generating Pseudo Dual Port RAM (RAM_DP) Module Customization



The left side of this window shows the block diagram of the module. The right side includes the **Configuration** tab where users can choose options to customize the RAM_DP (e.g. specify the address port sizes and data widths).

Users can specify the address depth and data width for the **Read Port** and the **Write Port** in the text boxes provided. In this example, we are generating a Pseudo Dual Port RAM of size 512 x 16. Users can also create RAMs of different port widths for Pseudo Dual Port and True Dual Port RAMs.

The Input Data and the Address Control are always registered, as the hardware only supports the clocked write operation for the EBR based RAMs. The check box **Enable Output Registers** inserts the output registers in the Read Data Port. Output registers are optional for EBR-based RAMs.

Users have the option to set the **Reset Mode** as Asynchronous Reset or Synchronous Reset. **Enable GSR** can be checked to enable the Global Set Reset. If an EBR is pre-loaded during configuration, the GSR input must be disabled or the release of the GSR during device Wake Up must occur before the release of the device I/Os becomes active. These instructions apply to all EBR RAM and ROM implementations. Note that there are no reset restrictions if the EBR synchronous reset is used and the EBR GSR input is disabled.

Users can also pre-initialize their memory with the contents specified in the **Memory File**. It is optional to provide this file in the RAM; however for ROM, the Memory File is required. These files can be of Binary, Hex or Addresses Hex format. The details of these formats are discussed in the Initialization File section of this document.

At this point, users can click the **Generate** button to generate the module they have customized. A VHDL or Verilog netlist is then generated and placed in the specified location. Users can incorporate this netlist in their designs.

Another important button is the **Load Parameters** button. IPexpress stores the parameters specified in a <module_name>.lpc file. This file is generated along with the module. Users can click on the Load Parameters button to load the parameters of a previously generated module to re-visit or make changes to them.

Once the module is generated, users can either instantiate the *.lpc or the Verilog-HDL/ VHDL file in top-level module of their design.

The various memory modules, both EBR and distributed, are discussed in detail in this document.

Memory Modules

ECC is supported in most memories. If you choose to use ECC, you will have a 2-bit error signal.

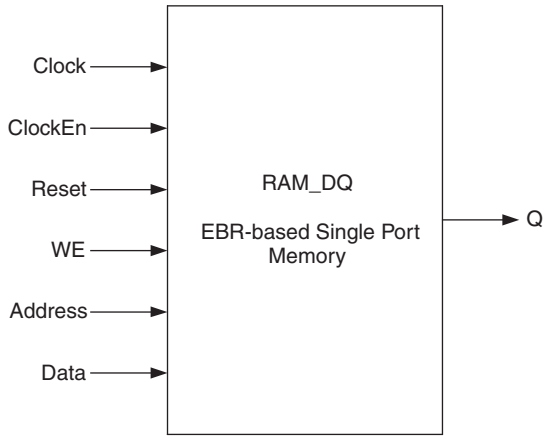
- When Error[1:0]=00, there is no error.
- When Error[0]=1, it indicates that there was a 1 bit error which was fixed.
- When Error[1]=1, it indicates that there was a 2-bit error which cannot be corrected.

Single Port RAM (RAM_DQ) – EBR Based

The EBR blocks in LatticeECP2/M devices can be configured as Single Port RAM or RAM_DQ. IPexpress allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirements.

IPexpress generates the memory module as shown in Figure 11-5.

Figure 11-5. Single Port Memory Module Generated by IPexpress



Since the device has a number of EBR blocks, the generated module makes use of these EBR blocks, or primitives, and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For memory sizes smaller than an EBR block, the module will be created in one EBR block. For memory sizes larger than one EBR block, multiple EBR blocks can be cascaded in depth or width (as required to create these sizes).

In Single Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for the Single Port Memory are listed in Table 11-2. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DQ primitive.

Table 11-2. EBR-based Single Port Memory Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
Clock	CLK	Clock	Rising Clock Edge
ClockEn	CE	Clock Enable	Active High
Address	AD[x:0]	Address Bus	—
Data	DI[y:0]	Data In	—
Q	DO[y:0]	Data Out	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

Reset (or RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

Chip Select (CS) is a useful port in the EBR primitive when multiple cascaded EBR blocks are required by the memory. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily. If the memory size specified by the user requires more than eight EBR blocks, the ispLEVER software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) for each EBR block for the devices are listed in Table 11-3.

Table 11-3. Single Port Memory Sizes for 16K Memories for LatticeECP2/M

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
16K x 1	DI	DO	AD[13:0]
8K x 2	DI[1:0]	DO[1:0]	AD[12:0]
4K x 4	DI[3:0]	DO[3:0]	AD[11:0]
2K x 9	DI[8:0]	DO[8:0]	AD[10:0]
1K x 18	DI[17:0]	DO[17:0]	AD[9:0]
512 x 36	DI[35:0]	DO[35:0]	AD[8:0]

Table 11-4 shows the various attributes available for the Single Port Memory (RAM_DQ). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 11-4. Single Port RAM Attributes for LatticeECP2/M

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Address depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES
Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Write Mode	Read / Write Mode for Write Port	NORMAL, WRITE-THROUGH	NORMAL	YES
Chip Select Decode	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x00000000000000000000 00000000000000000000 00000000000000000000 0000000000000.....0xFFFF FFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFF	0x000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000	NO

The Single Port RAM (RAM_DQ) can be configured as NORMAL or WRITE THROUGH modes. Each of these modes affects the data coming out of port Q of the memory during the write operation followed by the read operation at the same memory location.

Additionally, users can select to enable the output registers for RAM_DQ. Figures 11-6-11-9 show the internal timing waveforms for the Single Port RAM (RAM_DQ) with these options.

It is important that no setup and hold time violations occur on the address registers (Address). Failing to meet these requirements can result in corruption of memory contents. This applies to both read and write operations.

A Post Place and Route timing report in Lattice Diamond® or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

Figure 11-6. Single Port RAM Timing Waveform – NORMAL Mode, without Output Registers

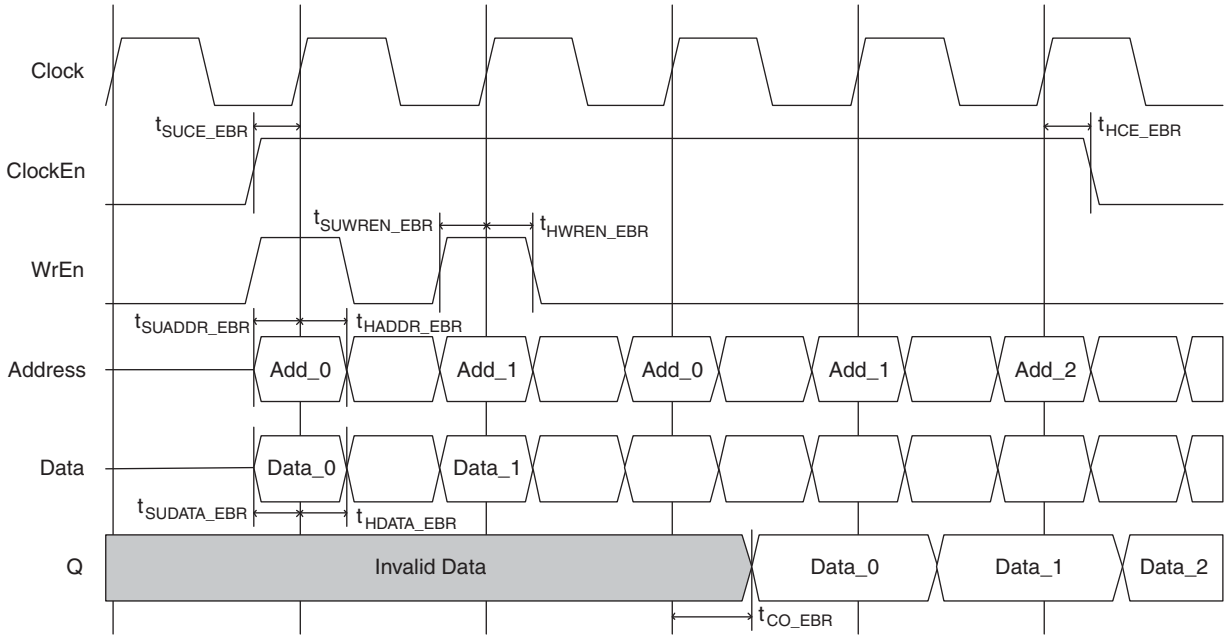


Figure 11-7. Single Port RAM Timing Waveform – NORMAL Mode, with Output Registers

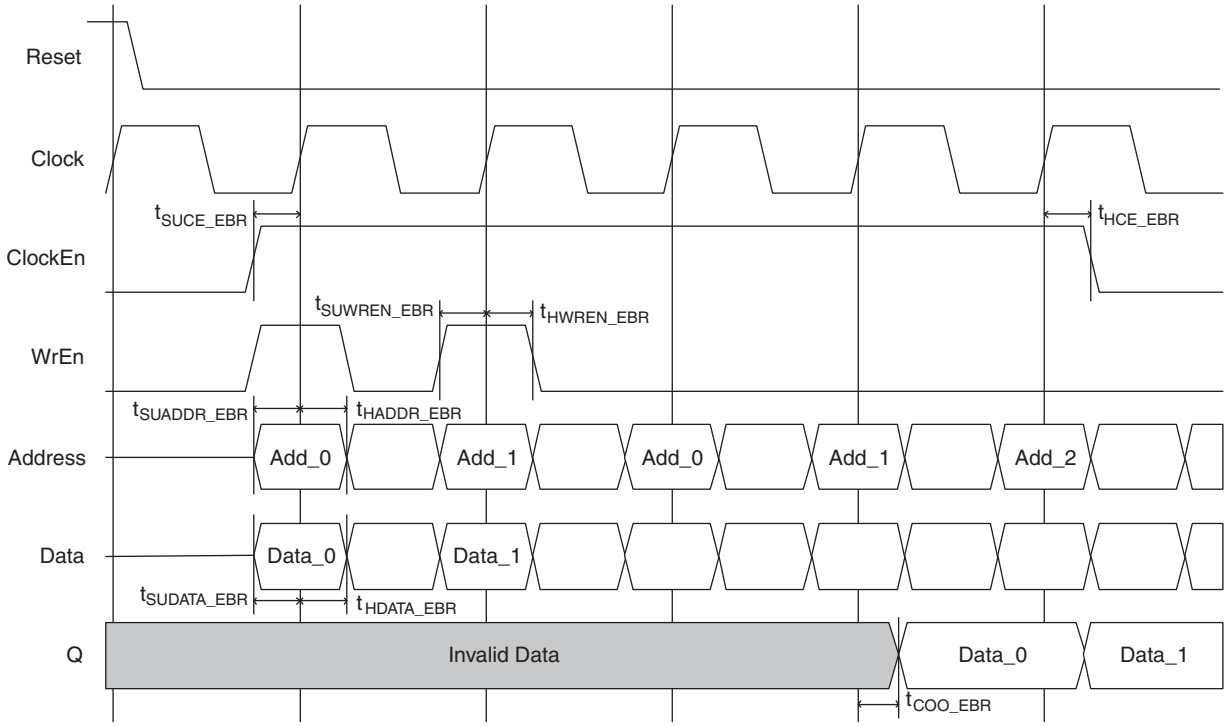


Figure 11-8. Single Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

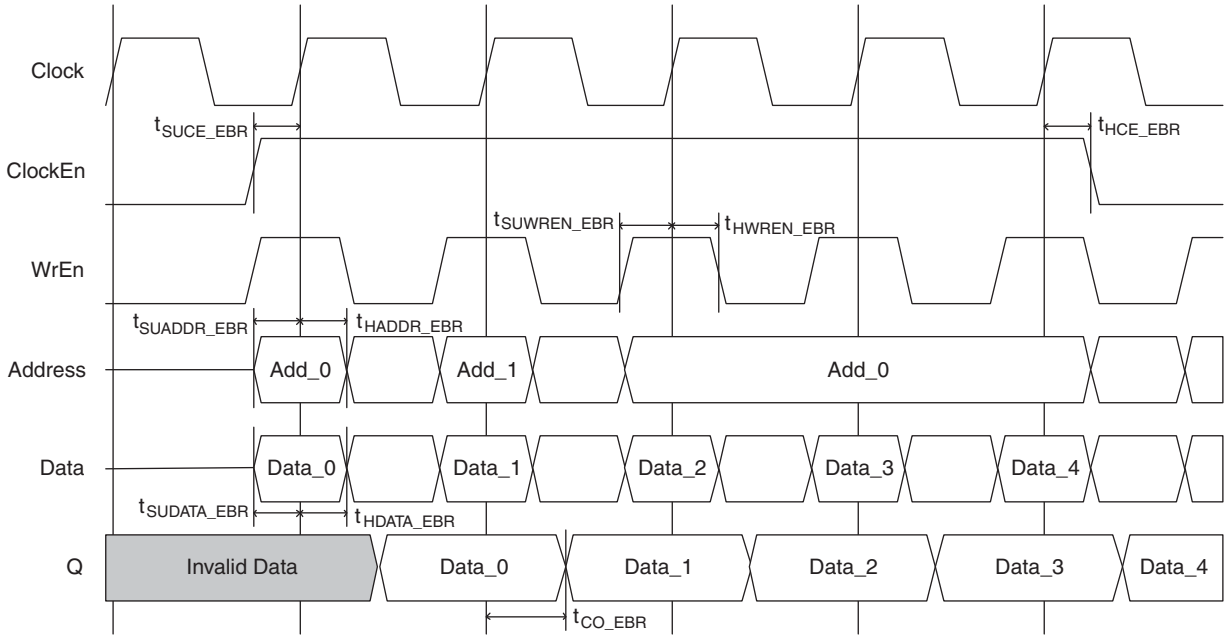
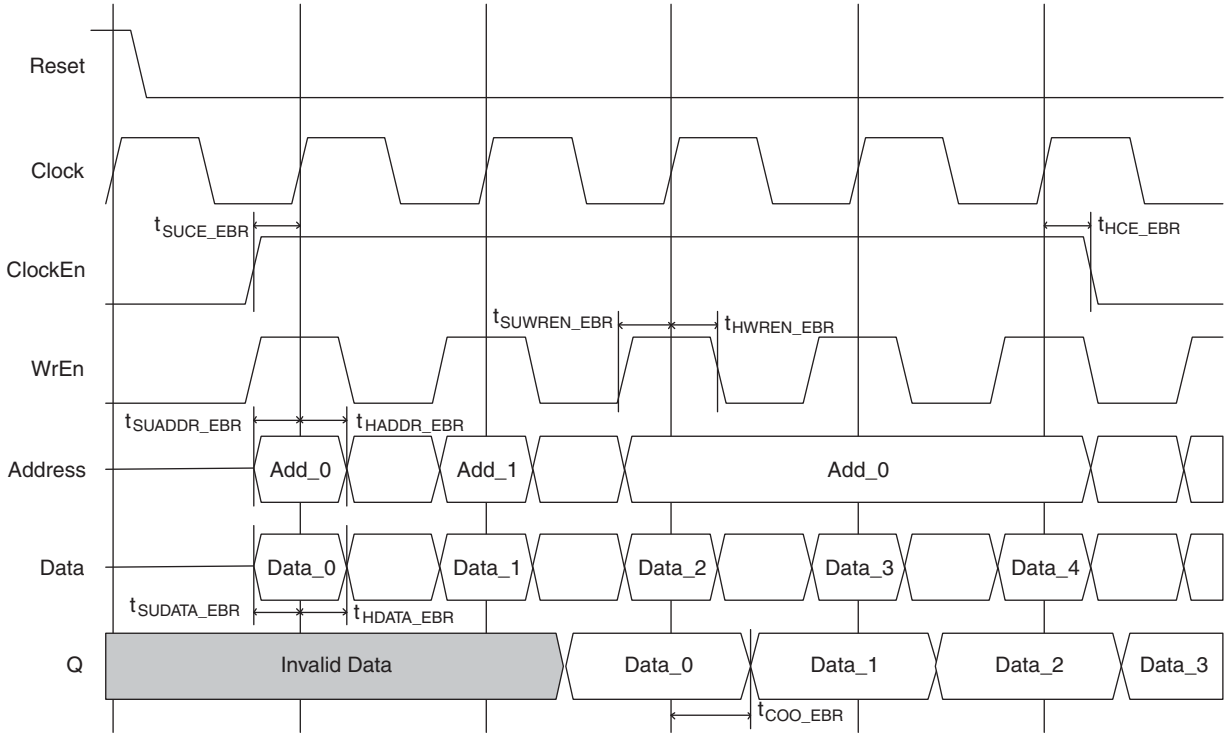


Figure 11-9. Single Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

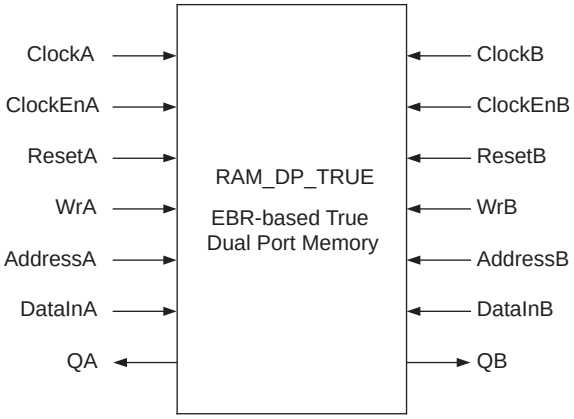


True Dual Port RAM (RAM_DP_TRUE) – EBR Based

The EBR blocks in the LatticeECP2/M devices can be configured as True-Dual Port RAM or RAM_DP_TRUE. IPexpress allows users to generate the Verilog-HDL, VHDL or EDIF netlists for the memory size as per design requirements.

IPexpress generates the memory module as shown in Figure 11-10.

Figure 11-10. True Dual Port Memory Module Generated by IPexpress



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. When the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded in depth or width (as required to create these sizes).

In True Dual Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for Single Port Memory are listed in Table 11-5. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP_TRUE primitive.

Table 11-5. EBR-based True Dual Port Memory Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
ClockA, ClockB	CLKA, CLKB	Clock for PortA and PortB	Rising Clock Edge
ClockEnA, ClockEnB	CEA, CEB	Clock Enables for Port CLKA and CLKB	Active High
AddressA, AddressB	ADA[18:19-x1], ADB[18:19-x2]	Address Bus port A and port B	—
DataA, DataB	DIA[y1:0], DIB[y2:0]	Input Data port A and port B	—
QA, QB	DOA[y1:0], DOB[y2:0]	Output Data port A and port B	—
WrA, WrB	WEA, WEB	Write enable port A and port B	Active High
ResetA, ResetB	RSTA, RSTB	Reset for PortA and PortB	Active High
—	CSA[2:0], CSB[2:0]	Chip Selects for each port	—

Reset (or RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

Chip Select (CS) is a useful port in the EBR primitive when multiple cascaded EBR blocks are required by the memory. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. Since CS is a 3-bit bus, it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the ispLEVER software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices are listed in Table 11-6.

Table 11-6. True Dual Port Memory Sizes for 16K Memory for LatticeECP2/M

Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Address Port A [MSB:LSB]	Address Port B [MSB:LSB]
16K x 1	DIA	DIB	DOA	DOB	ADA[13:0]	ADB[13:0]
8K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[12:0]	ADB[12:0]
4K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[11:0]	ADB[11:0]
2K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[10:0]	ADB[10:0]
1K x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	ADA[9:0]	ADB[9:0]

Table 11-7 shows the various attributes available for the Single Port Memory (RAM_DQ). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to the Appendix A.

Table 11-7. True Dual Port RAM Attributes for LatticeECP2/M

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Port A Address depth	Address Depth Port A	16K, 8K, 4K, 2K, 1K		YES
Port A Data Width	Data Word Width Port A	1, 2, 4, 9, 18	1	YES
Port B Address depth	Address Depth Port B	16K, 8K, 4K, 2K, 1K		YES
Port B Data Width	Data Word Width Port B	1, 2, 4, 9, 18	1	YES
Port A Enable Output Registers	Register Mode (Pipelining) for Port A	NOREG, OUTREG	NOREG	YES
Port B Enable Output Registers	Register Mode (Pipelining) for Port B	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Port A Write Mode	Read / Write Mode for Port A	NORMAL, WRITE-THROUGH	NORMAL	YES
Port B Write Mode	Read / Write Mode for Port B	NORMAL, WRITE-THROUGH	NORMAL	YES
Chip Select Decode for Port A	Chip Select Decode for Port A	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Chip Select Decode for Port B	Chip Select Decode for Port B	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x0000000000000000000000000000000000000000000000000000000000000000000000000000000......0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0x0000000000000000000000000000000000000000000000000000000000000000000000000000000	NO

The True Dual Port RAM (RAM_DP_TRUE) can be configured as NORMAL or WRITE THROUGH modes. Each of these modes affects what data comes out of the port Q of the memory during the write operation followed by the read operation at the same memory location. The detailed discussions of the WRITE modes and the constraints of the True Dual Port can be found in Appendix A.

Additionally, users can select to enable the output registers for RAM_DP_TRUE. Figures 11-11 through 11-14 show the internal timing waveforms for the True Dual Port RAM (RAM_DP_TRUE) with these options.

It is important that no setup and hold time violations occur on the address registers (AddressA and AddressB). Failing to meet these requirements can result in corruption of memory contents. This applies to both read and write operations.

A Post Place and Route timing report in Lattice Diamond or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

Figure 11-11. True Dual Port RAM Timing Waveform – NORMAL Mode, without Output Registers

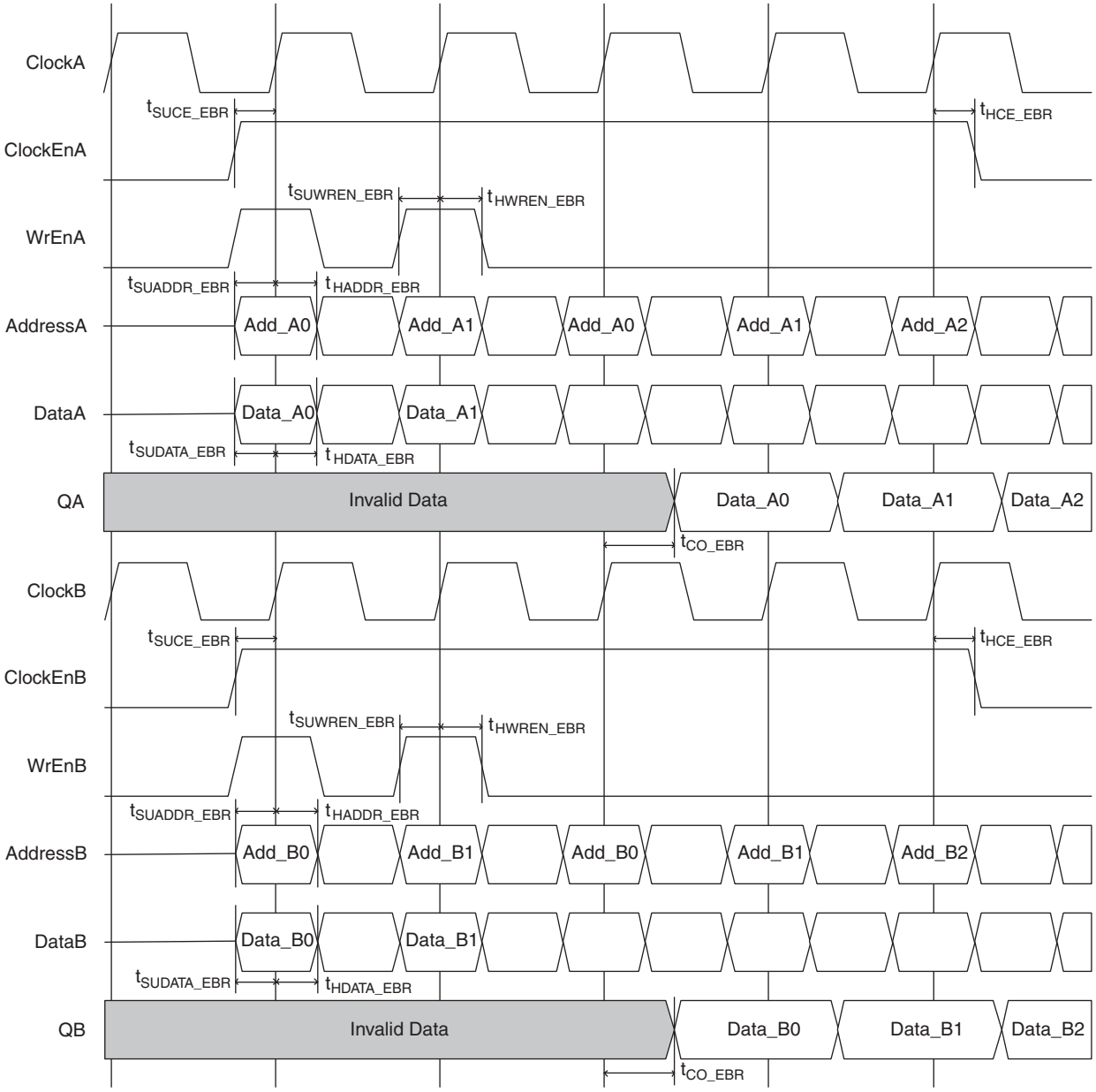


Figure 11-12. True Dual Port RAM Timing Waveform – NORMAL Mode with Output Registers

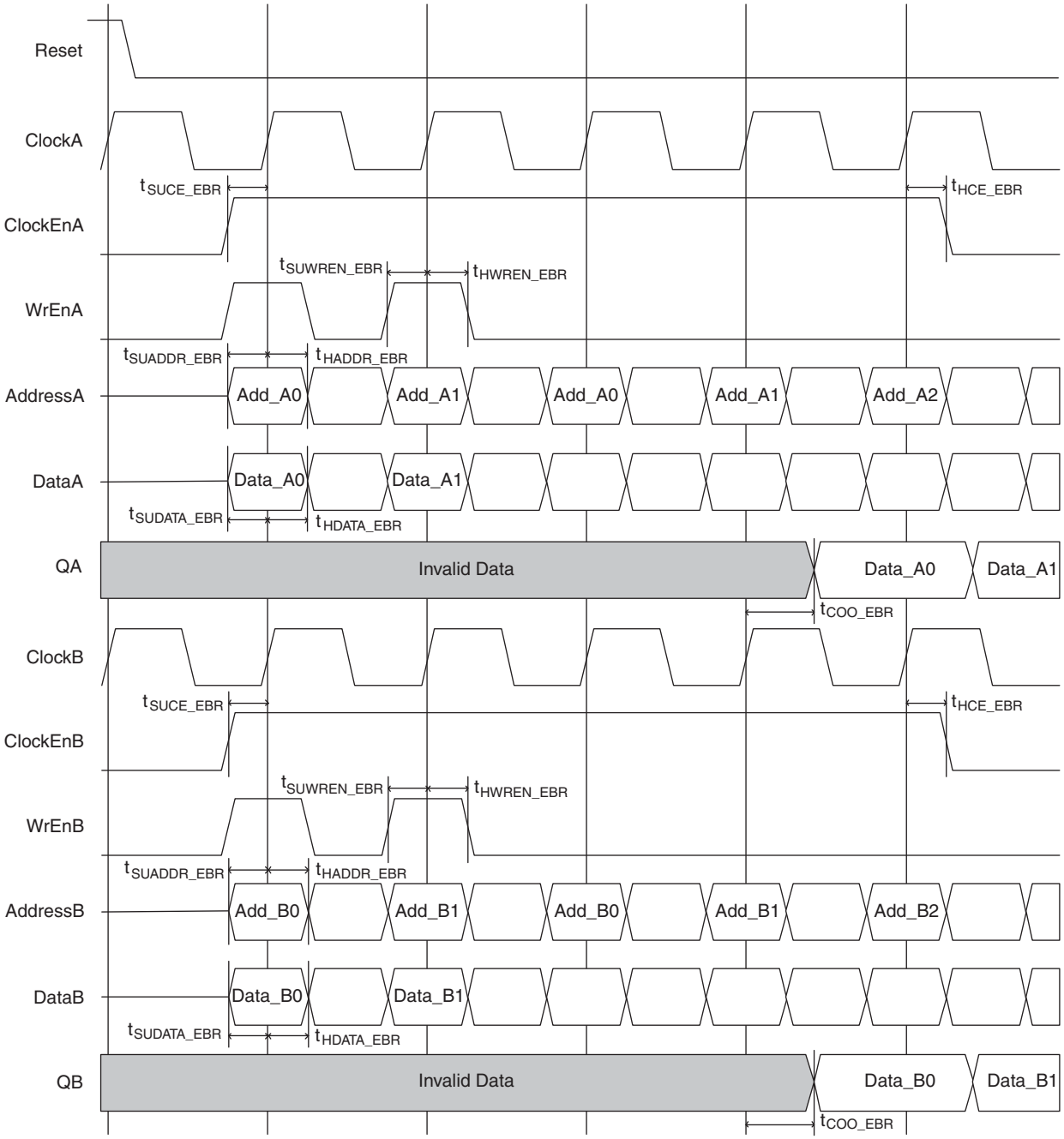


Figure 11-13. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

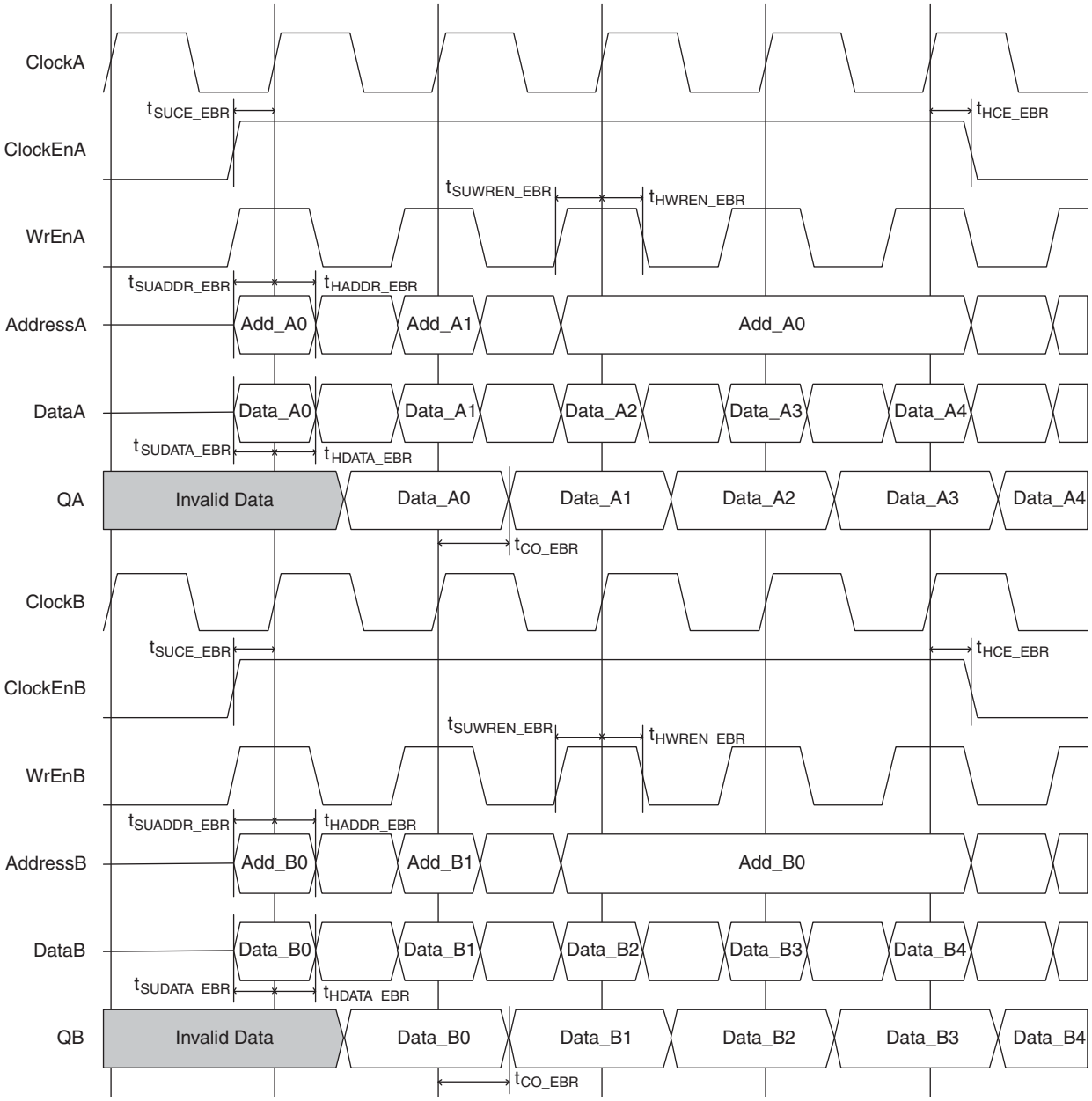
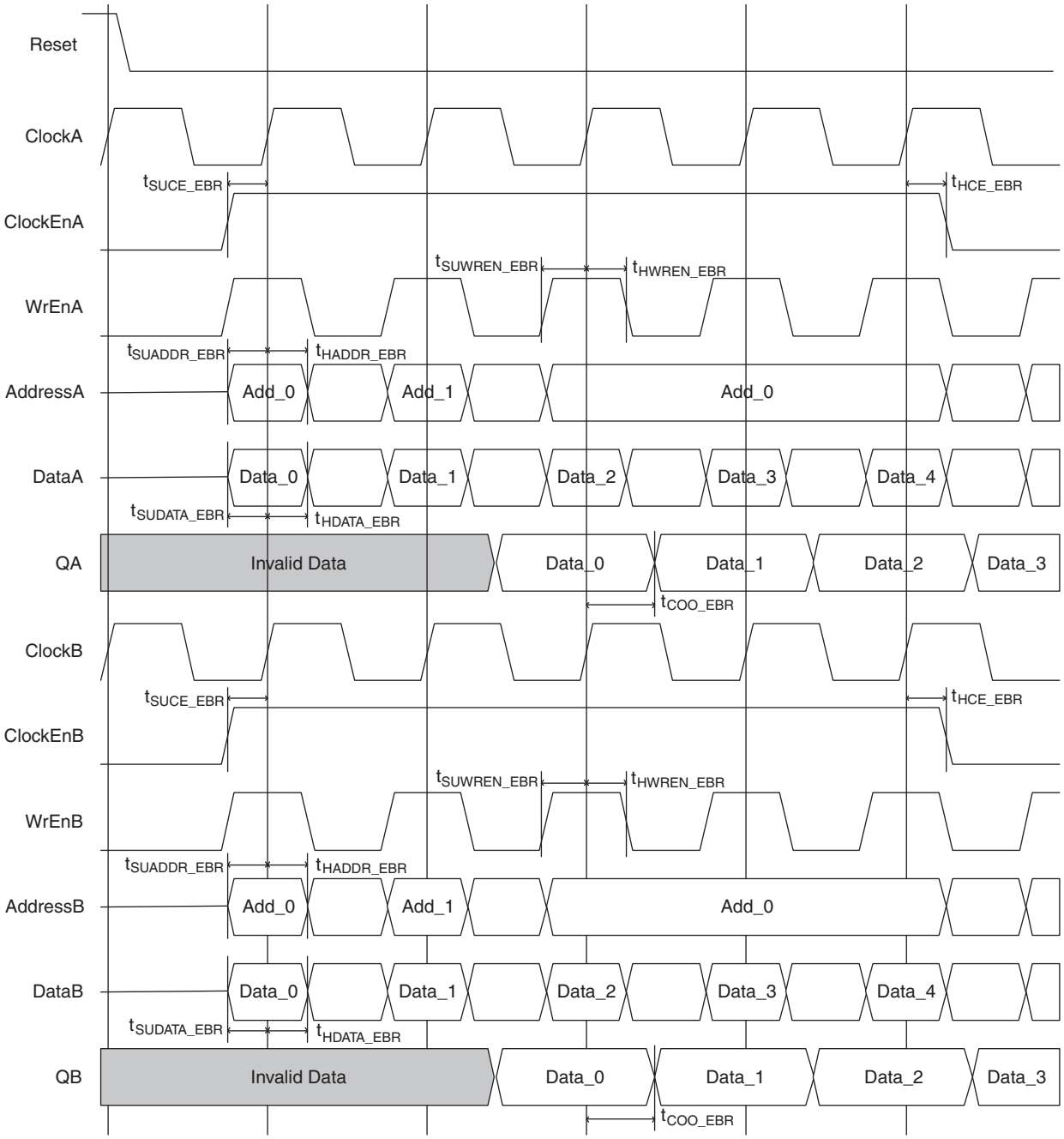


Figure 11-14. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

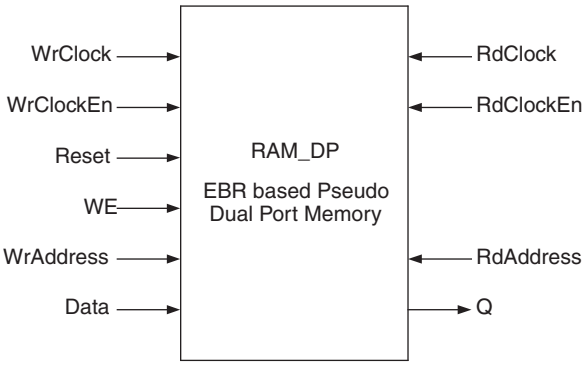


Pseudo Dual Port RAM (RAM_DP) – EBR Based

The EBR blocks in LatticeECP2/M devices can be configured as Pseudo-Dual Port RAM or RAM_DP. IPexpress allows users to generate the Verilog-HDL or VHDL along with EDIF netlists for the memory size as per design requirements.

IPexpress generates the memory module as shown in Figure 11-15.

Figure 11-15. Pseudo Dual Port Memory Module Generated by IPexpress



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded in depth or width (as required to create these sizes).

In Pseudo Dual Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for the Single Port Memory are listed in Table 11-8. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP primitive.

Table 11-8. EBR-based Pseudo-Dual Port Memory Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
RdAddress	ADR[x1:0]	Read Address	—
WrAddress	ADW[x2:0]	Write Address	—
RdClock	CLKR	Read Clock	Rising Clock Edge
WrClock	CLKW	Write Clock	Rising Clock Edge
RdClockEn	CER	Read Clock Enable	Active High
WrClockEn	CEW	Write Clock Enable	Active High
Q	DO[y1:0]	Read Data	—
Data	DI[y2:0]	Write Data	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

Chip Select (CS) is a useful port when multiple cascaded EBR blocks are required by the memory. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. Since CS is a 3-bit bus, it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the ispLEVER software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices are as in Table 11-9.

Table 11-9. Pseudo-Dual Port Memory Sizes for 16K Memory for LatticeECP2/M

Pseudo-Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Read Address Port A [MSB:LSB]	Write Address Port B [MSB:LSB]
16K x 1	DIA	DIB	DOA	DOB	RAD[13:0]	WAD[13:0]
8K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	RAD[12:0]	WAD[12:0]
4K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	RAD[11:0]	WAD[11:0]
2K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	RAD[10:0]	WAD[10:0]
1K x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	RAD[9:0]	WAD[9:0]
512 x 36	DIA[35:0]	DIB[35:0]	DOA[35:0]	DOB[35:0]	RAD[8:0]	WAD[8:0]

Table 11-10 shows the various attributes available for the Pseudo-Dual Port Memory (RAM_DP). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 11-10. Pseudo-Dual Port RAM Attributes for LatticeECP2/M

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Read Port Address Depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES
Read Port Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Write Port Address Depth	Address Depth Write Port	16K, 8K, 4K, 2K, 1K		YES
Write Port Data Width	Data Word Width Write Port	1, 2, 4, 9, 18, 36	1	YES
Write Port Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Read Port Write Mode	Read / Write Mode for Read Port	NORMAL	NORMAL	YES
Write Port Write Mode	Read / Write Mode for Write Port	NORMAL	NORMAL	YES
Chip Select Decode for Read Port	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Chip Select Decode for Write Port	Chip Select Decode for Write Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x0000000000000000000000000 000000000000000000000000000 000000000000000000000000000 0......0xFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFF	0x000000000000 00000000000000 00000000000000 00000000000000 00000000000000 000000000000	NO

Users have the option to enable the output registers for Pseudo-Dual Port RAM (RAM_DP). Figures 11-16 and 11-17 show the internal timing waveforms for Pseudo-Dual Port RAM (RAM_DP) with these options. It is important that no setup and hold time violations occur on the address registers (RdAddress and WrAddress). Failing to meet these requirements can result in corruption of memory contents. This applies to both read and write operations.

A Post Place and Route timing report in Lattice Diamond or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

Figure 11-16. PSEUDO DUAL PORT RAM Timing Diagram – without Output Registers

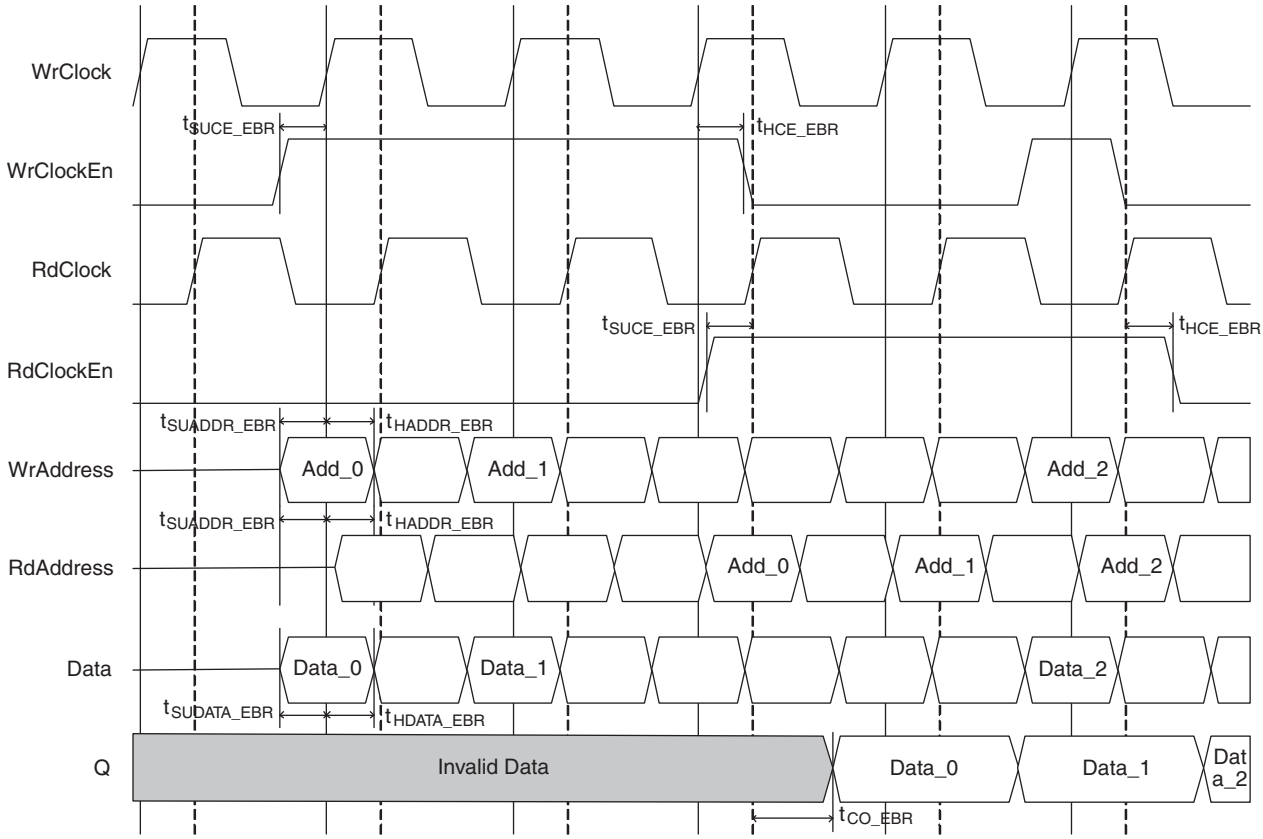
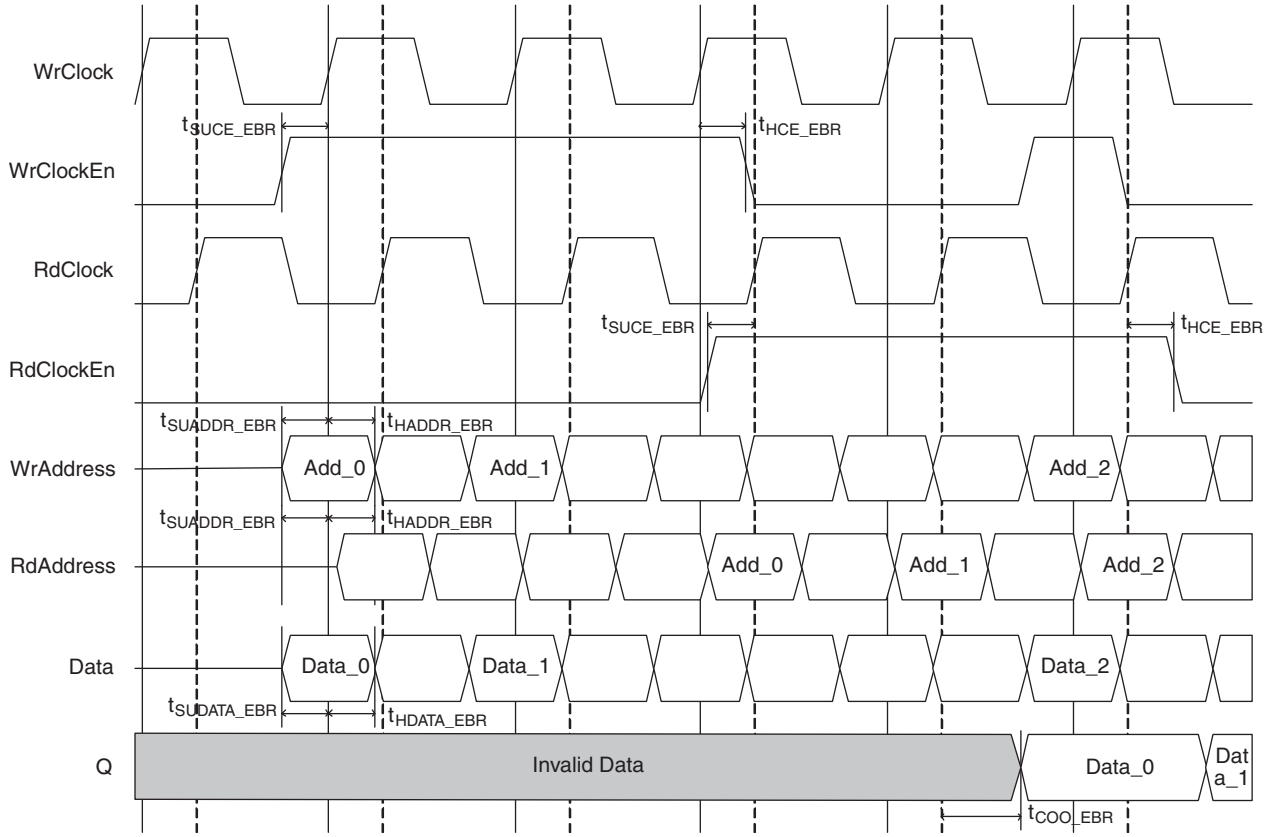


Figure 11-17. PSEUDO DUAL PORT RAM Timing Diagram – with Output Registers

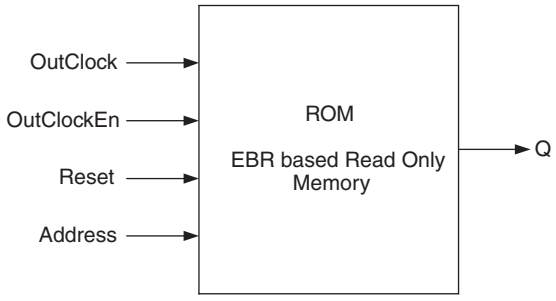


Read Only Memory (ROM) – EBR Based

The EBR blocks in the LatticeECP2/M devices can be configured as Read Only Memory or ROM. IPexpress allows users to generate the Verilog-HDL or VHDL and the EDIF netlist for the memory size, as per design requirements. Users are required to provide the ROM memory content in the form of an initialization file.

IPexpress generates the memory module as shown in Figure 11-18.

Figure 11-18. Read-Only Memory Module Generated by IPexpress



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

In ROM mode, the address for the port is registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for the ROM are listed in Table 11-11. The table lists the corresponding ports for the module generated by IPexpress and for the ROM primitive.

Table 11-11. EBR-based ROM Port Definitions

Port Name in Generated Module	Port Name in the EBR block Primitive	Description	Active State
Address	AD[x:0]	Read Address	—
OutClock	CLK	Clock	Rising Clock Edge
OutClockEn	CE	Clock Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

Chip Select (CS) is a useful port when multiple cascaded EBR blocks are required by the memory. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. Since CS is a 3-bit bus, it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the ispLEVER software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

While generating the ROM using IPexpress, the user must provide the initialization file to pre-initialize the contents of the ROM. These files are the *.mem files and they can be of Binary, Hex or the Addressed Hex formats. The initialization files are discussed in detail in the Initializing Memory section of this document.

Users have the option of enabling the output registers for Read Only Memory (ROM). Figures 11-19 and 11-20 show the internal timing waveforms for the Read Only Memory (ROM) with these options.

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices are as per Table 11-12.

Table 11-12. ROM Memory Sizes for 16K Memory for LatticeECP2/M

ROM	Output Data	Address Port [MSB:LSB]
16K x 1	DOA	WAD[13:0]
8K x 2	DOA[1:0]	WAD[12:0]
4K x 4	DOA[3:0]	WAD[11:0]
2K x 9	DOA[8:0]	WAD[10:0]
1K x 18	DOA[17:0]	WAD[9:0]
512 x 36	DOA[35:0]	WAD[8:0]

Table 11-13 shows the various attributes available for the Read Only Memory (ROM). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Users have the option to enable the output registers for Read Only Memory (ROM). Figures 11-19 and 11-20 show the internal timing waveforms for ROM with these options.

It is important that no setup and hold time violations occur on the address registers (Address). Failing to meet these requirements can result in corruption of memory contents. This applies to both read operations in this case.

A Post Place and Route timing report in Lattice Diamond or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

Table 11-13. ROM Attributes for LatticeECP2/M

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Address depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES
Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Chip Select Decode	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO

Figure 11-19. ROM Timing Waveform – without Output Registers

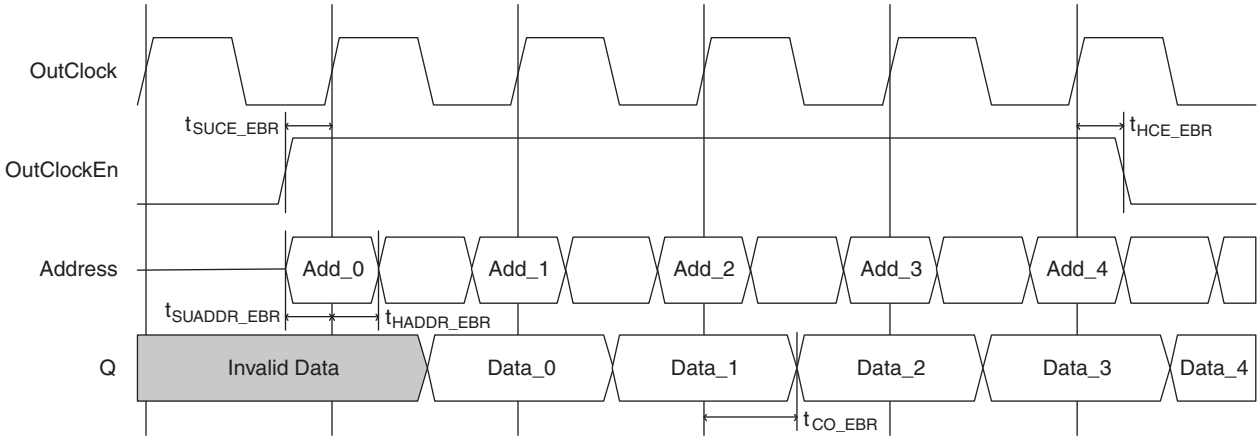
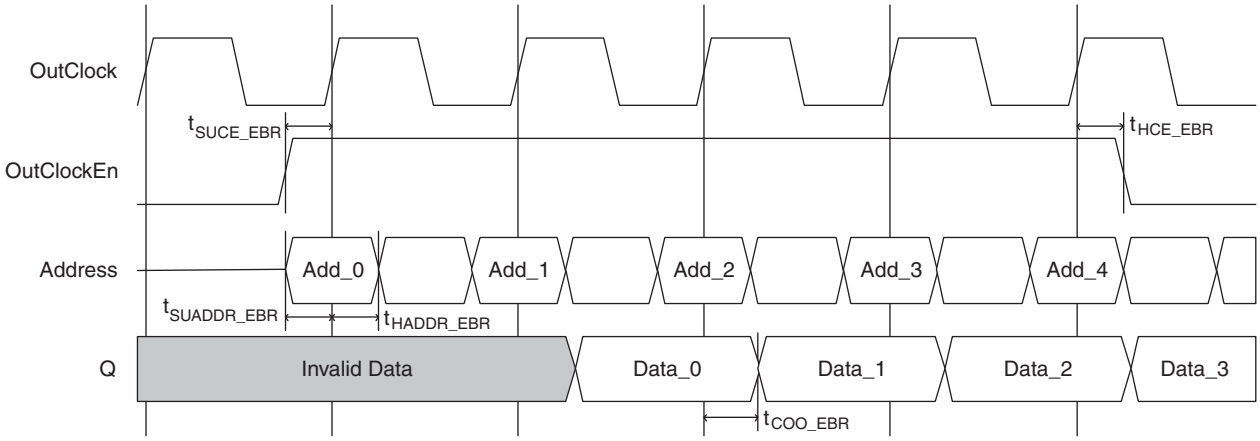


Figure 11-20. ROM Timing Waveform – with Output Registers



First In First Out (FIFO, FIFO_DC) – EBR Based

The hardware has Embedded Block RAM (EBR) which can be configured in Single Port (RAM_DQ), Pseudo-Dual Port (RAM_DP) and True Dual Port (RAM_DP_TRUE) RAMs. The FIFOs in these devices can be built using these RAMs. The IPexpress point tool in the ispLEVER design software allows users to build a FIFO and FIFO_DC around Pseudo Dual Port RAM (or DP_RAM).

Each of these FIFOs can be configured with (pipelined) and without (non-pipelined) output registers. In the pipelined mode users have an extra option to enable the output registers by the RdEn signal. We will discuss the operation in the following sections.

Let us take a look at the operation of these FIFOs.

First In First Out (FIFO) Memory

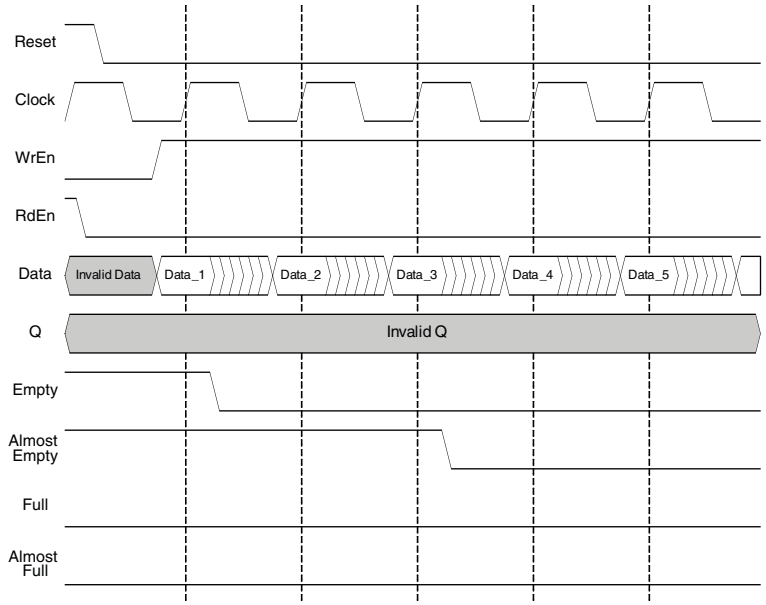
The FIFO, or the single clock FIFO, is an emulated FIFO. The address logic and the flag logic is implemented in the FPGA fabric around the RAM.

The ports available on the FIFO are:

- Reset
- Clock
- WrEn
- RdEn
- Data
- Q
- Full Flag
- Almost Full Flag
- Empty Flag
- Almost Empty Flag

Let us first discuss the non-pipelined or the FIFO without output registers. Figure 11-21 shows the operation of the FIFO when it is empty and the data starts to get written into it.

Figure 11-21. FIFO without Output Registers, Start of Data Write Cycle

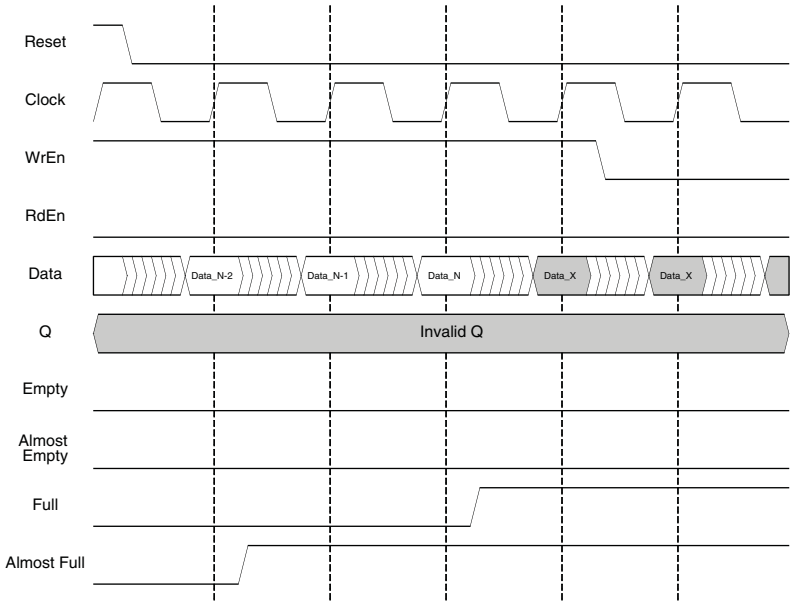


The WrEn signal must be high to start writing into the FIFO. The Empty and Almost Empty flags are high to begin and Full and Almost Full are low.

When the first data is written into the FIFO, the Empty flag de-asserts (or goes low), as the FIFO is no longer empty. In this figure we assume that the Almost Empty setting flag setting is 3 (address location 3). So the Almost Empty flag gets de-asserted when the third address location is filled.

Now let us assume that we continue to write into the FIFO to fill it. When the FIFO is filled, the Almost Full and Full Flags are asserted. Figure 11-22 shows the behavior of these flags. In this figure we assume that FIFO depth is 'N'.

Figure 11-22. FIFO without Output Registers, End of Data Write Cycle

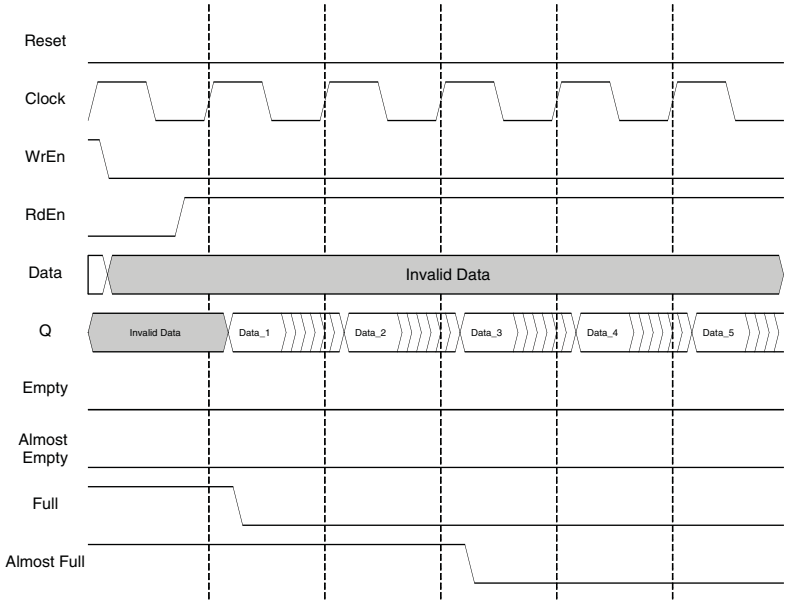


In this case, the Almost Full flag is in the 2 location before the FIFO is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO.

Data_X data inputs do not get written as the FIFO is full (the Full flag is high).

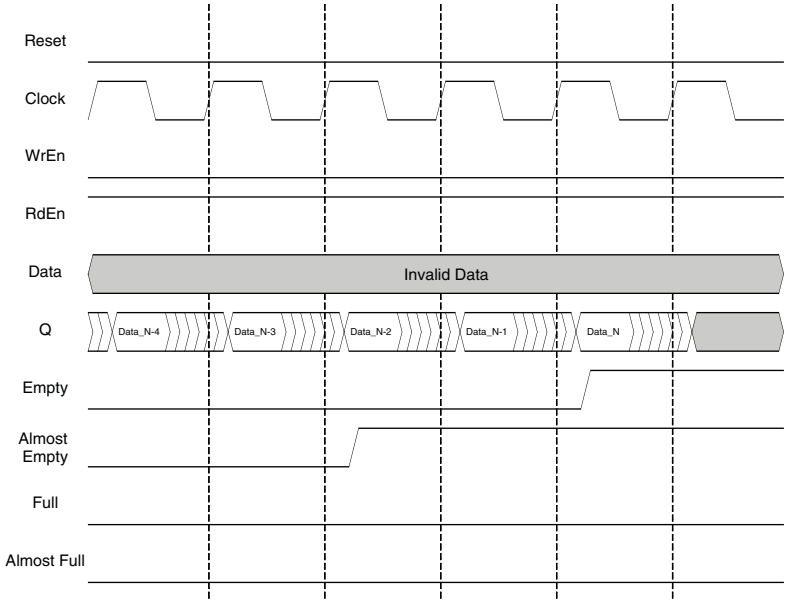
Now let us look at the waveforms when the contents of the FIFO are read out. Figure 11-23 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted, as shown.

Figure 11-23. FIFO without Output Registers, Start of Data Read Cycle



Similarly, as the data is read out and FIFO is emptied, the Almost Empty and Empty flags are asserted.

Figure 11-24. FIFO without Output Registers, End of Data Read Cycle



Figures 11-21 to 11-24 show the behavior of non-pipelined FIFO or FIFO without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is also the extra option for Output registers to be enabled by the RdEn signal.

Figures 11-25 to 11-28 show the similar waveforms for the FIFO with output register and with output register enable with RdEn. It should be noted that flags are asserted and de-asserted with similar timing to the FIFO without output registers. However, it is only the data out 'Q' that is delayed by one clock cycle.

Figure 11-25. FIFO with Output Registers, Start of Data Write Cycle

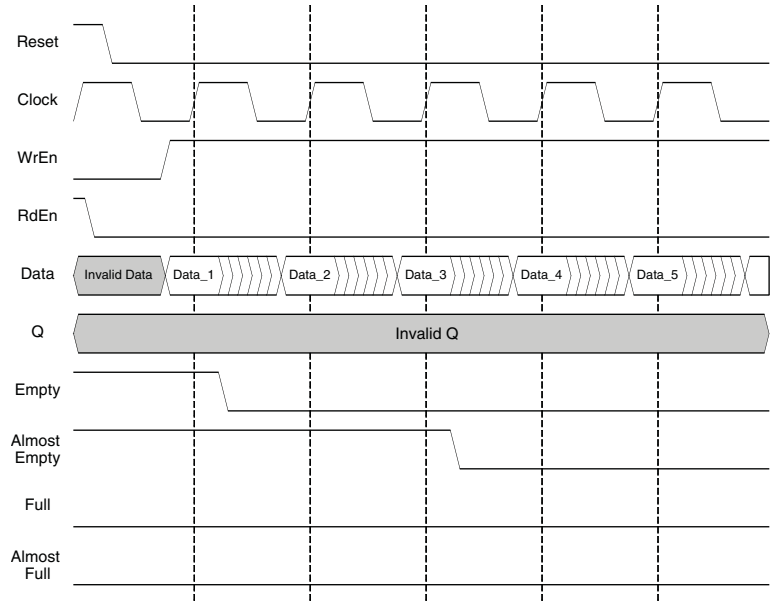


Figure 11-26. FIFO with Output Registers, End of Data Write Cycle

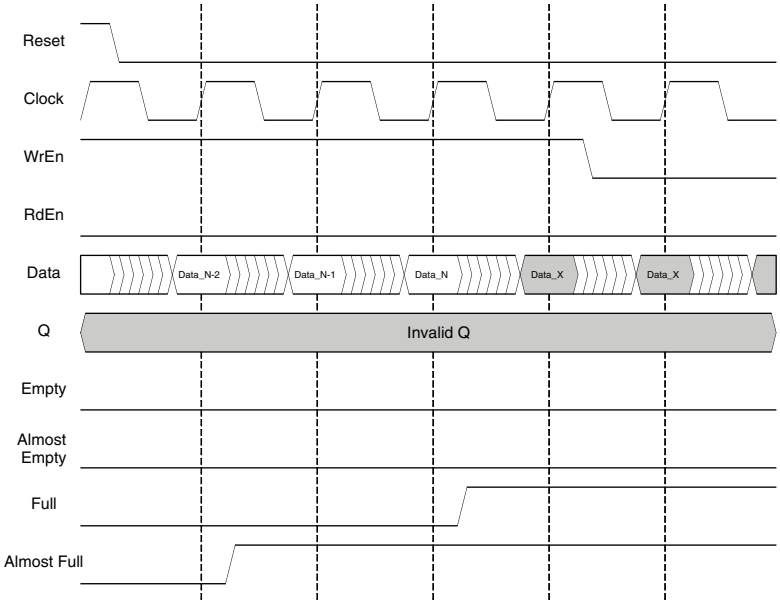


Figure 11-27. FIFO with Output Registers, Start of Data Read Cycle

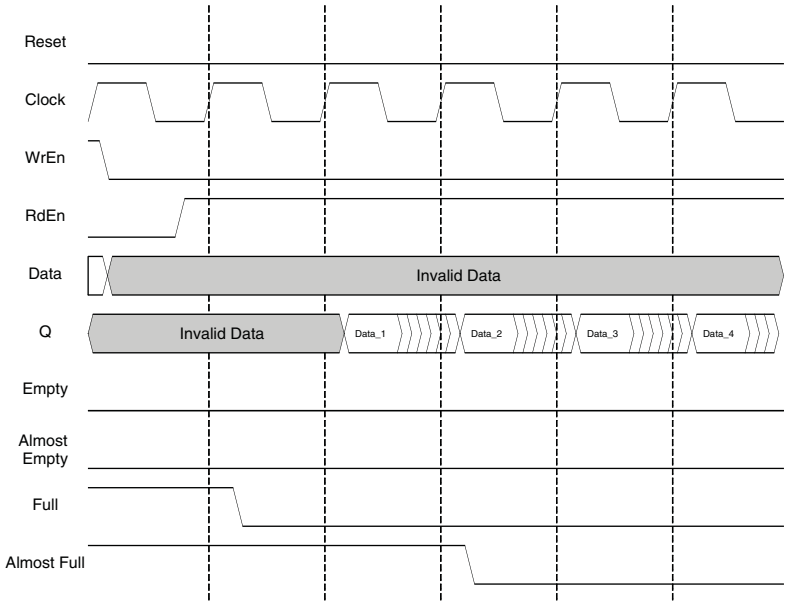
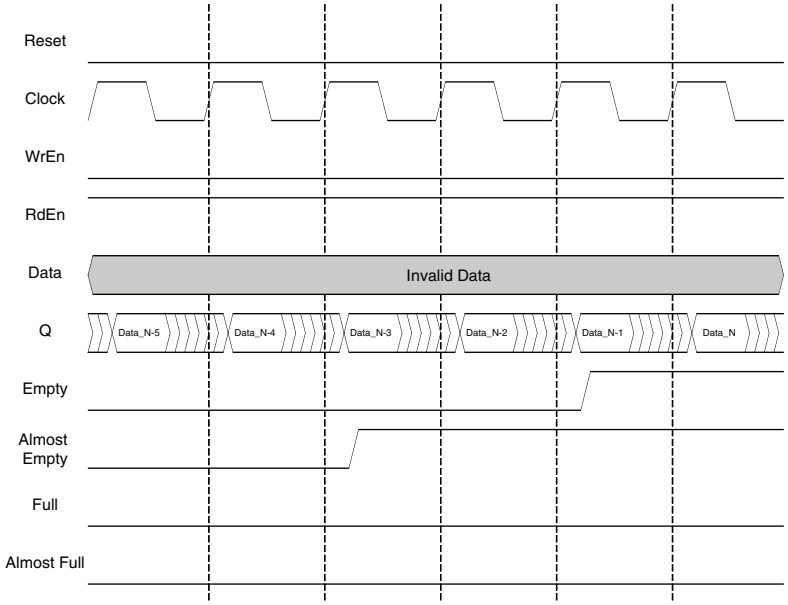
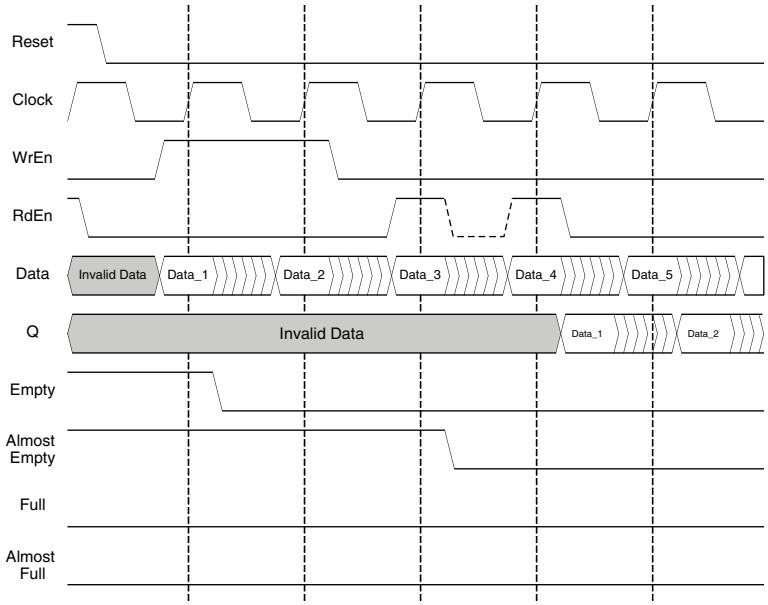


Figure 11-28. FIFO with Output Registers, End of Data Read Cycle



And finally, if you select the option enable output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO). The RdEn should also be high during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn goes true.

Figure 11-29. FIFO with Output Registers and RdEn on Output Registers



Dual Clock First In First Out (FIFO_DC) Memory:
The FIFO_DC or the dual clock FIFO is also an emulated FIFO. Again, the address logic and the flag logic is implemented in the FPGA fabric around the RAM.

The ports available on the FIFO_DC are:

- Reset
- RPReset
- WrClock
- RdClock
- WrEn
- RdEn
- Data
- Q
- Full Flag
- Almost Full Flag
- Empty Flag
- Almost Empty Flag

FIFO_DC Flags
The FIFO_DC, as an emulated FIFO, required the flags to be implemented in the FPGA logic around the block RAM. Because of the two clocks, the flags were required to change clock domains from read clock to write clock and vice versa. This adds latency to the flags either during assertion or de-assertion. The latency can be avoided only in one of the cases (either assertion or de-assertion) or distributed among these cases.

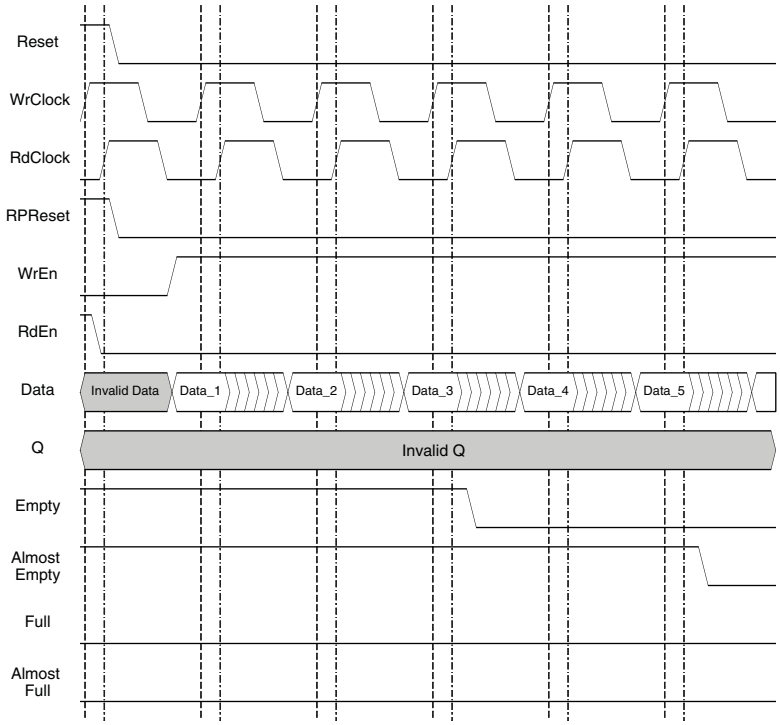
In the current emulated FIFO_DC, there is no latency during assertion of these flags which we feel is more important. Thus, when these flags are required to go true, there is no latency. However, due to the design of the flag logic running on two clock domains, there is latency during the de-assertion.

Let us assume that we start to write into the FIFO_DC to fill it. The write operation is controlled by WrClock and WrEn, however it takes extra RdClock cycles for de-assertion of the Empty and Almost Empty flags.

On the other hand, de-assertion of Full and Almost Full result in the reading out of the data from the FIFO_DC. It takes extra WrClock cycles, after reading this data, for the flags to come out.

With this in mind, let us look at the FIFO_DC without output registers waveforms. Figure 11-30 shows the operation of the FIFO_DC when it is empty and the data starts to be written into it.

Figure 11-30. FIFO_DC without Output Registers, Start of Data Write Cycle

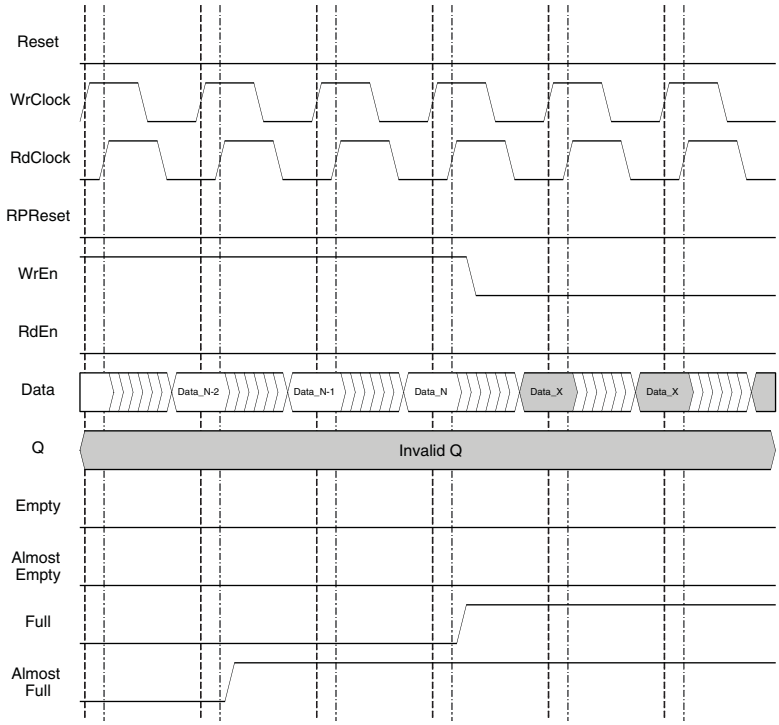


The WrEn signal must be high to start writing into the FIFO_DC. The Empty and Almost Empty flags are high to begin and Full and Almost full are low.

When the first data is written into the FIFO_DC, the Empty flag de-asserts (or goes low), as the FIFO_DC is no longer empty. In this figure we assume that the Almost Empty setting flag setting is 3 (address location 3). So the Almost Empty flag is de-asserted when the third address location is filled.

Now let us assume that we continue to write into the FIFO_DC to fill it. When the FIFO_DC is filled, the Almost Full and Full Flags are asserted. Figure 11-31 shows the behavior of these flags. In this figure the FIFO_DC depth is 'N'.

Figure 11-31. FIFO_DC without Output Registers, End of Data Write Cycle



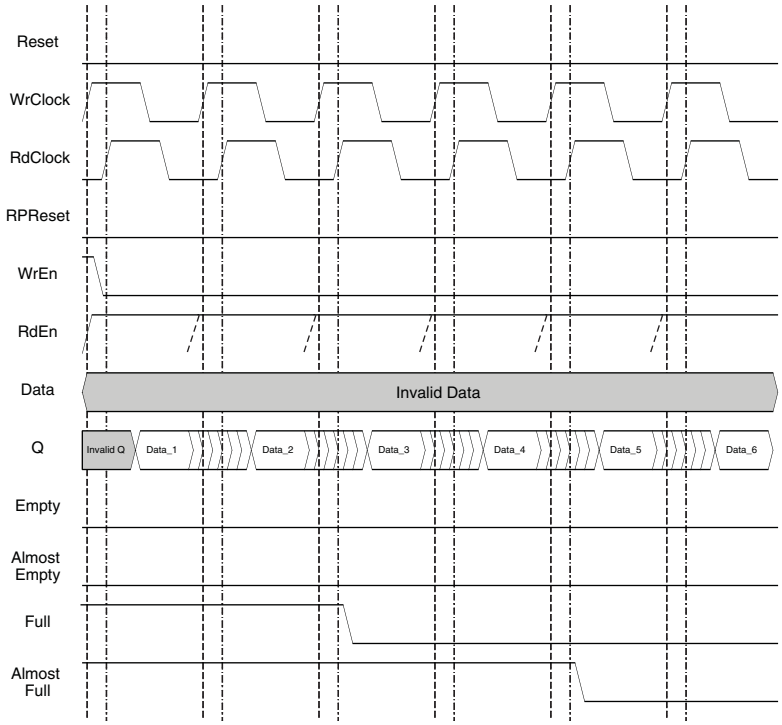
In this case, the Almost Full flag is in the 2 location before the FIFO_DC is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO_DC.

Data_X data inputs do not get written as the FIFO_DC is full (the Full flag is high).

Note that the assertion of these flags is immediate and there is no latency when they go true.

Now let us look at the waveforms when the contents of the FIFO_DC are read out. Figure 11-32 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted, as shown. In this case, note that the de-assertion is delayed by two clock cycles.

Figure 11-32. FIFO_DC without Output Registers, Start of Data Read Cycle



Similarly, as the data is read out, and FIFO_DC is emptied, the Almost Empty and Empty flags are asserted.

Figure 11-33. FIFO_DC without Output Registers, End of Data Read Cycle

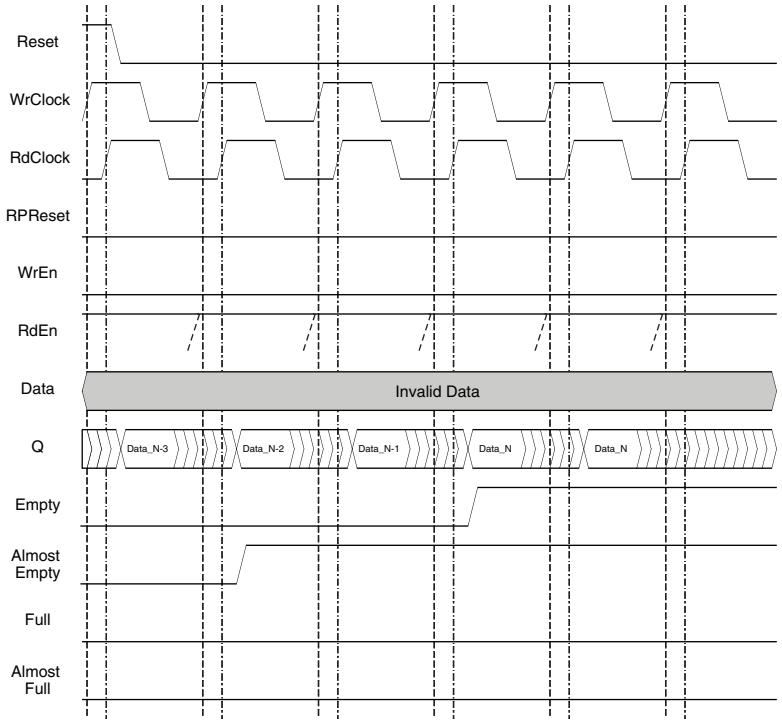


Figure 11-33 show the behavior of non-pipelined FIFO_DC or FIFO_DC without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is an extra option for the output registers to be enabled by the RdEn signal.

Figures 11-34 to 11-37 show the similar waveforms for the FIFO_DC with output register and without output register enable with RdEn. Note that flags are asserted and de-asserted with similar timing to the FIFO_DC without output registers. However, it is only the data out 'Q' that is delayed by one clock cycle.

Figure 11-34. FIFO_DC with Output Registers, Start of Data Write Cycle

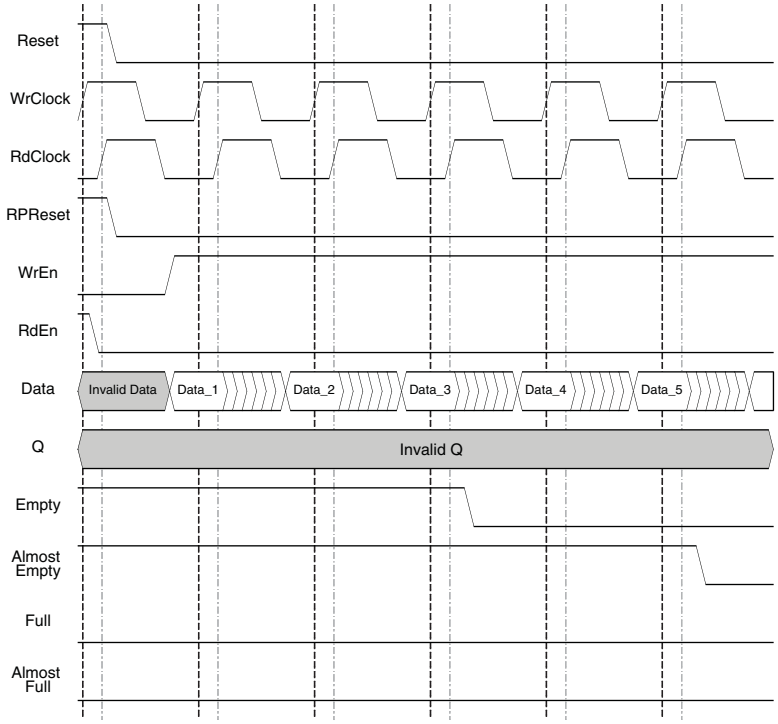


Figure 11-35. FIFO_DC with Output Registers, End of Data Write Cycle

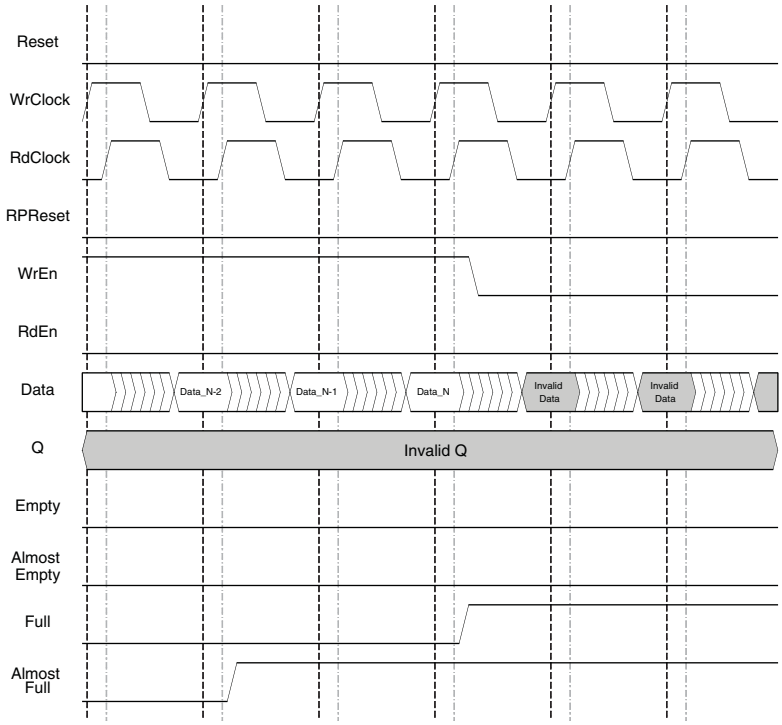


Figure 11-36. FIFO_DC with Output Registers, Start of Data Read Cycle

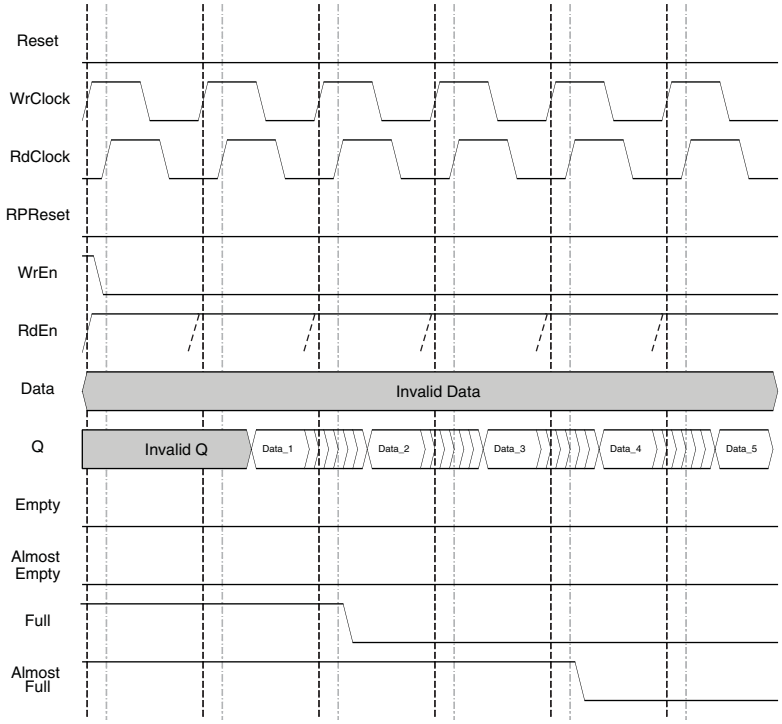
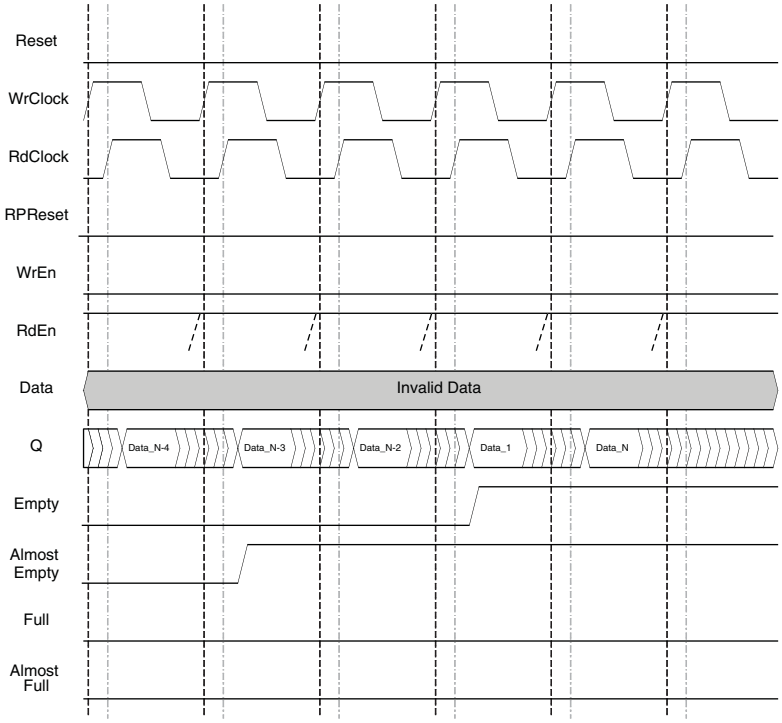
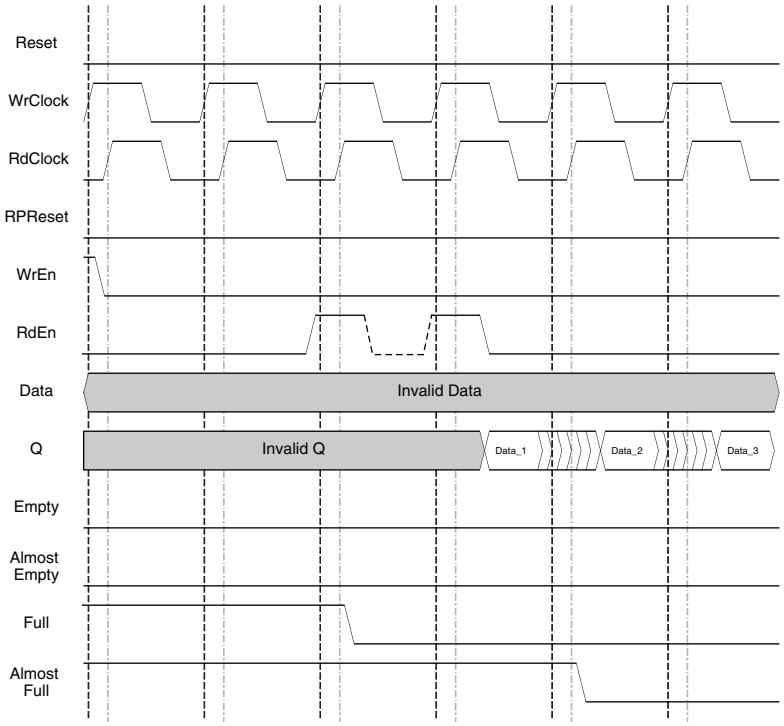


Figure 11-37. FIFO_DC with Output Registers, End of Data Read Cycle



And finally, if you select the option to enable the output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO_DC). The RdEn should also be high during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn is goes true.

Figure 11-38. FIFO_DC with Output Registers and RdEn on Output Registers

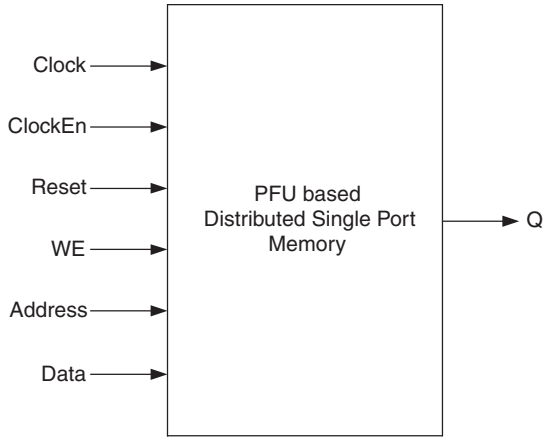


Distributed Single Port RAM (Distributed_SPRAM) – PFU Based

PFU-based Distributed Single Port RAM is created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger Distributed Memory sizes.

Figure 11-39 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 11-39. Distributed Single Port RAM Module Generated by IPexpress



The generated module makes use 4-input LUT available in the PFU. Additional logic like Clock, Reset is generated by utilizing the resources available in the PFU.

Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn), are not available in the hardware primitive. These are generated by IPexpress when the user wants the to enable the output registers in their IPexpress configuration.

The various ports and their definitions for the memory are as per Table 11-14. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 11-14. PFU-based Distributed Single Port RAM Port Definitions

Port Name in Generated Module	Port Name in the PFU Primitive	Description	Active State
Clock	CK	Clock	Rising Clock Edge
ClockEn	—	Clock Enable	Active High
Reset	—	Reset	Active High
WE	WRE	Write Enable	Active High
Address	AD[3:0]	Address	—
Data	DI[1:0]	Data In	—
Q	DO[1:0]	Data Out	—

Ports such as Clock Enable (ClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wishes to enable the output registers in the IPexpress configuration.

Users have the option of enabling the output registers for Distributed Single Port RAM (Distributed_SPRAM). Figures 11-40 and 11-41 show the internal timing waveforms for the Distributed Single Port RAM (Distributed_SPRAM) with these options.

Figure 11-40. PFU Based Distributed Single Port RAM Timing Waveform – without Output Registers

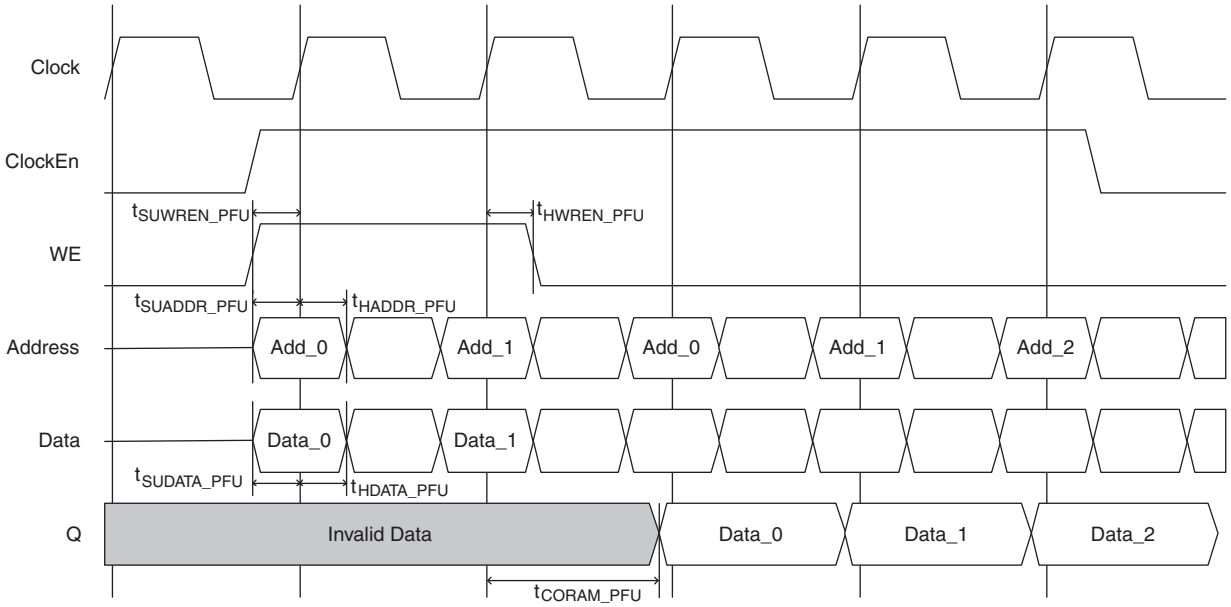
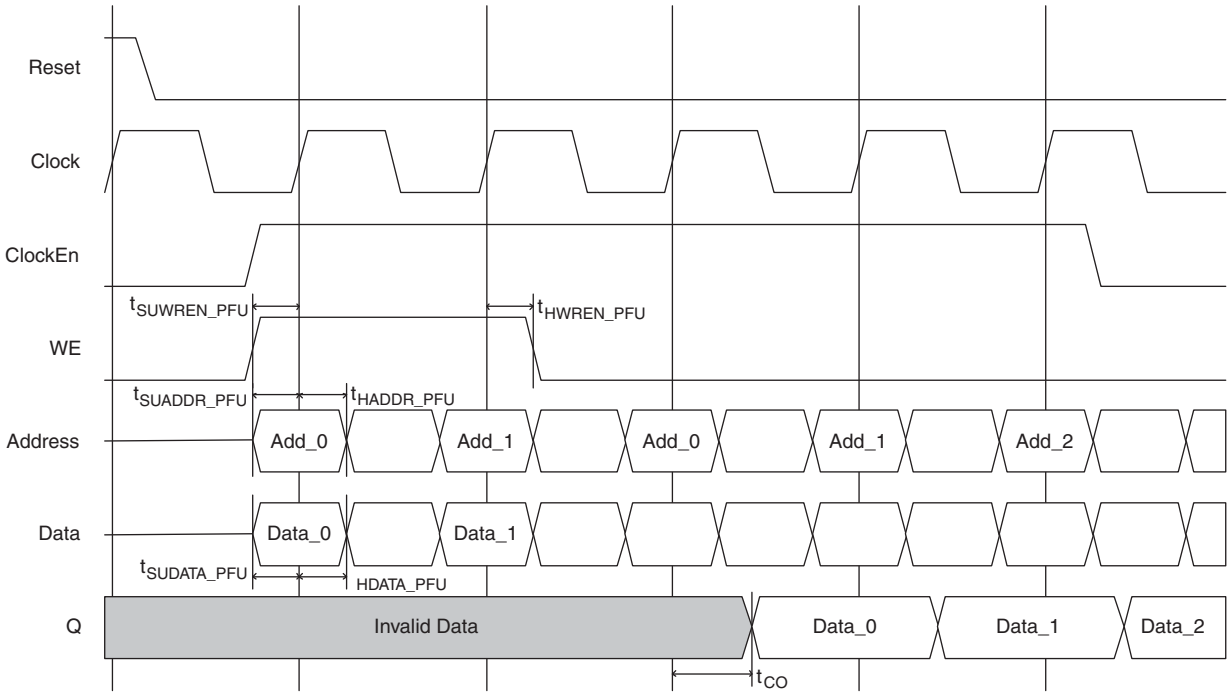


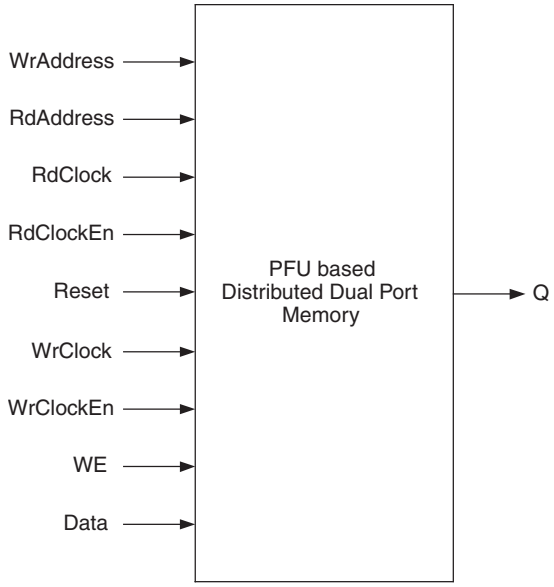
Figure 11-41. PFU Based Distributed Single Port RAM Timing Waveform – with Output Registers



Distributed Dual Port RAM (Distributed_DPRAM) – PFU Based

PFU-based Distributed Dual Port RAM is also created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create a larger Distributed Memory sizes.

Figure 11-42. Distributed Dual Port RAM Module Generated by IPexpress



The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock and Reset is generated by utilizing the resources available in the PFU.

Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn), are not available in the hardware primitive. These are generated by IPexpress when the user wants the to enable the output registers in the IPexpress configuration.

The various ports and their definitions for memory are as per Table 11-15. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 11-15. PFU-based Distributed Dual-Port RAM Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
WrAddress	WAD[3:0]	Write Address	—
RdAddress	RAD[3:0]	Read Address	—
RdClock	—	Read Clock	Rising Clock Edge
RdClockEn	—	Read Clock Enable	Active High
WrClock	WCK	Write Clock	Rising Clock Edge
WrClockEn	—	Write Clock Enable	Active High
WE	WRE	Write Enable	Active High
Data	DI[1:0]	Data Input	—
Q	RDO[1:0]	Data Out	—

Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants to enable the output registers in the IPexpress configuration.

Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed_DPRAM). Figures 11-43 and 11-44 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed_DPRAM) with these options.

Figure 11-43. PFU Based Distributed Dual Port RAM Timing Waveform – without Output Registers

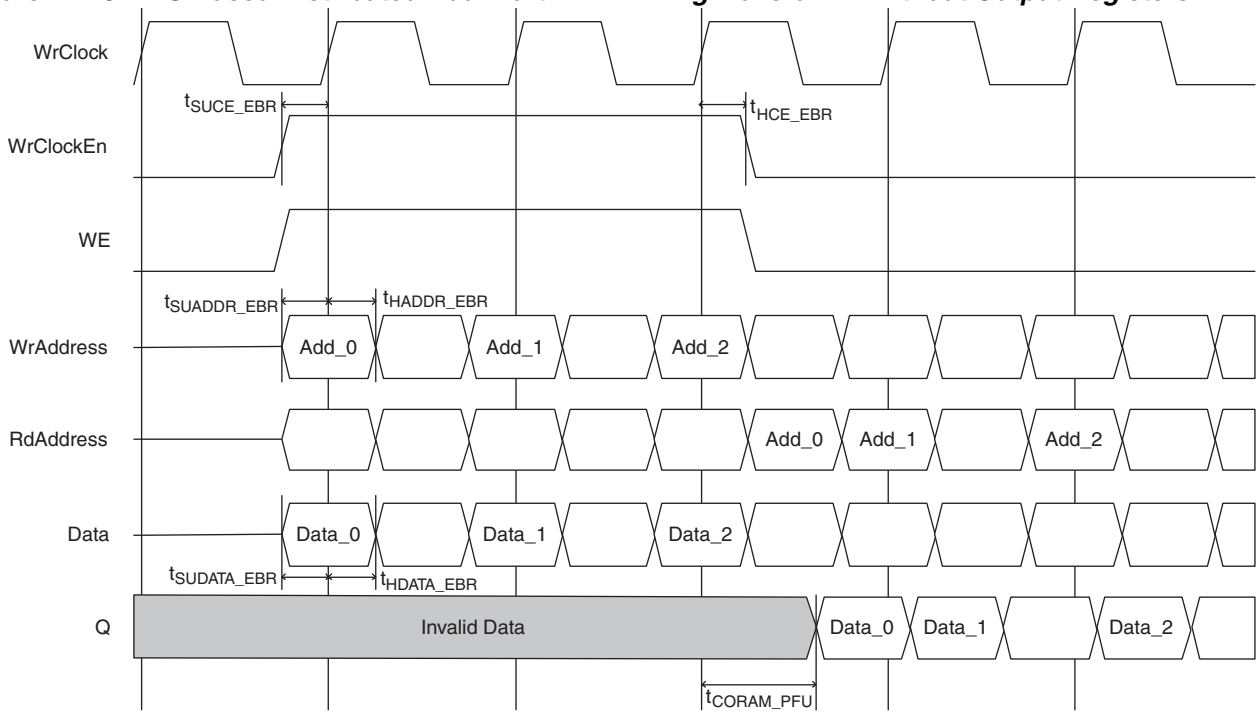
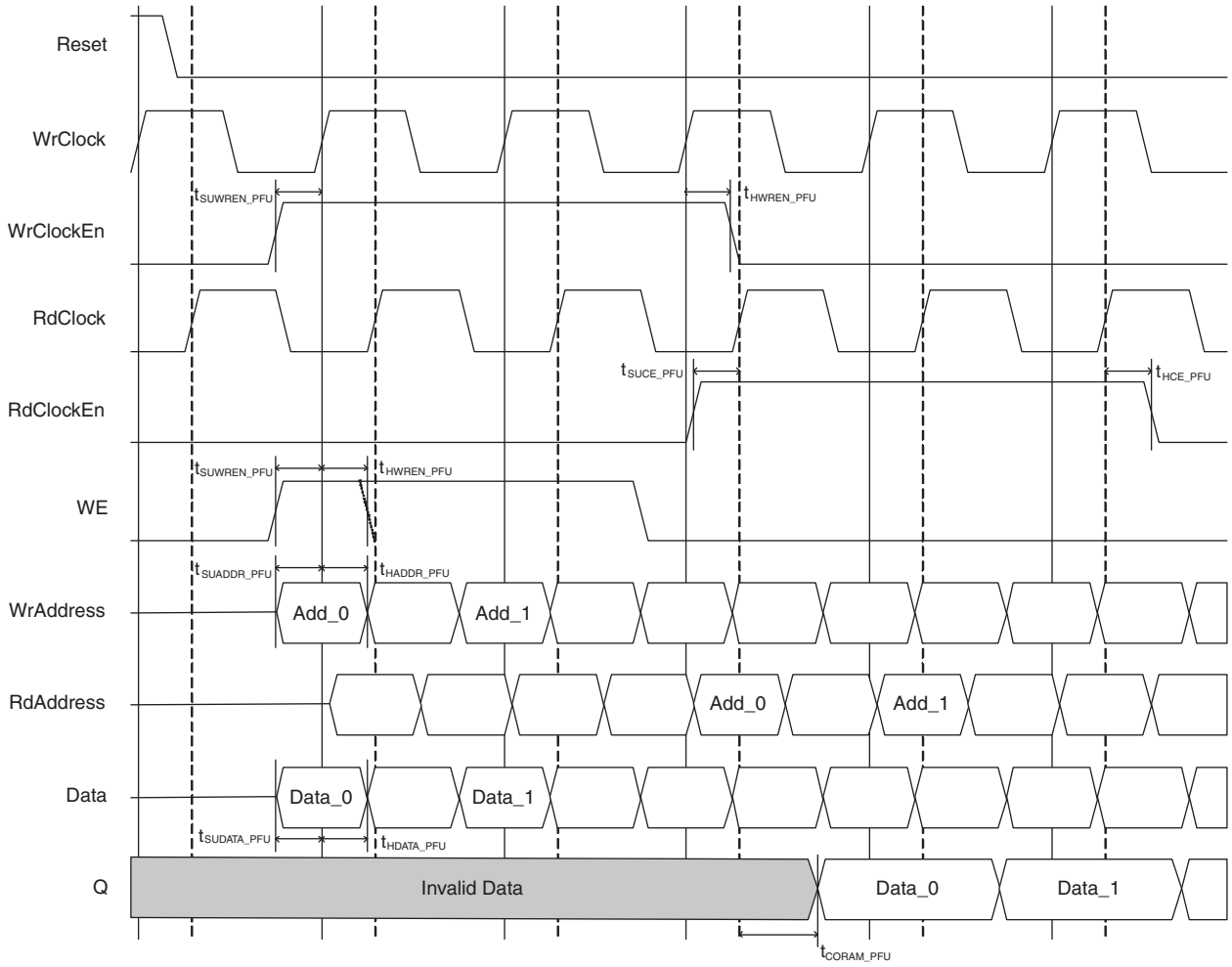


Figure 11-44. PFU Based Distributed Dual Port RAM Timing Waveform – with Output Registers

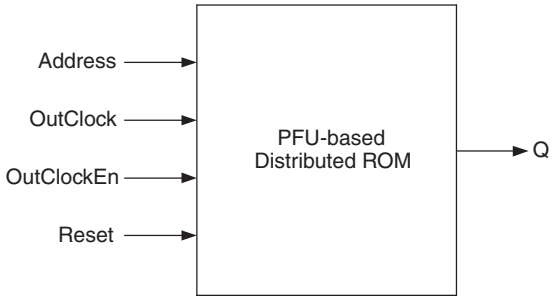


Distributed ROM (Distributed_ROM) – PFU Based

PFU-based Distributed ROM is also created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger Distributed Memory sizes.

Figure 11-45 shows the Distributed ROM module as generated by IPexpress.

Figure 11-45. Distributed ROM Generated by IPexpress



The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock and Reset is generated by utilizing the resources available in the PFU.

Ports such as Out Clock (OutClock) and Out Clock Enable (OutClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants to enable the output registers in the IPexpress configuration.

The various ports and their definitions for memory are as per Table 11-16. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 11-16. PFU-based Distributed ROM Port Definitions

Port Name in Generated Module	Port Name in the PFU Block Primitive	Description	Active State
Address	AD[3:0]	Address	—
OutClock	—	Out Clock	Rising Clock Edge
OutClockEn	—	Out Clock Enable	Active High
Reset	—	Reset	Active High
Q	DO	Data Out	—

Users have the option to enable the output registers for Distributed ROM (Distributed_ROM). Figures 11-46 and 11-47 show the internal timing waveforms for the Distributed ROM with these options.

Figure 11-46. PFU Based ROM Timing Waveform – without Output Registers

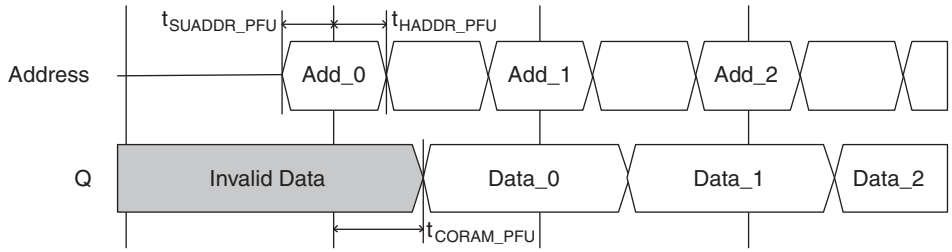
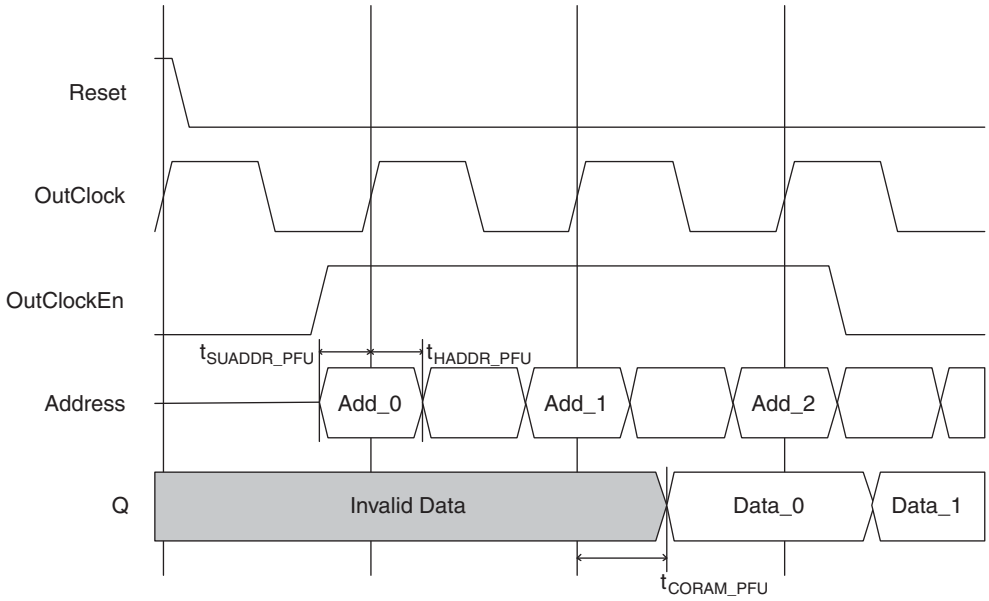


Figure 11-47. PFU Based ROM Timing Waveform – with Output Registers



Initializing Memory

In the EBR based ROM or RAM memory modes and the PFU based ROM memory mode, it is possible to specify the power-on state of each bit in the memory array. Each bit in the memory array can have one of two values: 0 or 1.

Initialization File Format

The initialization file is an ASCII file, which users can create or edit using any ASCII editor. IPexpress supports three types of memory file formats:

- Binary file
- Hex File
- Addressed Hex

The file name for the memory initialization file is *.mem (<file_name>.mem). Each row depicts the value to be stored in a particular memory location and the number of characters (or the number of columns) represents the number of bits for each address (or the width of the memory module).

The Initialization File is primarily used for configuring the ROMs. The EBR in RAM mode can optionally use this Initialization File also to preload the memory contents.

Binary File

The file is essentially a text file of 0's and 1's. The rows indicate the number of words and columns indicate the width of the memory.

```
Memory Size 20x32
00100000010000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
00000111000001110000011100000111
00001000010010000000100001001000
00001001010010010000100101001001
00001010010010100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
00001110001111100000111000111110
00001111001111110000111100111111
00010000000100000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011
```

Hex File

The Hex file is essentially a text file of Hex characters arranged in a similar row-column arrangement. The number of rows in the file is same as the number of address locations, with each row indicating the content of the memory location.

```
Memory Size 8x16
A001
0B03
1004
CE06
0007
040A
0017
02A4
```

Addressed Hex

Addressed Hex consists of lines of address and data. Each line starts with an address, followed by a colon, and any number of data. The format of memfile is address: data data data data ... where address and data are hexadecimal numbers.

```
-A0 : 03 F3 3E 4F
-B2 : 3B 9F
```

The first line puts 03 at address A0, F3 at address A1, 3E at address A2,and 4F at address A3. The second line puts 3B at address B2 and 9F at address B3.

There is no limitation on the values of address and data. The value range is automatically checked based on the values of addr_width and data_width. If there is an error in an address or data value, an error message is printed. Users need not specify data at all address locations. If data is not specified at certain address, the data at that location is initialized to 0. IPexpress makes memory initialization possible both through the synthesis and simulation flows.

Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
February 2006	01.0	Initial release.
April 2006	01.1	Updated the Initializing Memory section
September 2006	01.2	Added LatticeECP2M device information. Added dual port memory access notes.
April 2007	01.3	Updated Utilizing IPexpress section.
February 2008	01.4	Updated Pseudo-Dual Port RAM Attributes for LatticeECP2/M table.
March 2008	01.5	Updated FIFO_DC without Output Registers (Non-Pipelined) figure.
June 2008	01.6	Updated First In First Out (FIFO, FIFO_DC) – EBR Based section.
		Removed Read-Before-Write sysMEM EBR mode.
August 2008	01.7	Corrected AddressA, AddressB information in EBR-based True Dual Port Memory Port Definitions table.
September 2008	01.8	Updated IPexpress Flow text section.
March 2009	01.9	Updated Memory Modules text section.
July 2011	02.0	Added the setup and hold requirements for addresses to EBR-based memories.
June 2013	02.1	Updated document with new corporate logo.
		Updated Technical Support Assistance information.

Appendix A. Attribute Definitions

DATA_WIDTH

Data width is associated with the RAM and FIFO elements. The DATA_WIDTH attribute will define the number of bits in each word. It takes the values as defined in the RAM size tables in each memory module.

REGMODE

REGMODE or the Register mode attribute is used to enable pipelining in the memory. This attribute is associated with the RAM and FIFO elements. The REGMODE attribute takes the NOREG or OUTREG mode parameter that disables and enables the output pipeline registers.

RESETMODE

The RESETMODE attribute allows users to select the mode of reset in the memory. This attribute is associated with the block RAM elements. RESETMODE takes two parameters: SYNC and ASYNC. SYNC means that the memory reset is synchronized with the clock. ASYNC means that the memory reset is asynchronous to clock.

CSDECODE

CSDECODE or the Chip Select Decode attributes are associated to block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily. CSDECODE takes the following parameters: “000”, “001”, “010”, “011”, “100”, “101”, “110”, and “111”. CSDECODE values determine the decoding value of CS[2:0]. CSDECODE_W is chip select decode for write and CSDECODE_R is chip select decode for read for Pseudo Dual Port RAM. CSDECODE_A and CSDECODE_B are used for true dual port RAM elements and refer to the A and B ports.

WRITEMODE

The WRITEMODE attribute is associated with the block RAM elements. It takes the NORMAL and WRITETHROUGH mode parameters.

In NORMAL mode, the output data does not change or get updated, during the write operation. This mode is supported for all data widths.

In WRITETHROUGH mode, the output data is updated with the input data during the write cycle. This mode is supported for all data widths.

WRITEMODE_A and WRITEMODE_B are used for dual port RAM elements and refer to the A and B ports in case of a True Dual Port RAM.

For all modes (of the True Dual Port module), simultaneous read access from one port and write access from the other port to the same memory address is not recommended. The read data may be unknown in this situation. Also, simultaneous write access to the same address from both ports is not recommended. (When this occurs, the data stored in the address becomes undetermined when one port tries to write a ‘H’ and the other tries to write a ‘L’).

It is recommended that the designer implements control logic to identify this situation if it occurs and either:

1. Implement status signals to flag the read data as possibly invalid, or
2. Implement control logic to prevent the simultaneous access from both ports.

GSR

GSR or the Global Set/ Reset attribute is used to enable or disable the global set/reset for RAM element.