Memory Usage Guide for LatticeECP/EC and LatticeXP Devices

September 2012 Technical Note TN1051

Introduction

This technical note discusses memory usage in the LatticeEC™, LatticeECP™ and LatticeXP™ device families. It is intended to be used as a guide for integrating the EBR and PFU based memories for these device families using the ispLEVER® design tool.

The architecture of the LatticeECP/EC and LatticeXP devices provides a large amount of resources for memory intensive applications. The sysMEM™ Embedded Block RAM (EBR) complements its distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM and ROM memories can be constructed using the EBR. LUTs and PFU can implement Distributed Single-Port RAM, Dual-Port RAM and ROM. The internal logic of the device can be used to configure the memory elements as FIFO and other storage types.

The capabilities of the EBR Block RAM and PFU RAM are referred to as primitives and described later in this document. Designers can generate the memory primitives using 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 how to utilize IPexpress, memory modules and memory primitives.

Memories in LatticeECP/EC and LatticeXP Devices

The LatticeECP/EC and LatticeXP architectures contain an array of logic blocks called PFUs or PFFs surrounded by Programmable I/O Cells (PICs). Interspersed between the rows of logic blocks are rows of sysMEM Embedded Block RAM (EBR) as shown in Figures 9-1, 9-2 and 9-3.

The PFU contains the building blocks for logic, and Distributed RAM and ROM. The PFF provides the logic building blocks without the distributed RAM

This document describes the memory usage and implementation for both embedded memory blocks (EBR) and distributed RAM of the PFU. Refer to the device data sheet for details on the hardware implementation of the EBR and Distributed RAM.

The logic blocks are arranged in a two-dimensional grid with rows and columns as shown in the figures below. The physical location of the EBR and Distributed RAM follows the row and column designation. The Distributed RAM, since it is part of the PFU resource, follows the PFU/PFF row and column designation. The EBR occupies two columns per block to account for the wider port interface.

Figure 9-1. Simplified Block Diagram, LatticeEC Device (Top Level)

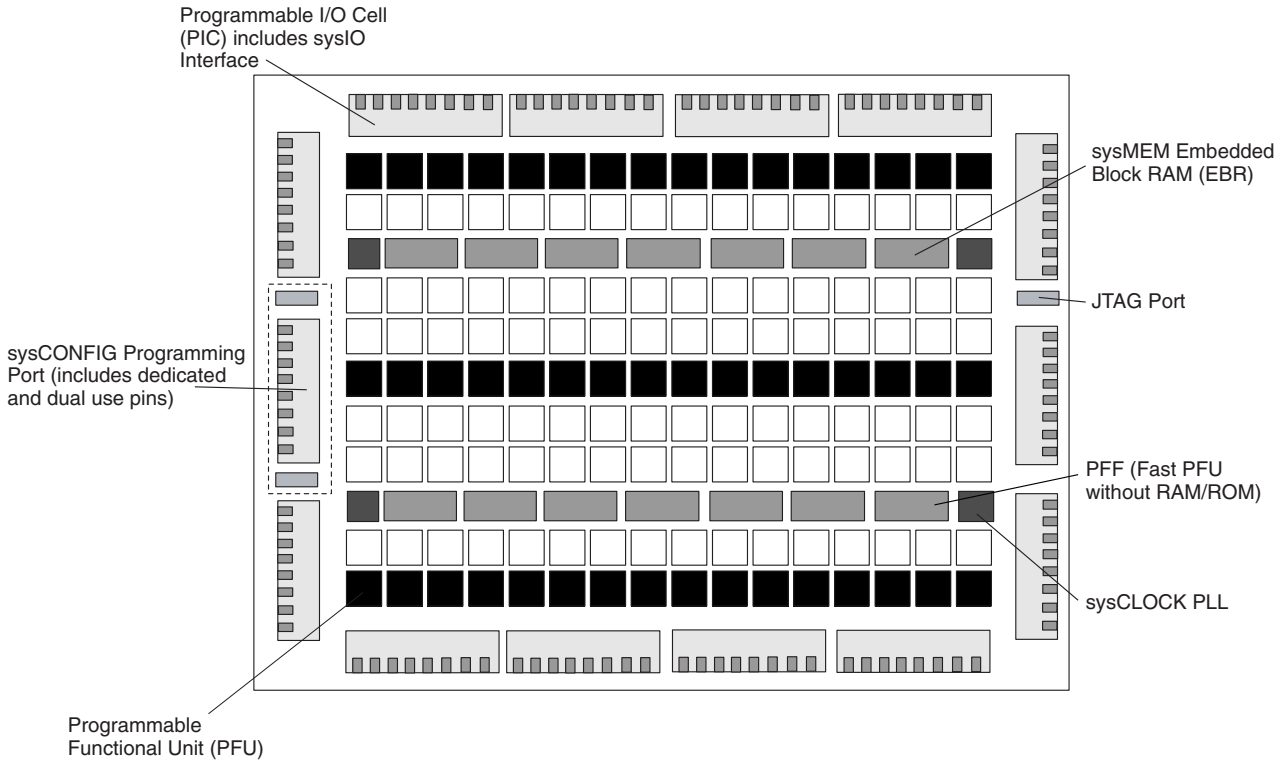


Figure 9-2. Simplified Block Diagram, LatticeECP Device (Top Level)

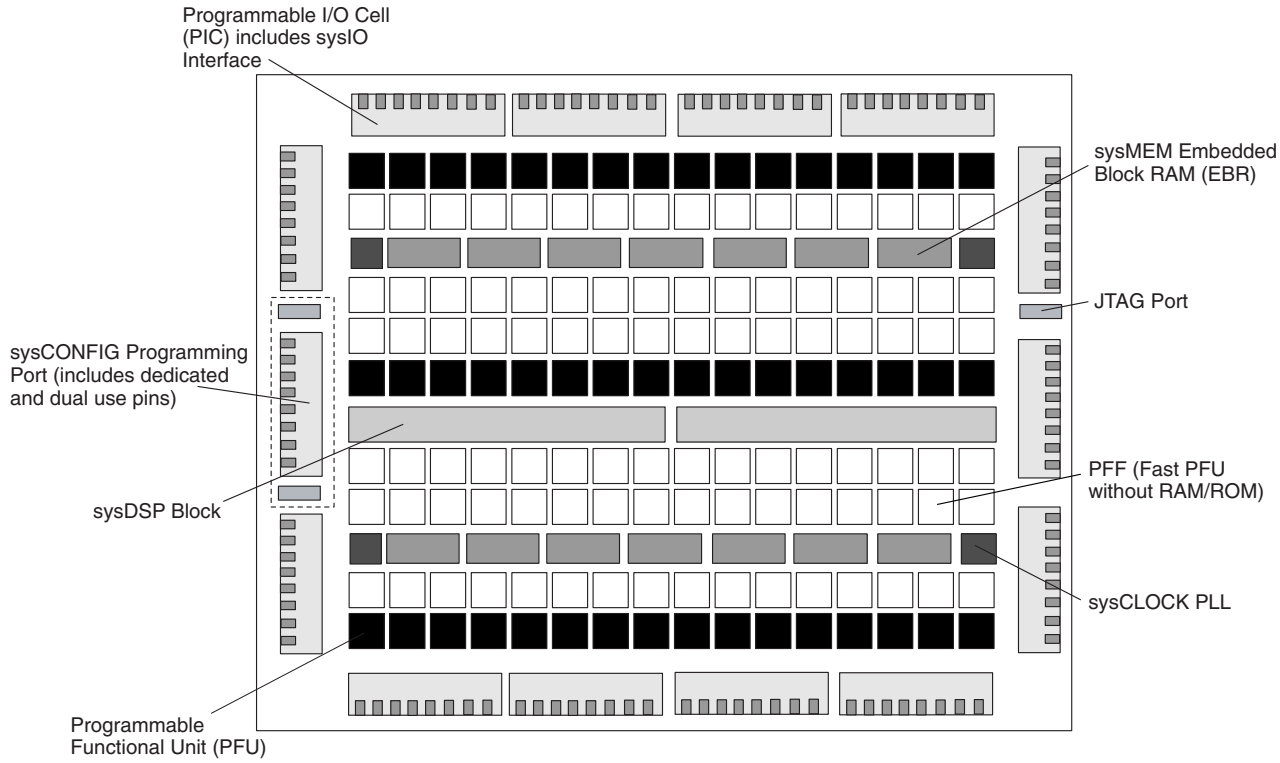
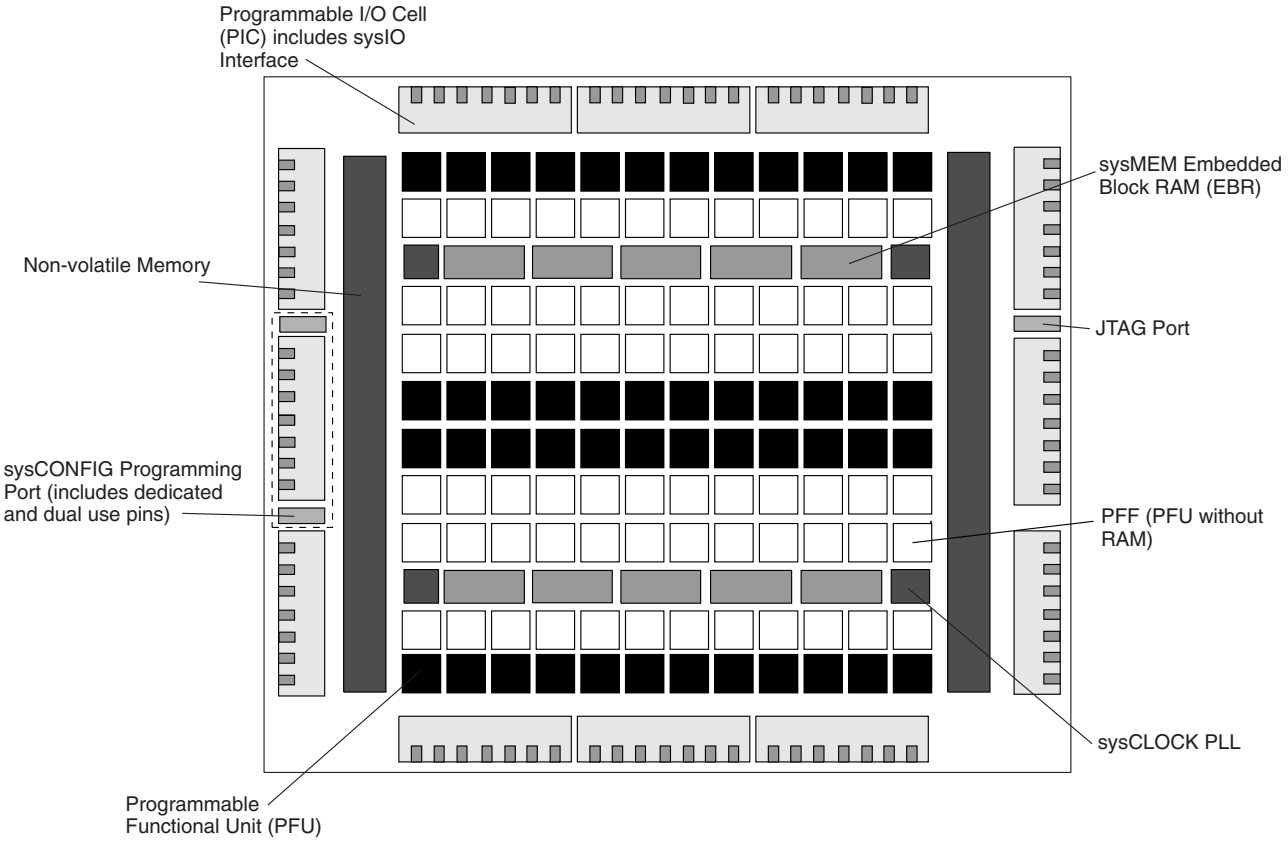


Figure 9-3. Simplified Block Diagram, LatticeXP Device (Top Level)



Utilizing IPexpress

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

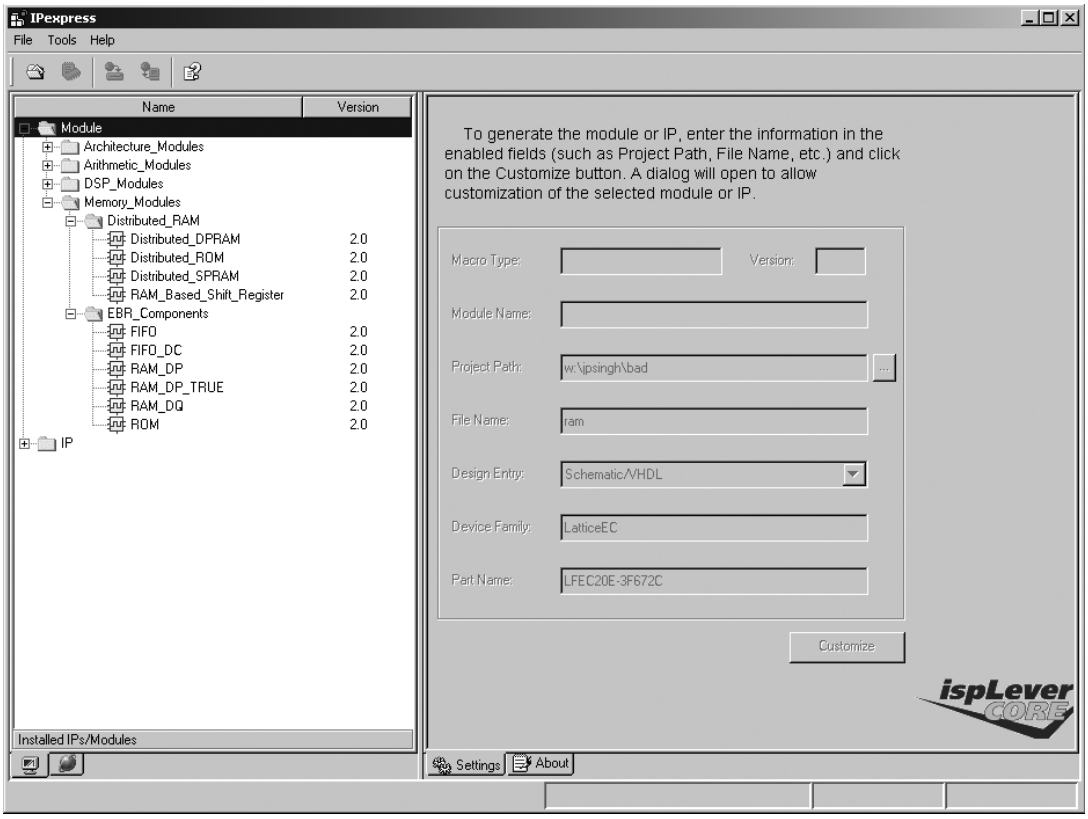
IPexpress Flow

For generating any of these memories, create (or open) a project for the LatticeECP/EC or LatticeXP devices.

From the Project Navigator, select **Tools > IPexpress**. Alternatively, users can also click on the button in the toolbar when the LatticeECP/EC and LatticeXP devices are targeted in the project.

This opens the IPexpress window as shown in Figure 9-4.

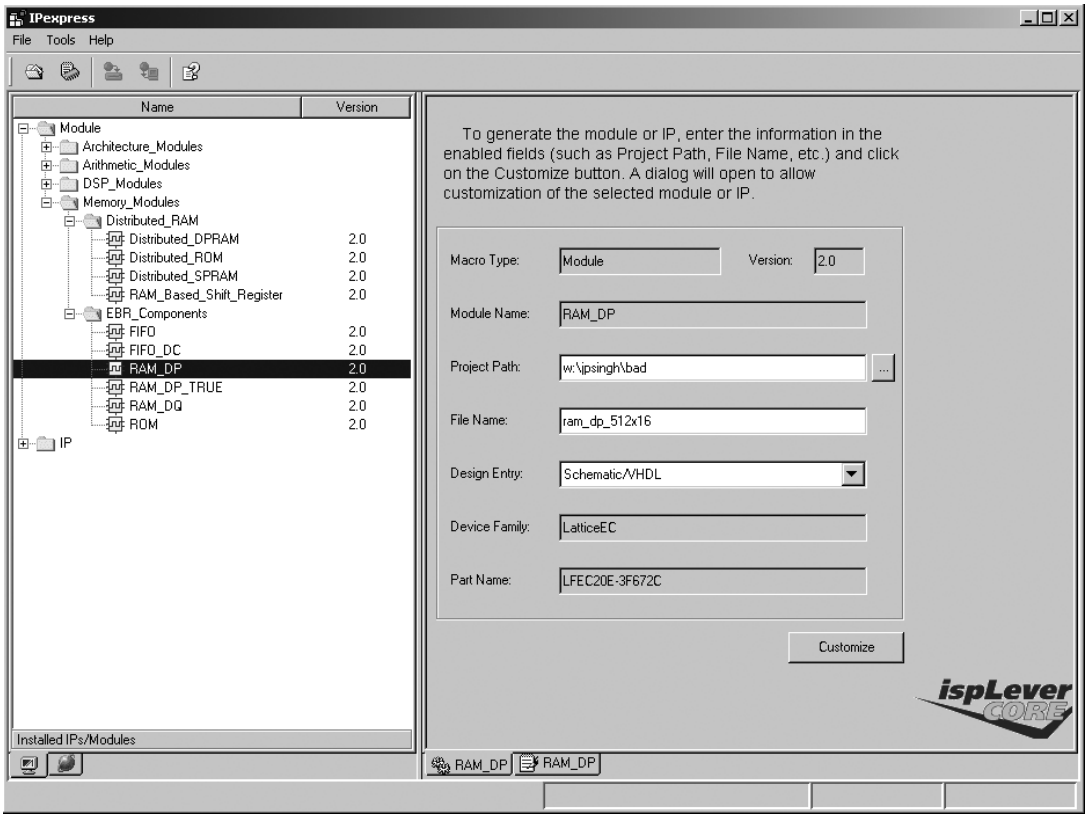
Figure 9-4. IPexpress - Main Window



The left pane of this window has the Module Tree. The EBR-based Memory Modules are under the **Module > Memory- Module > Distributed RAM and EBR_Components** and the PFU-based Distributed Memory Modules are under **Storage_Components** as shown in Figure 9-4.

Let us look at an example of the generating an EBR-based Pseudo Dual Port RAM of size 512 x 16. Select RAM_DP under the EBR_Components. The right pane changes, as shown in Figure 9-5.

Figure 9-5. Example Generating Pseudo Dual Port RAM (RAM_DP) Using IPexpress



In the right-hand pane, options like **Macro Type**, **Version**, 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**.

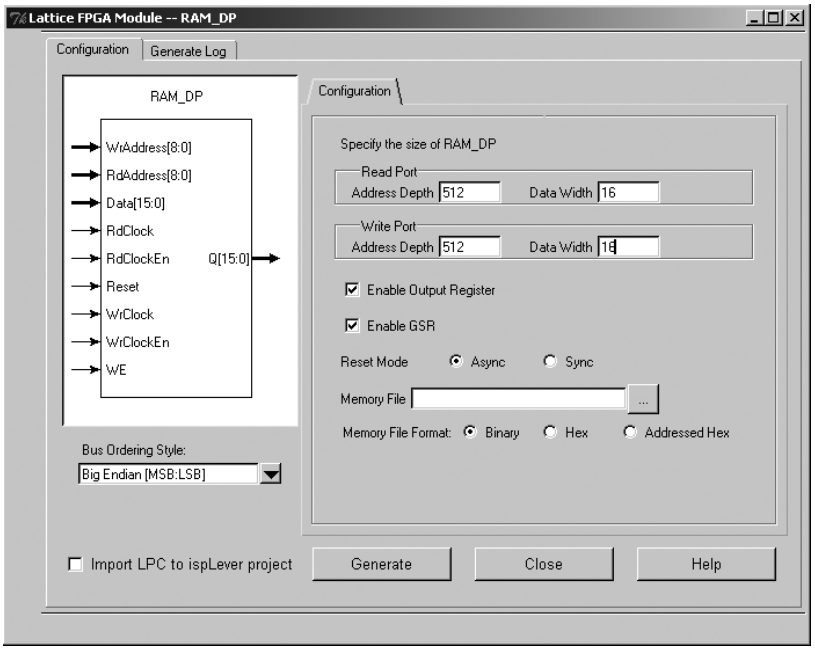
The **File Name** text box allows users to specify the entity and file name for the module they are about to generate. Users must provide this name.

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.

Then click the **Customize** button. This opens another window where the RAM can be customized.

The the left-hand side of this window shows the block diagram of the module. The right-hand side includes the Configuration tab.

Figure 9-6. Generating Pseudo Dual Port RAM (RAM_DP) Module Customization – Configuration Tab



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 in the case of Pseudo Dual Port and True Dual Port RAMs.

The check box **Enable Output Registers** inserts the output registers in the Read Data Port, as the output registers are optional for the EBR-based RAMs.

The **Reset Mode** can be selected to be Asynchronous Reset or Synchronous Reset. **GSR** or Global Set Reset can be checked to be Enabled or Disabled.

The Input Data and the Address Control is always registered, as the hardware only supports synchronous operation for the EBR based RAMs

Users can also pre-initialize their memory with the contents they specify in the Memory file. It is optional to provide this file in the RAMs. However, in the case of ROM, it is required to provide the Memory file. 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 technical note.

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

Users can check the Import LPC to ispLEVER project check box to automatically import the file in the Project Navigator.

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

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

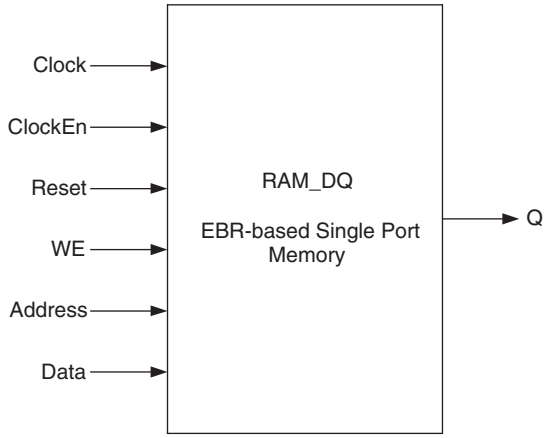
Memory Modules

Single Port RAM (RAM_DQ) – EBR Based

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

IPexpress generates the memory module as shown in Figure 9-7.

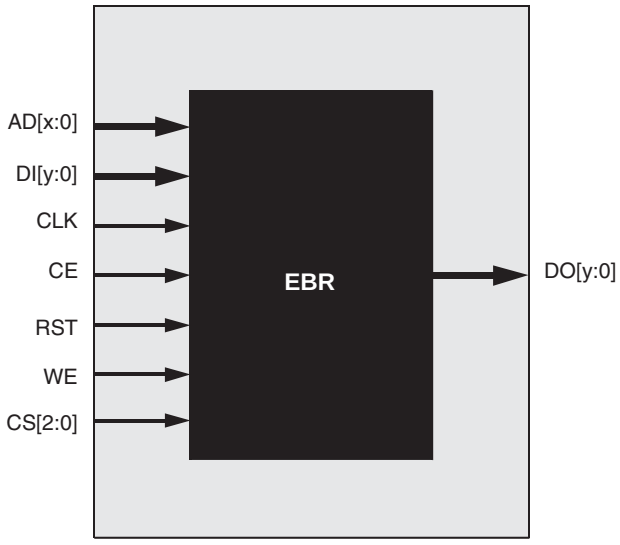
Figure 9-7. Single Port Memory Module generated by IPexpress



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. In cases where the specified memory is larger than one EBR block, multiple EBR block can be cascaded, in depth or width (as required to create these sizes).

The memory primitive for RAM_DQ for LatticeECP/EC and LatticeXP devices is shown in Figure 9-8.

Figure 9-8. Single Port RAM Primitive or RAM_DQ for LatticeECP/EC and LatticeXP Devices



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.

The various ports and their definitions for the Single Port Memory are included in Table 9-1. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DQ primitive.

Table 9-1. 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) only resets the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, a port available in the EBR primitive, is useful when memory requires multiple EBR blocks to be 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 easily cascade eight memories. If the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 9,216 bits of RAM. The values for x (for Address) and y (Data) for each EBR block for the devices are included in Table 9-2.

Table 9-2. Single Port Memory Sizes for 9K Memories for LatticeECP/EC Devices

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
8K x 1	DI	DO	AD[12:0]
4K x 2	DI[1:0]	DO[1:0]	AD[11:0]
2K x 4	DI[3:0]	DO[3:0]	AD[10:0]
1K x 9	DI[8:0]	DO[8:0]	AD[9:0]
512 x 18	DI[17:0]	DO[17:0]	AD[8:0]
256 x 36	DI[35:0]	DO[35:0]	AD[7:0]

Table 9-3 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 9-3. Single Port RAM Attributes for LatticeECP/EC Devices

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH	Data Word Width	1, 2, 4, 9, 18, 36	1	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE	Chip Select Decode	000, 001, 010, 011, 100, 101, 110, 111	000	NO
WRITEMODE	Read / Write Mode	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
GSR	Global Set Reset	ENABLED, DISABLED	ENABLED	YES

The Single Port RAM (RAM_DQ) can be configured as NORMAL, READ BEFORE WRITE 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 READ BEFORE WRITE attribute is supported for x9, x18 and x36 data widths.

Additionally users can select to enable the output registers for RAM_DQ. Figures 8-7 through 8-12 show the internal timing waveforms for the Single Port RAM (RAM_DQ) with these options.

Figure 9-9. Single Port RAM Timing Waveform – NORMAL Mode, without Output Registers

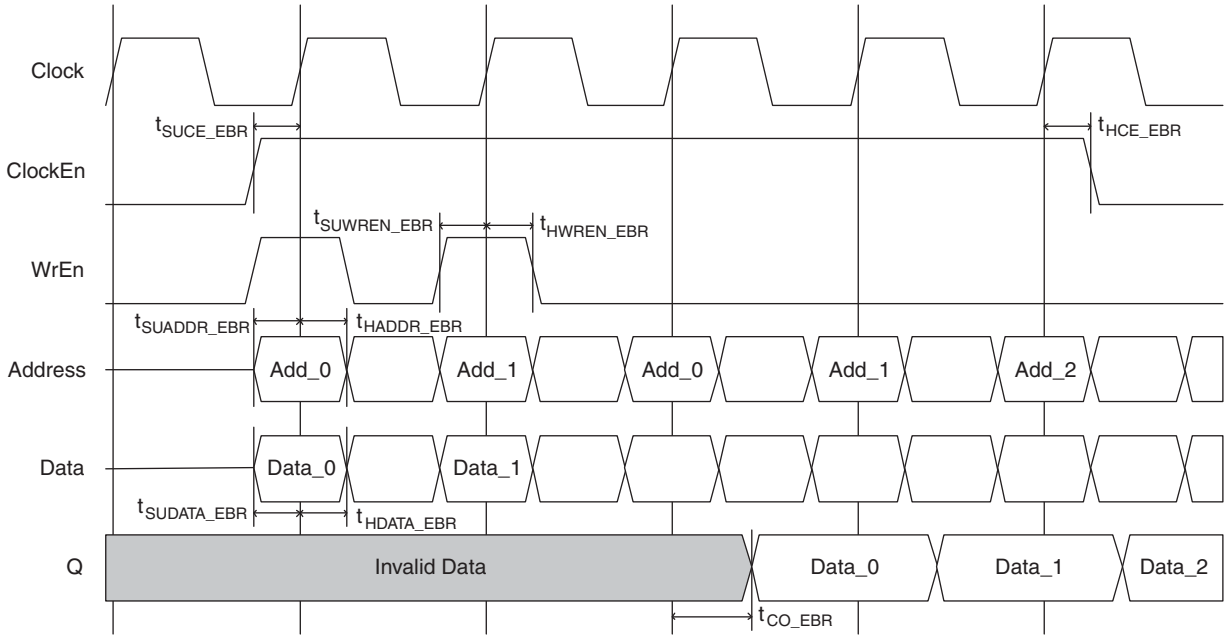


Figure 9-10. Single Port RAM Timing Waveform – NORMAL Mode, with Output Registers

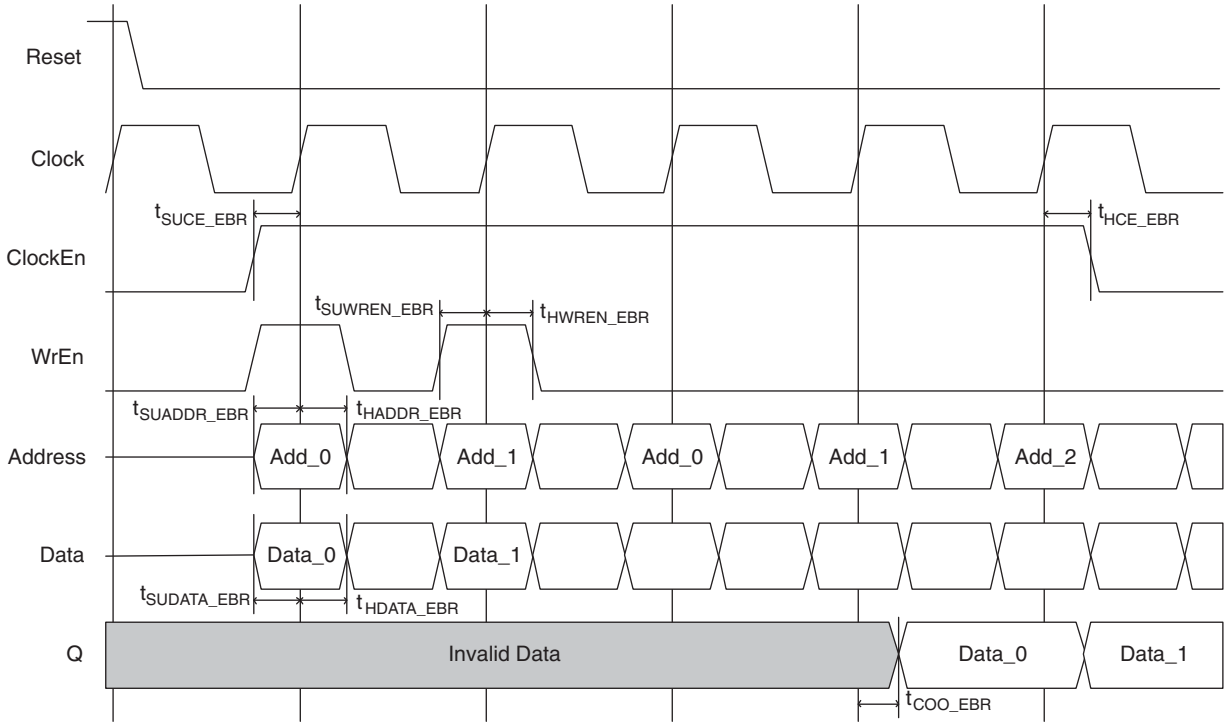


Figure 9-11. Single Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers

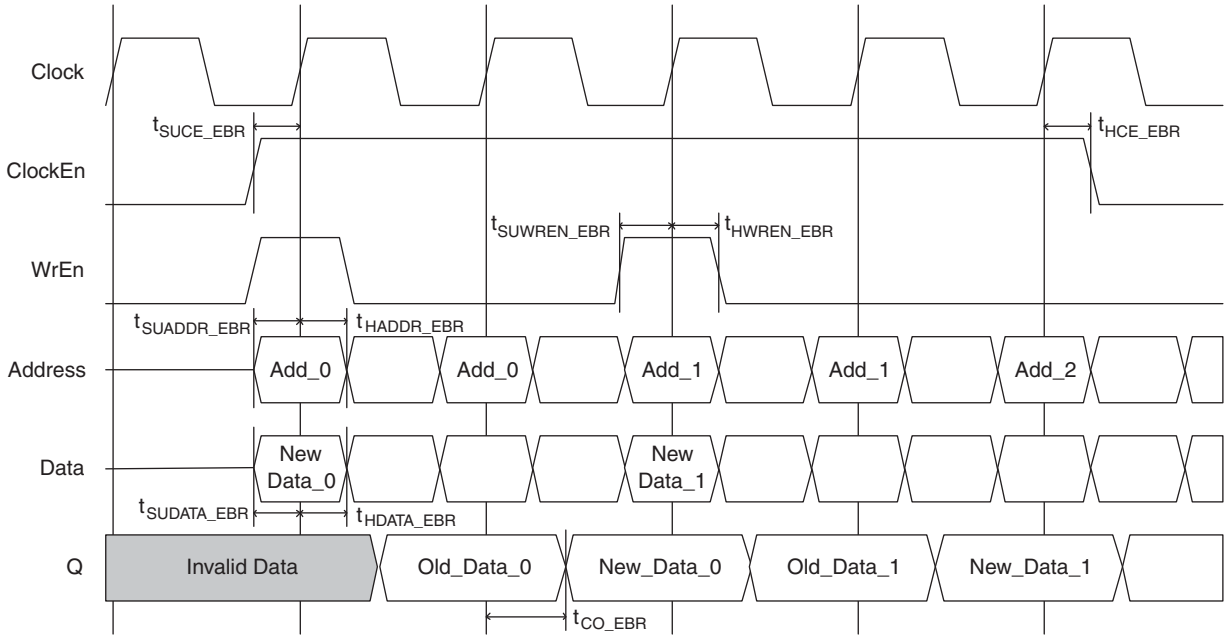


Figure 9-12. Single Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers

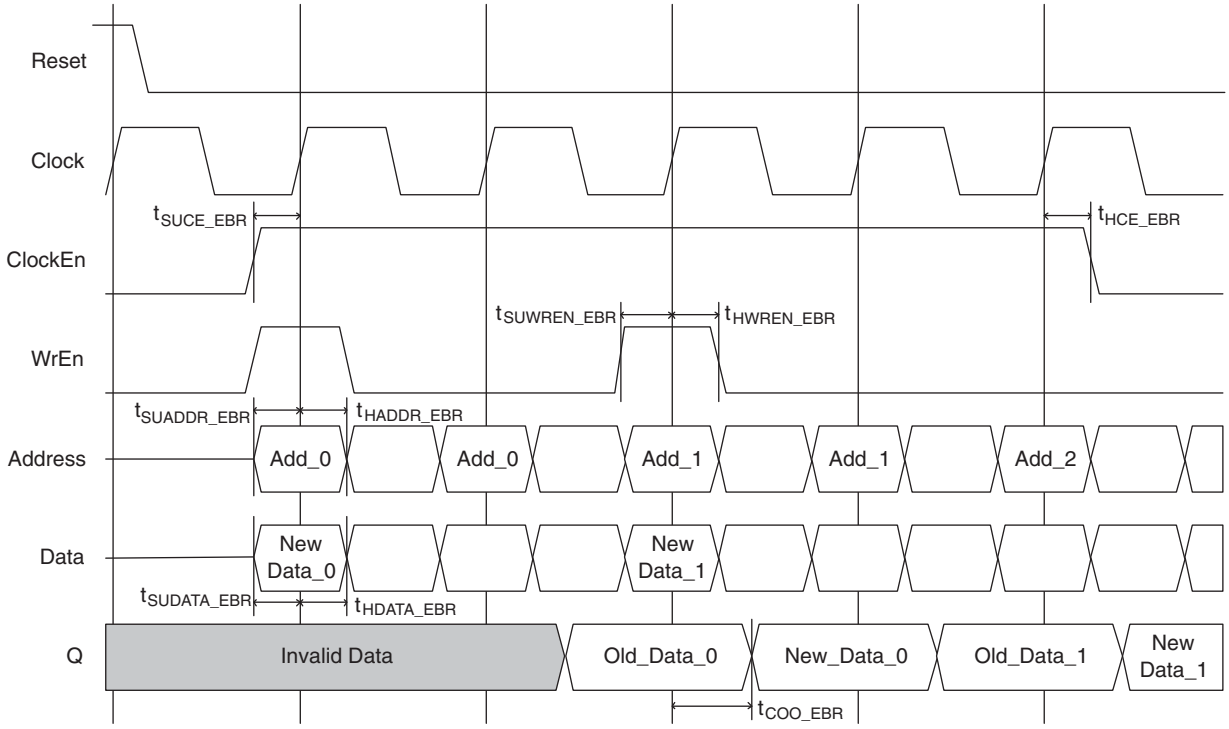


Figure 9-13. Single Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

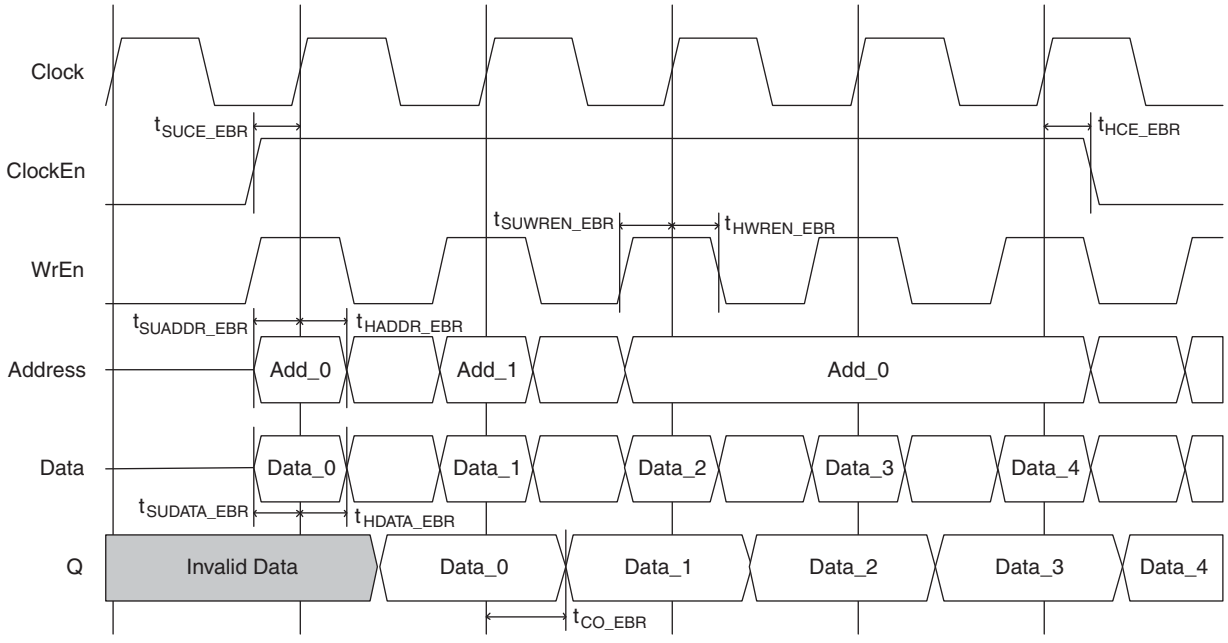
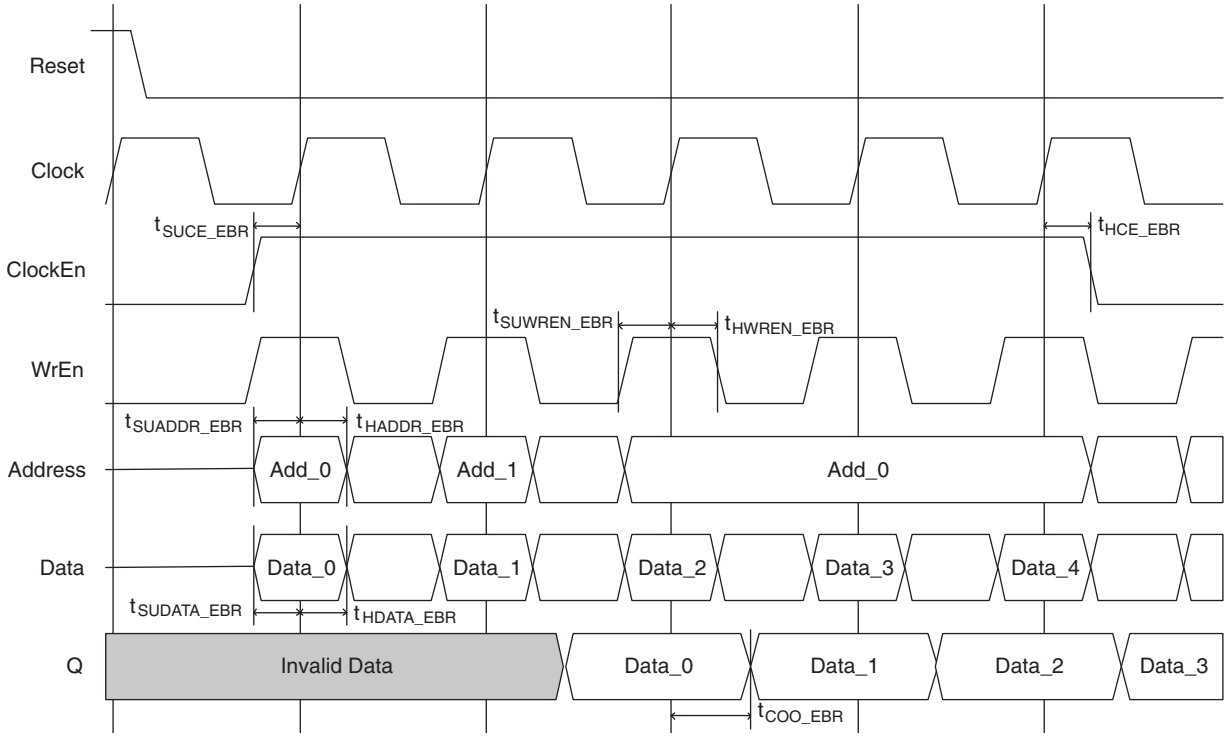


Figure 9-14. Single Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

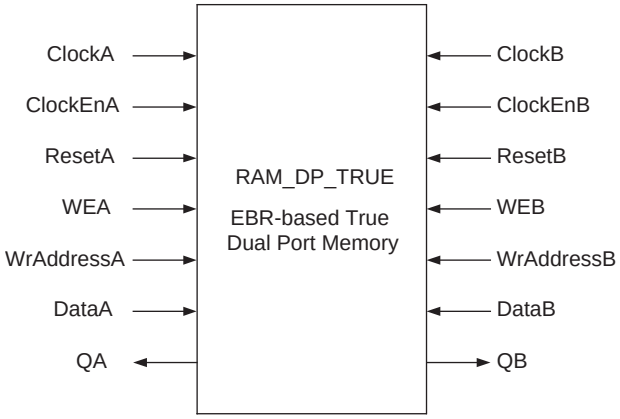


True Dual Port RAM (RAM_DP_TRUE) – EBR Based

The EBR blocks in the LatticeECP/EC and LatticeXP 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 9-15.

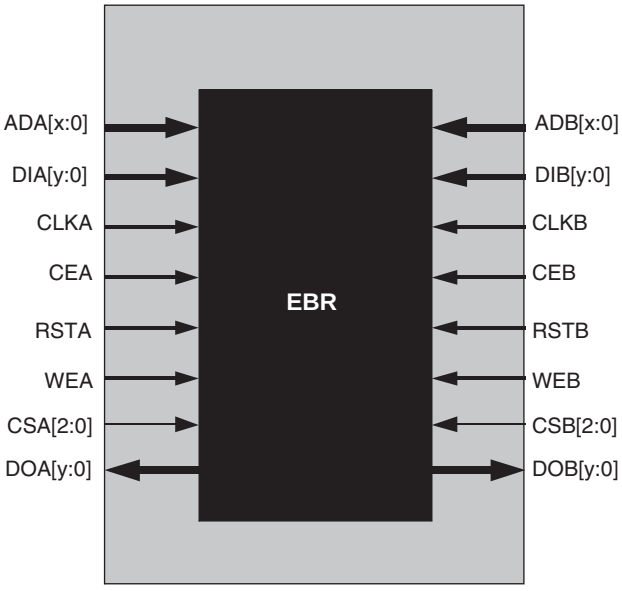
Figure 9-15. True Dual Port Memory Module Generated by IPexpress



The generated module makes use of the RAM_DP_TRUE primitive. For memory sizes smaller than one EBR block, the module will be created in one EBR block. In cases where 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).

The basic memory primitive for the LatticeECP/EC and LatticeXP devices, RAM_DP_TRUE, is shown in Figure 9-16.

Figure 9-16. True Dual Port RAM Primitive or RAM_DP_TRUE for LatticeECP/EC and LatticeXP Devices



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 the True Dual Memory are included in Table 9-4. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP_TRUE primitive.

Table 9-4. 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[x:0], ADB[x:0]	Address Bus Port A and Port B	—
DataA, DataB	DIA[y:0], DIB[y:0]	Input Data Port A and Port B	—
QA, QB	DOA[y:0], DOB[y:0]	Output Data Port A and Port B	—
WEA, WEB	WEA, WEB	Write Enable Port A and Port B	Active High
ResetA, ResetB	RSTA, RSTB	Reset for Port A and Port B	Active High
—	CSA[2:0], CSB[2:0]	Chip Selects for Each Port	—

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

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

Each EBR block consists of 9,216 bits of RAM. The values for x (for Address) and y (Data) for each EBR block for the devices are included in Table 9-5.

Table 9-5. True Dual Port Memory Sizes for 9K Memory for LatticeECP/EC and LatticeXP Devices

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]
8K x 1	DIA	DIB	DOA	DOB	ADA[12:0]	ADB[12:0]
4K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[11:0]	ADB[11:0]
2K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[10:0]	ADB[10:0]
1K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[9:0]	ADB[9:0]
512 x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	ADA[8:0]	ADB[8:0]

Table 9-6 shows the various attributes available for True Dual Port Memory (RAM_DP_TRUE). Some of these attributes are user selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 9-6. True Dual Port RAM Attributes for LatticeECP/EC and LatticeXP

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH_A	Data Word Width Port A	1, 2, 4, 9, 18	1	YES
DATA_WIDTH_B	Data Word Width Port B	1, 2, 4, 9, 18	1	YES
REGMODE_A	Register Mode (Pipelining) for Port A	NOREG, OUTREG	NOREG	YES
REGMODE_B	Register Mode (Pipelining) for Port B	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE_A	Chip Select Decode for Port A	000, 001, 010, 011, 100, 101, 110, 111	000	NO
CSDECODE_B	Chip Select Decode for Port B	000, 001, 010, 011, 100, 101, 110, 111	000	NO
WRITEMODE_A	Read / Write Mode for Port A	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
WRITEMODE_B	Read / Write Mode for Port B	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
GSR	Global Set Reset	ENABLED, DISABLED	ENABLED	YES

The True Dual Port RAM (RAM_DP_TRUE) can be configured as NORMAL, READ BEFORE WRITE 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 READ BEFORE WRITE attribute is supported for x9 and x18 data widths. 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 8-15 through 8-20 show the internal timing waveforms for the True Dual Port RAM (RAM_DP_TRUE) with these options.

Figure 9-17. True Dual Port RAM Timing Waveform – NORMAL Mode, without Output Registers

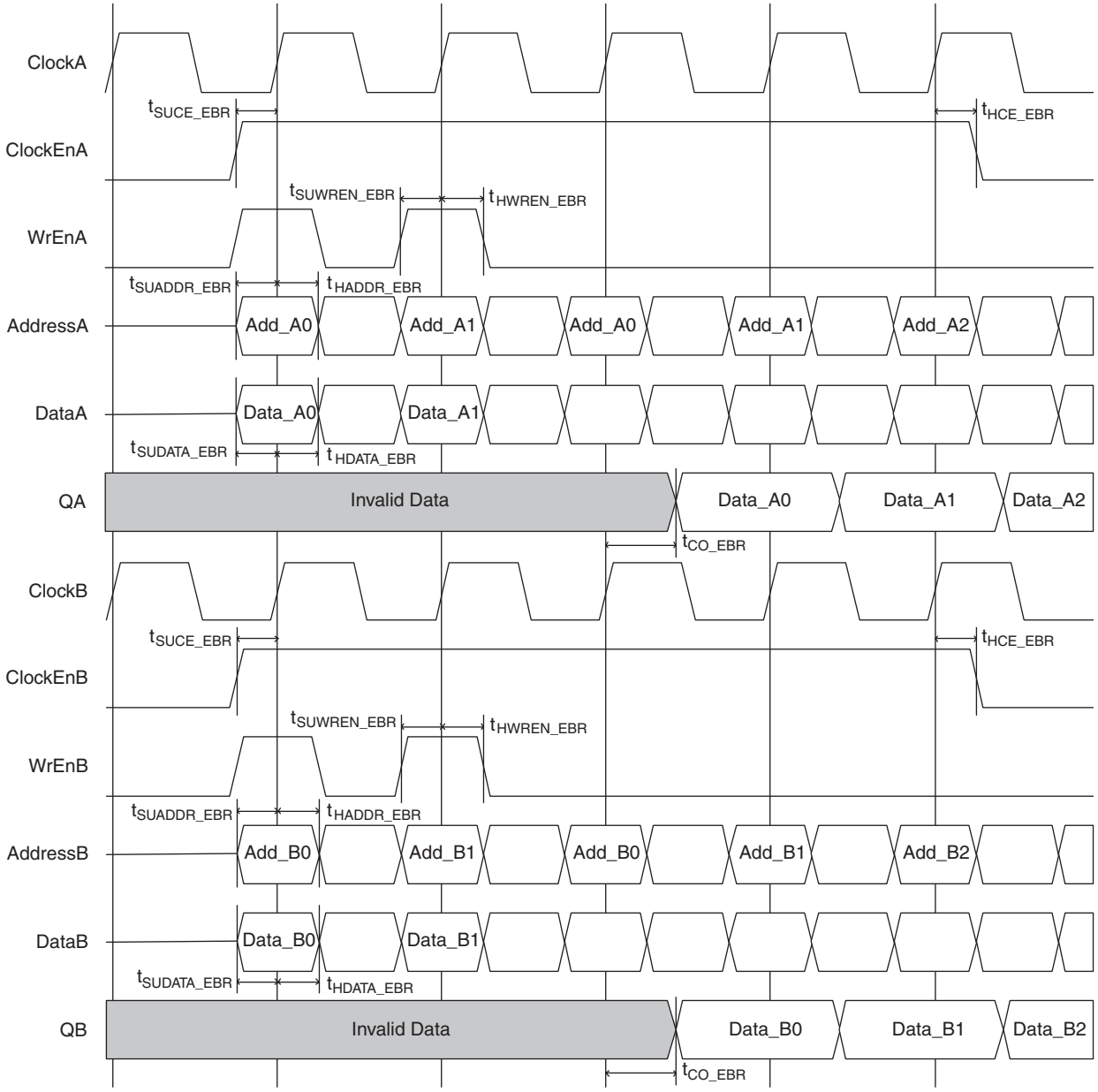


Figure 9-18. True Dual Port RAM Timing Waveform – NORMAL Mode with Output Registers

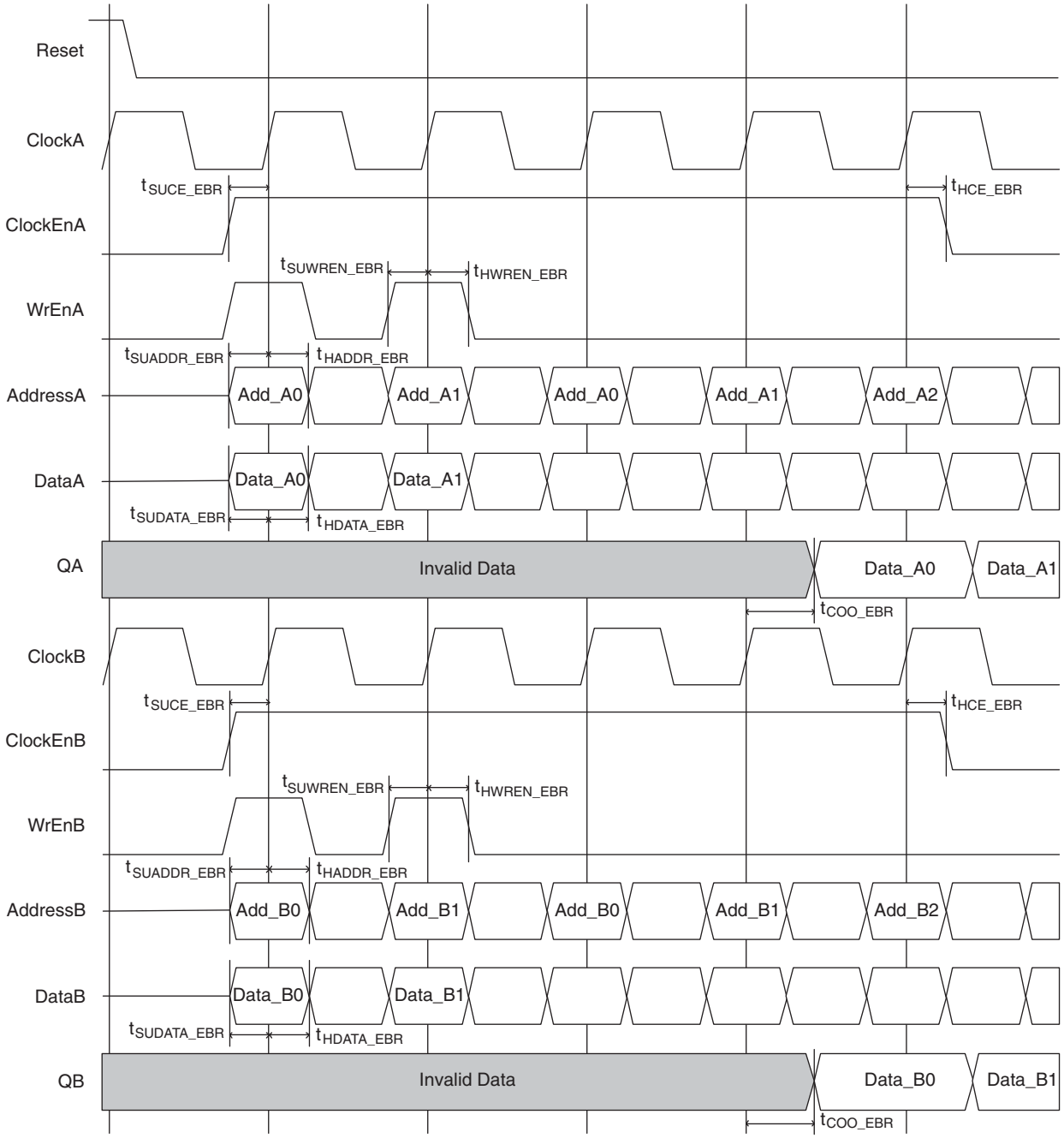


Figure 9-19. True Dual Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers

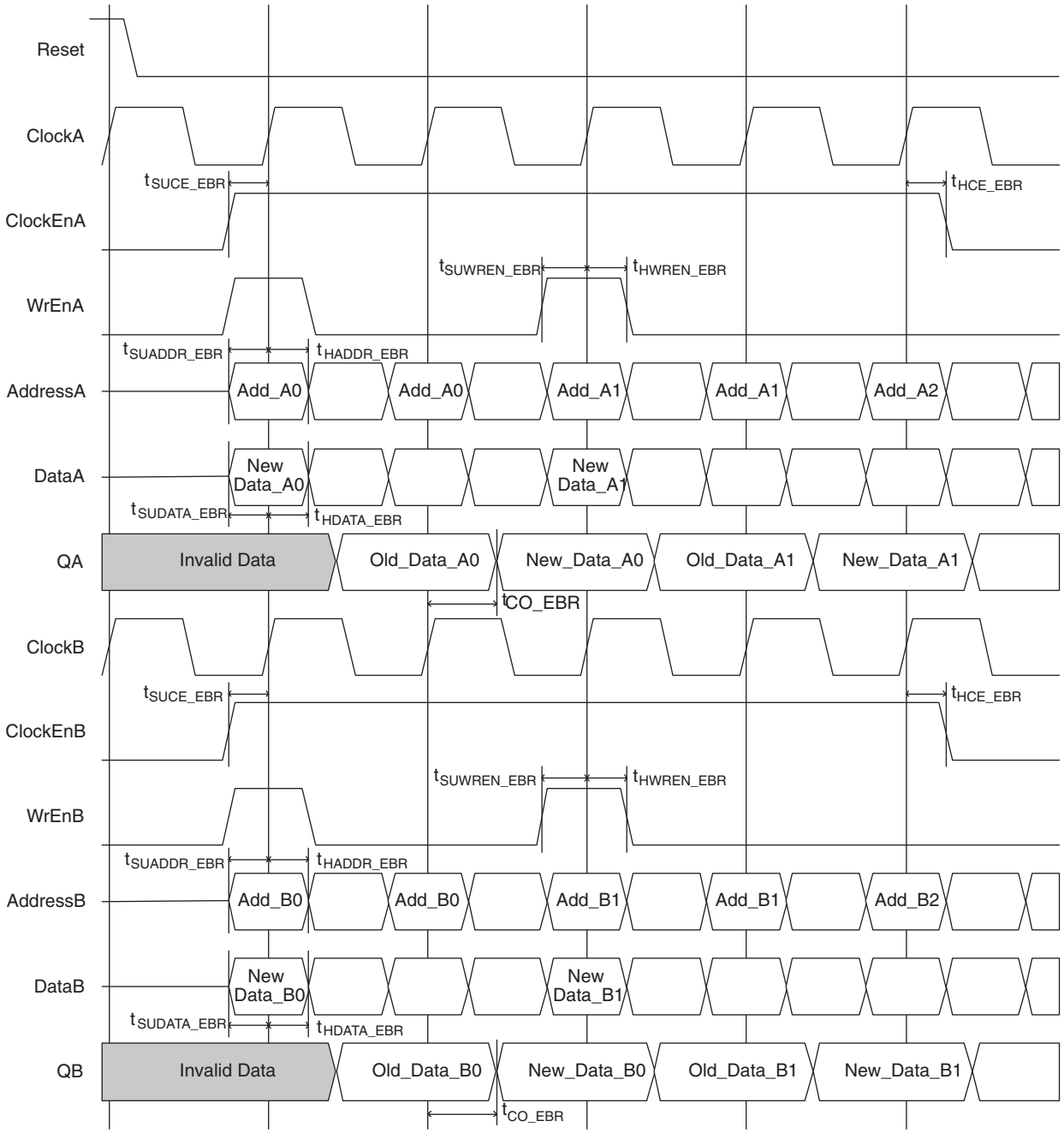


Figure 9-20. True Dual Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers

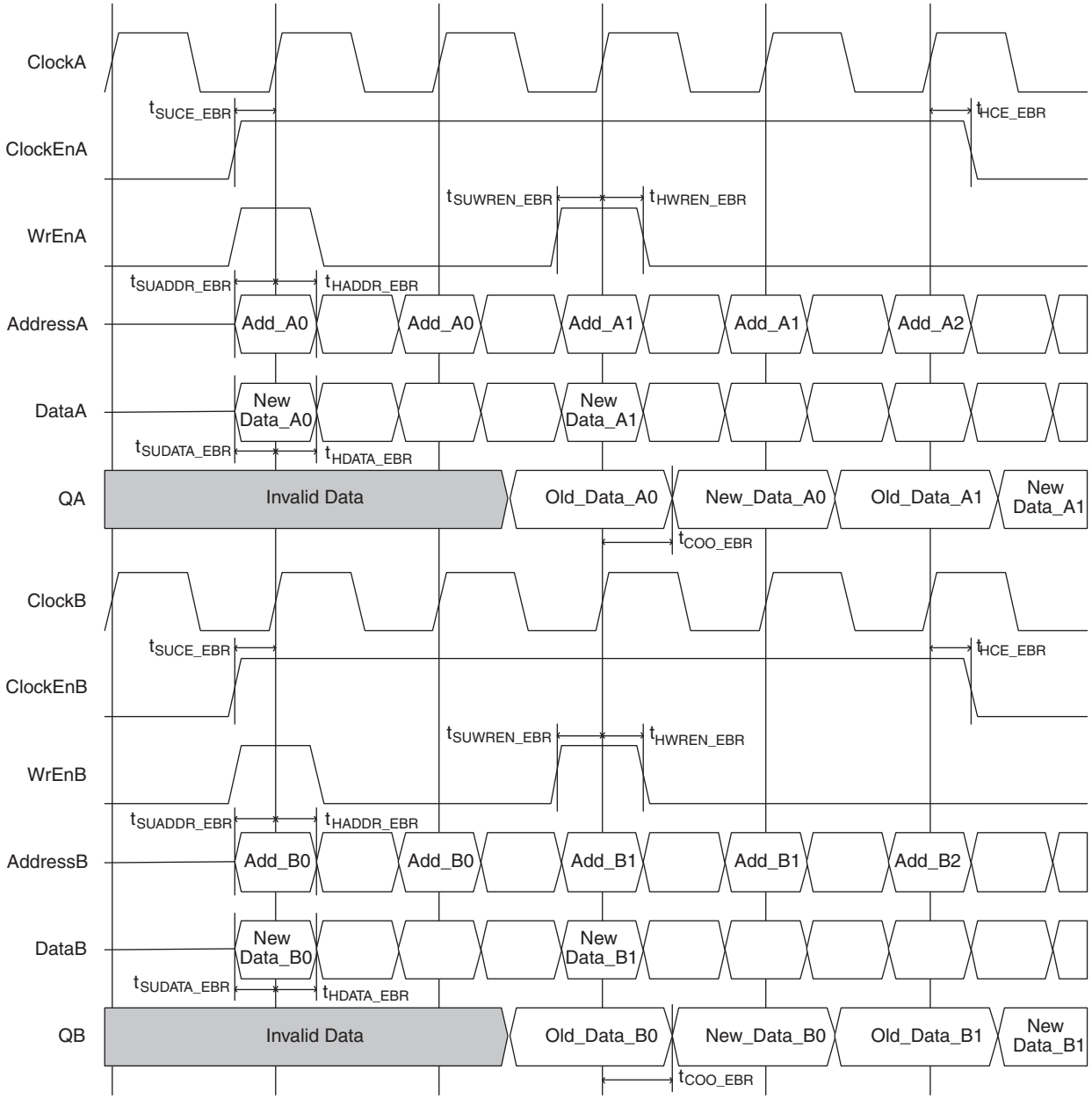


Figure 9-21. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

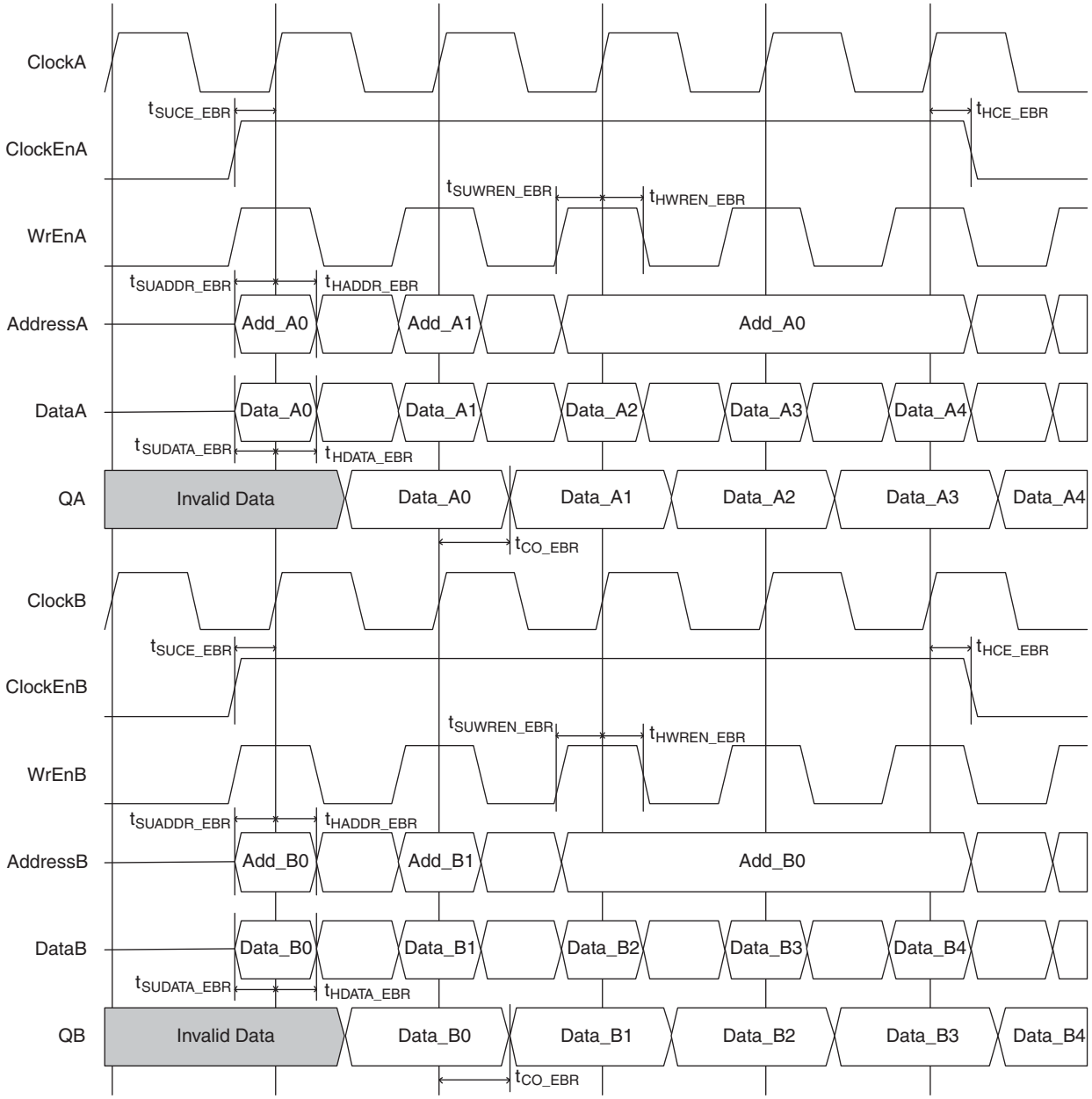
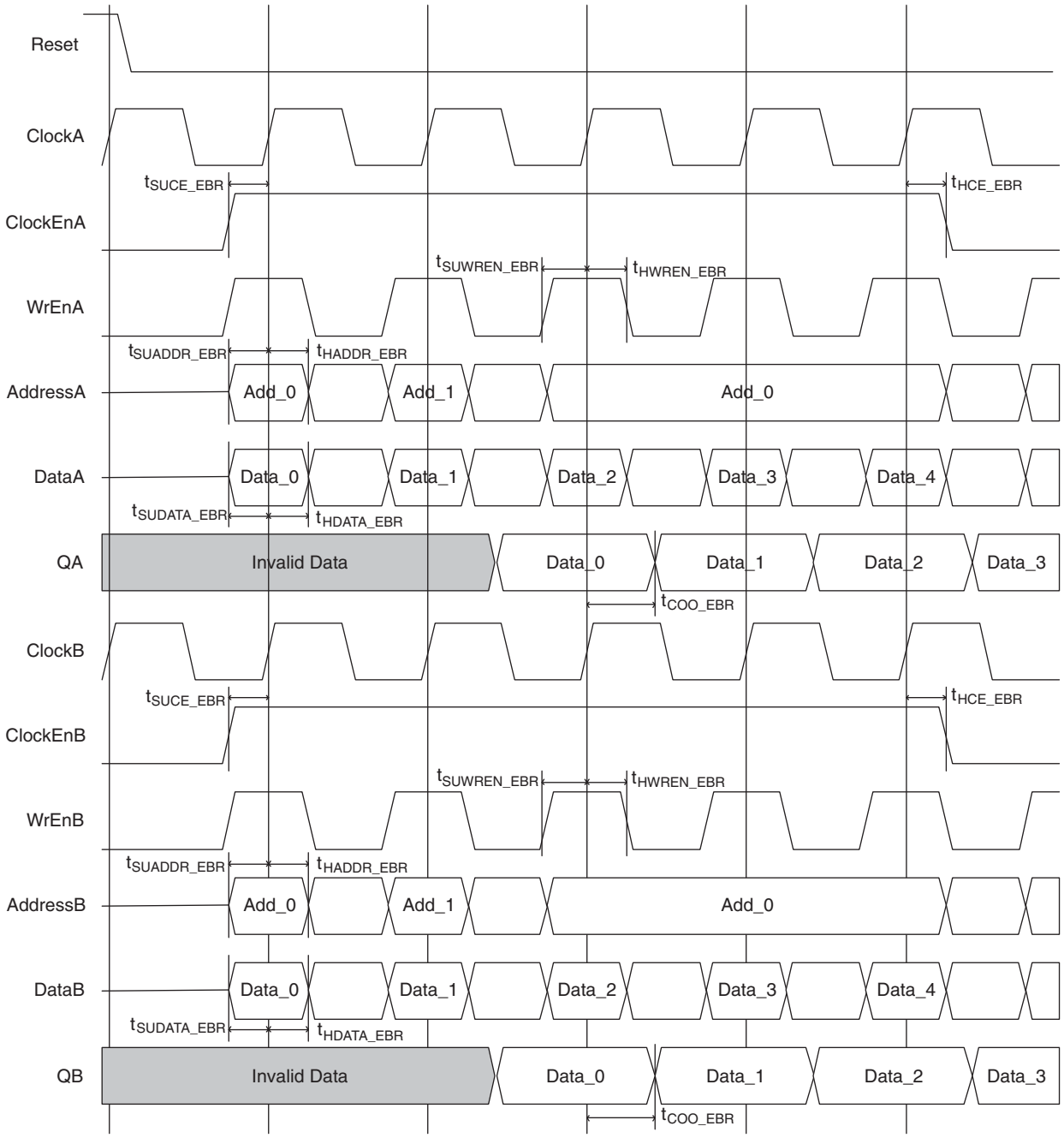


Figure 9-22. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

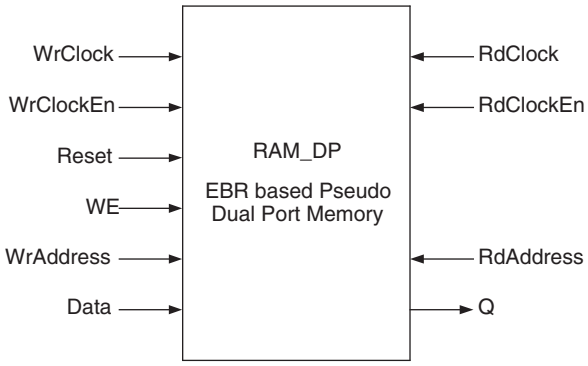


Pseudo Dual Port RAM (RAM_DP) – EBR-Based

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

IPexpress generates the memory module, as shown in Figure 9-23.

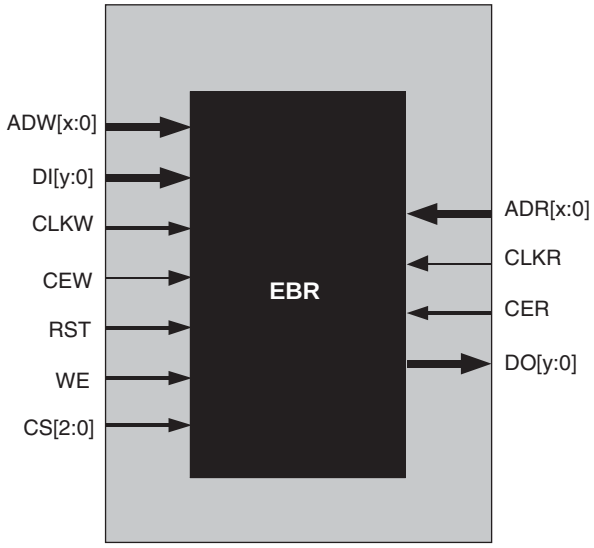
Figure 9-23. Pseudo Dual Port Memory Module Generated by IPexpress



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 block can be cascaded, in depth or width (as required to create these sizes).

The basic Pseudo Dual Port memory primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-24.

Figure 9-24. Pseudo Dual Port RAM primitive or RAM_DP for LatticeECP/EC and LatticeXP Devices



In the 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 included in Table 9-7. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP primitive.

Table 9-7. 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[x:0]	Read Address	—
WrAddress	ADW[x: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[y:0]	Read Data	—
Data	DI[y:0]	Write Data	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

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

CS, or Chip Select, a port available in the EBR primitive, is useful when memory requires multiple EBR blocks to be 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. However, if the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 9,216 bits of RAM. The values for x (for Address) and y (Data) for each EBR block for the devices are included in Table 9-8.

Table 9-8. Pseudo-Dual Port Memory Sizes for 9K Memory for LatticeECP/EC and LatticeXP Devices

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]
8K x 1	DIA	DIB	DOA	DOB	RAD[12:0]	WAD[12:0]
4K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	RAD[11:0]	WAD[11:0]
2K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	RAD[10:0]	WAD[10:0]
1K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	RAD[9:0]	WAD[9:0]
512 x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	RAD[9:0]	WAD[9:0]

Table 9-9 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 9-9. Pseudo-Dual Port RAM Attributes for LatticeECP/EC and LatticeXP Devices

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH_W	Write Data Word Width	1, 2, 4, 9, 18, 36	1	YES
DATA_WIDTH_R	Read Data Word Width	1, 2, 4, 9, 18, 36	1	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE_W	Chip Select Decode for Write	000, 001, 010, 011, 100, 101, 110, 111	000	NO
CSDECODE_R	Chip Select Decode for Read	000, 001, 010, 011, 100, 101, 110, 111	000	NO
GSR	Global Set Reset	ENABLED, DISABLED	ENABLED	YES

Users have the option of enabling the output registers for Pseudo-Dual Port RAM (RAM_DP). Figures 8-23 and 8-24 show the internal timing waveforms for the Pseudo-Dual Port RAM (RAM_DP) with these options.

Figure 9-25. PSEUDO DUAL PORT RAM Timing Diagram – without Output Registers

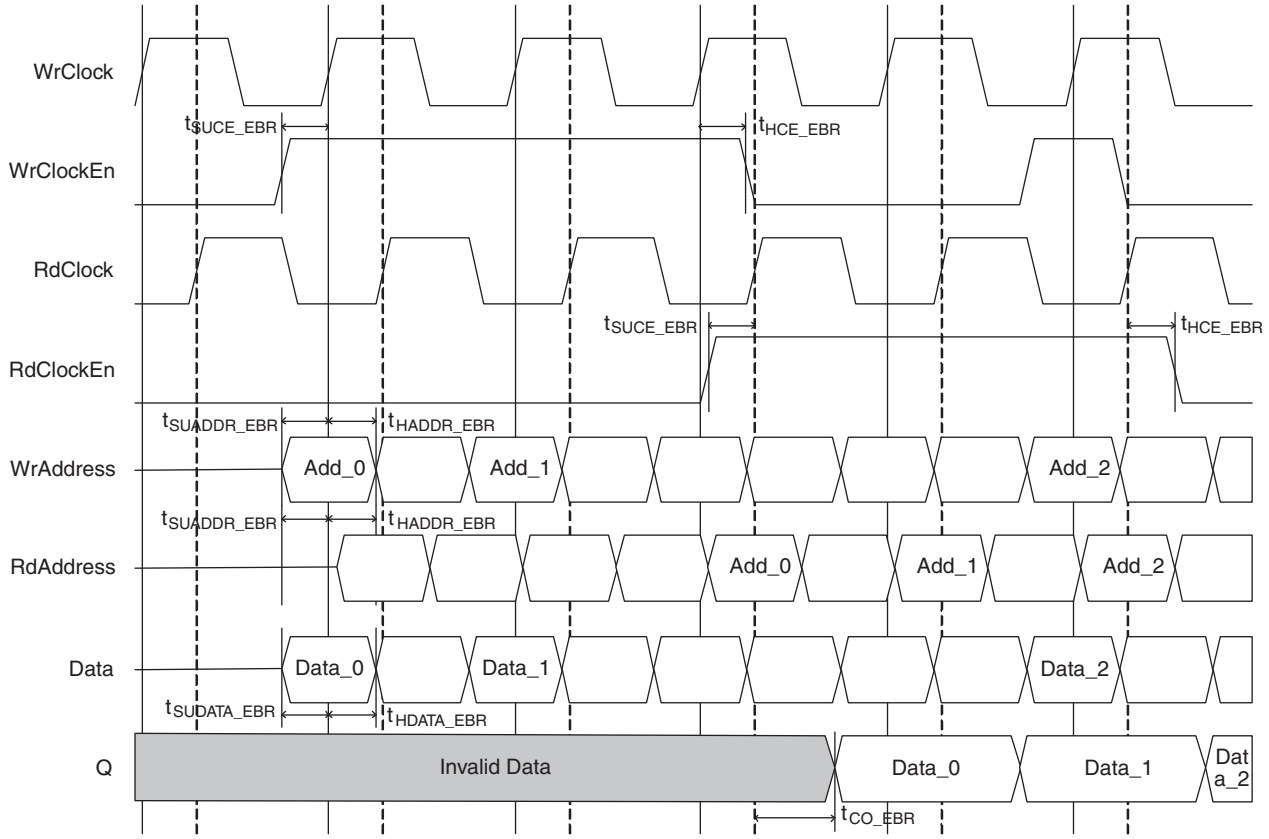
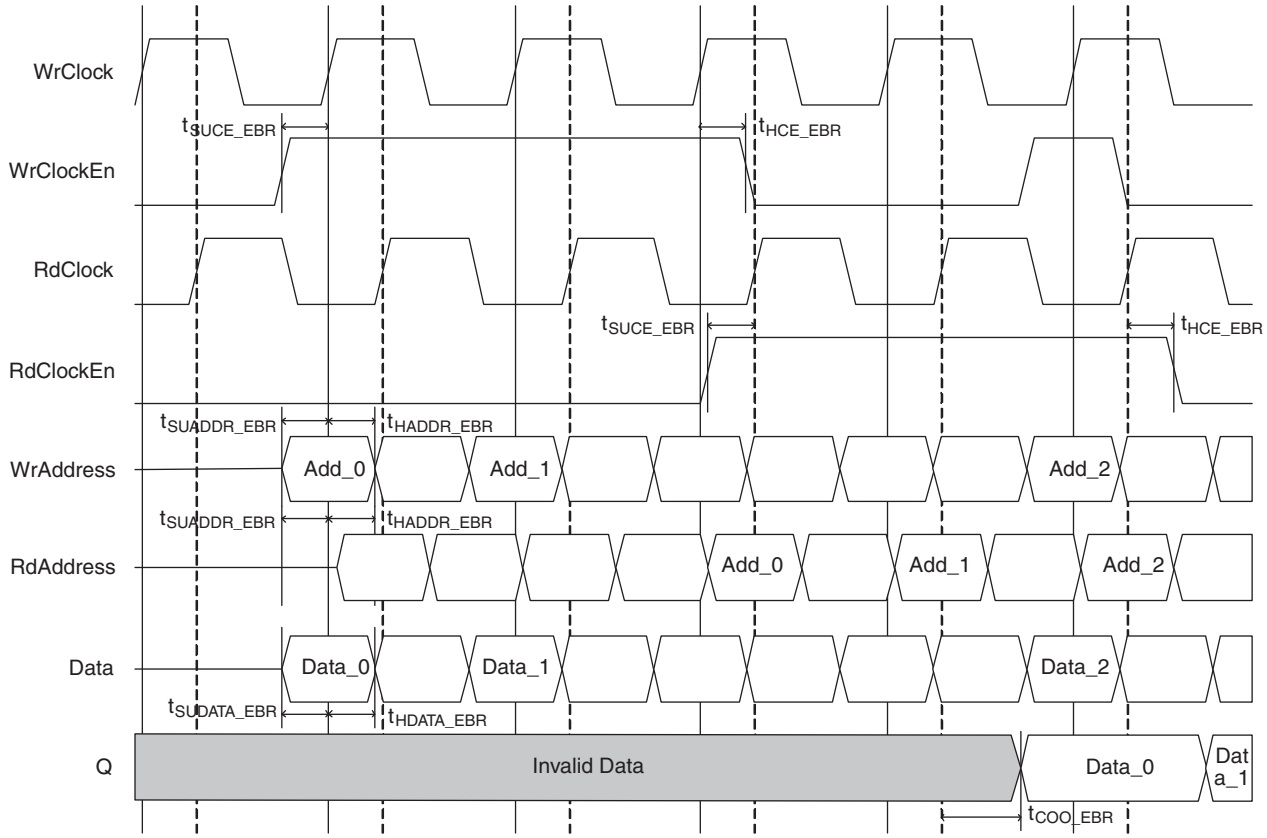


Figure 9-26. PSEUDO DUAL PORT RAM Timing Diagram – with Output Registers

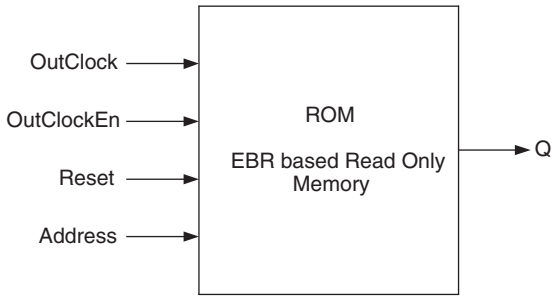


Read Only Memory (ROM) – EBR Based

The EBR blocks in the LatticeECP/EC and LatticeXP devices can be configured as Read Only Memory or ROM. IPexpress allows users to generate the Verilog-HDL or VHDL along with an 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 9-27.

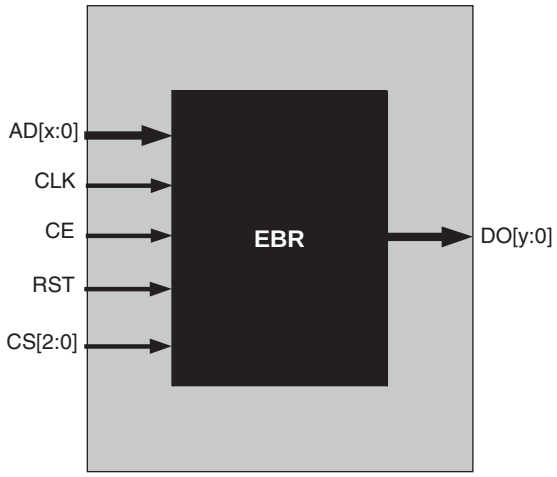
Figure 9-27. ROM - Read Only Memory Module Generated by IPexpress



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).

The basic ROM primitive for the LatticeECP/EC and LatticeXP devices is as shown in Figure 9-28.

Figure 9-28. ROM Primitive for LatticeECP/EC and LatticeXP Devices



In the 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 included in Table 9-10. The table lists the corresponding ports for the module generated by IPexpress and for the ROM primitive.

Table 9-10. 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 (or RST) only resets the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, a port available in the EBR primitive, is useful when memory requires multiple EBR blocks to be 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. However, if the memory size specified by the user requires more than eight EBR blocks, the 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 is required to provide an initialization file to pre-initialize the contents of the ROM. These file 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 technical note.

Users have the option of enabling the output registers for Read Only Memory (ROM). Figures 8-27 and 8-28 show the internal timing waveforms for the Read Only Memory (ROM) with these options.

Figure 9-29. ROM Timing Waveform – without Output Registers

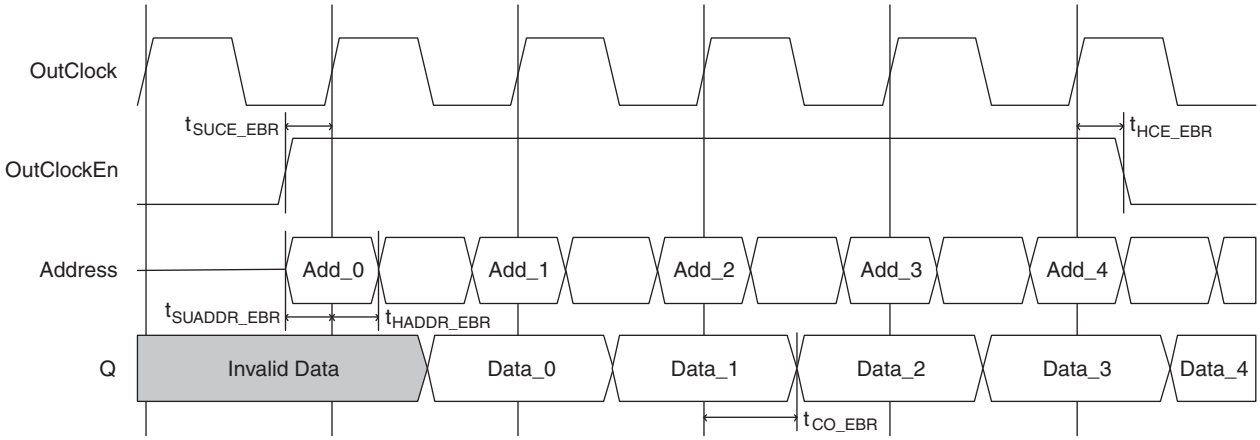
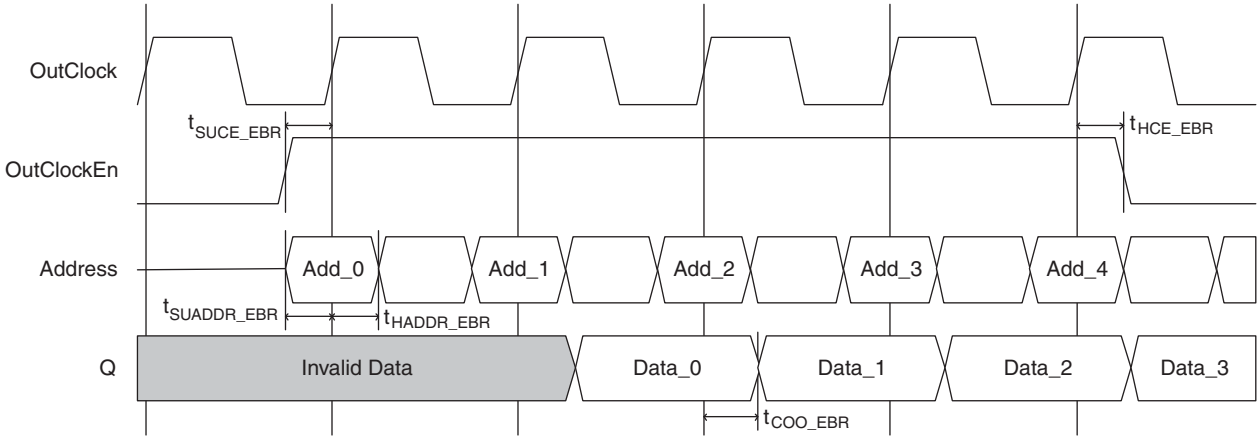


Figure 9-30. ROM Timing Waveform – with Output Registers



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

The EBR blocks in the LatticeECP/EC and LatticeXP devices can be configured as First In First Out Memories – FIFO and FIFO_DC. FIFO has a common clock for both read and write ports and FIFO_DC (or Dual Clock FIFO) has separate clocks for these ports. IPexpress allows users to generate the Verilog-HDL or VHDL along with an EDIF netlist for the memory size as per design requirement.

IPexpress generates the FIFO and FIFO_DC memory module as shown in Figures 9-31 and 9-32.

Figure 9-31. FIFO Module Generated by IPexpress

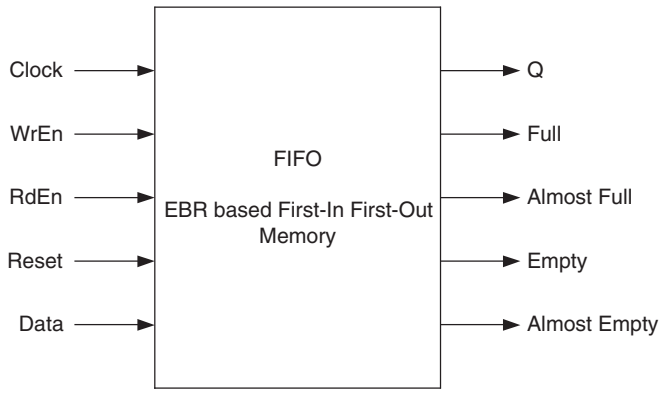
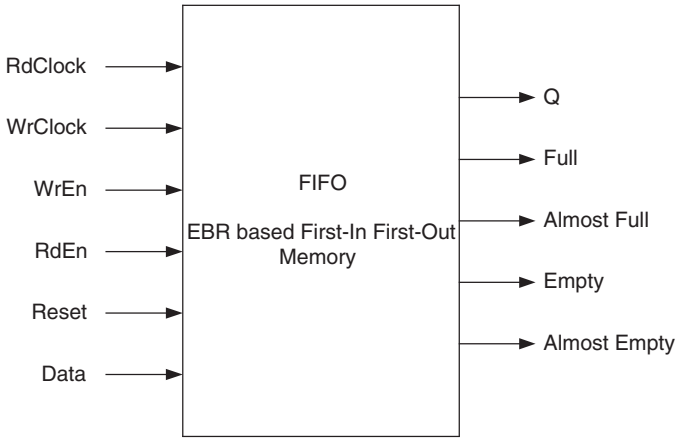


Figure 9-32. FIFO_DC Module Generated by IPexpress



LatticeECP/EC and LatticeXP devices do not have a built in FIFO. These devices have an emulated FIFO and FIFO_DC. These are emulated by creating a wrapper around the existing RAMs (like RAM_DP). This wrapper also includes address pointer generation and FIFO flag generation logic which will be implemented external to the EBR block. Therefore, in addition to the regular EBR usage, there is extra logic for the address pointer generation and FIFO flag generation.

A clock is always required as only synchronous write is supported. The various ports and their definitions for the FIFO and FIFO_DC are included in Table 11.

Table 9-11. EBR-based FIFO and FIFO_DC Memory Port Definitions

Port Name in Generated Module	Description	
CLK	Clock (FIFO)	Rising Clock Edge
CLKR	Read Port Clock (FIFO_DC)	Rising Clock Edge
CLKW	Write Port Clock (FIFO_DC)	Rising Clock Edge
WE	Write Enable	Active High
RE	Read Enable	Active High
RST	Reset	Active High
DI	Data Input	—
DO	Data Output	—
FF	Full Flag	Active High
AF	Almost Full Flag	Active High
EF	Empty Flag	Active High
AE	Almost Empty	Active High

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

The various supported sizes for the FIFO and FIFO_DC in LatticeECP/EC and LatticeXP devices are shown in Table 9-12.

Table 9-12. FIFO and FIFO_DC Data Widths Sizes for LatticeECP/EC and LatticeXP Devices

FIFO Size	Input Data	Output Data
8K x 1	DI	DO
4K x 2	DI[1:0]	DO[1:0]
2K x 4	DI[3:0]	DO[3:0]
1K x 9	DI[8:0]	DO[8:0]
512 x 18	DI[17:0]	DO[17:0]
256 x 36	DI[35:0]	DO[35:0]

FIFO Flags

The FIFO and FIFO_DC have four flags available: Empty, Almost Empty, Almost Full and Full. The Almost Empty and Almost Full flags have a programmable range.

The program ranges for the four FIFO flags are specified in Table 9-13.

Table 9-13. FIFO Flag Settings

FIFO Attribute Name	Description	Programming Range	Program Bits
FF	Full flag setting	2N - 1	14
AFF	Almost full setting	1 to (FF-1)	14
AEF	Almost empty setting	1 to (FF-1)	14
EF	Empty setting	0	5

The only restriction on the flag setting is that the values must be in a specific order (Empty=0, Almost Empty next, followed by Almost Full and Full, respectively). The value of Empty is not equal to the value of Almost Empty (or Full is equal to Almost Full). In this case, a warning is generated and the value of Empty (or Full) is used in place of Almost Empty (or Almost Full). When coming out of reset, the Active High Flags empty and Almost Empty are set to high, since they are true.

The user should specify the absolute value of the address at which the Almost Empty and Almost Full Flags will go true. For example, if the Almost Full Flag is required to go true at the address location 500 for a FIFO of depth 512, the user should specify the value 500 in the IPexpress.

The Empty and Almost Empty Flags are always registered with the read clock and the Full and Almost Full Flags are always registered to the write clock.

FIFO Operation

FIFOs are not supported in the hardware. The hardware has Embedded block RAMs (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 are emulated FIFOs that are built around these RAMs.

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

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

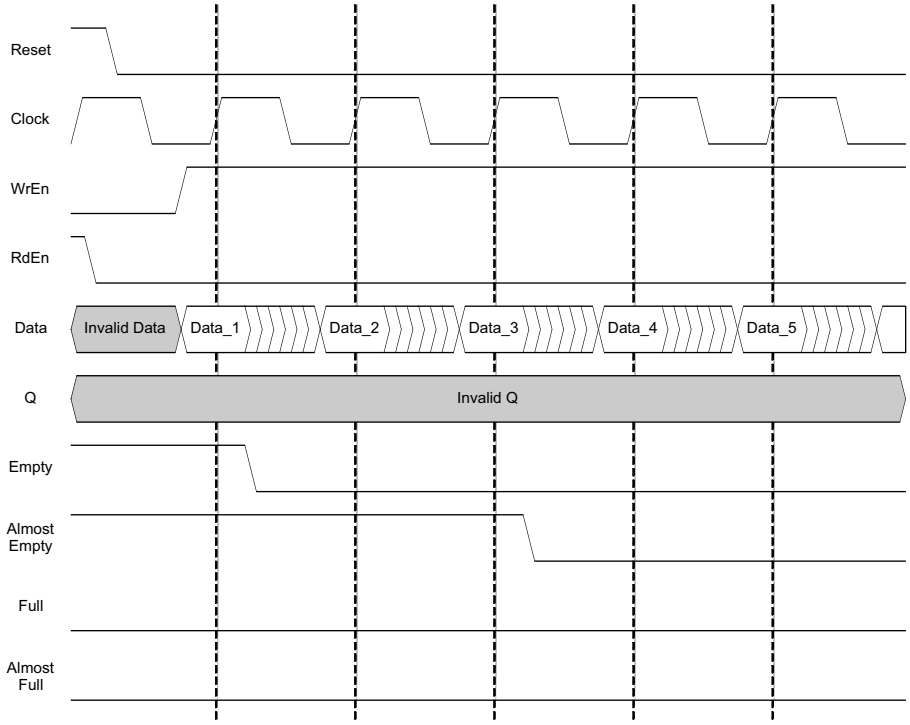
First In First Out (FIFO) Memory: The FIFO or the single clock FIFO is an emulated FIFO. The address logic and the flag logic is implemented in the FPGA fabric around the RAM.

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 9-33 shows the operation of the FIFO when it is empty and the data starts to get written into it.

Figure 9-33. FIFO Without Output Registers, Start of Data Write Cycle

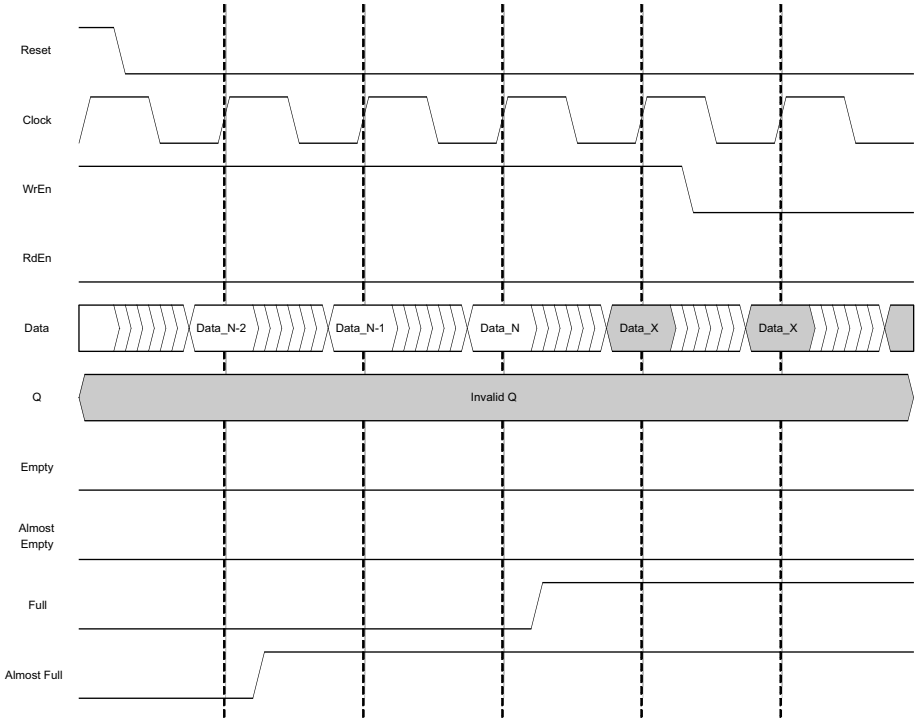


The WrEn signal has to be high to start writing into the FIFO. The Empty and Almost Empty flags are high to begin and Full and Almost full are low.

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

Now let is 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 9-34 shows the behavior of these flags. In this figure we assume that FIFO depth is ‘N’.

Figure 9-34. FIFO Without Output Registers, End of Data Write Cycle

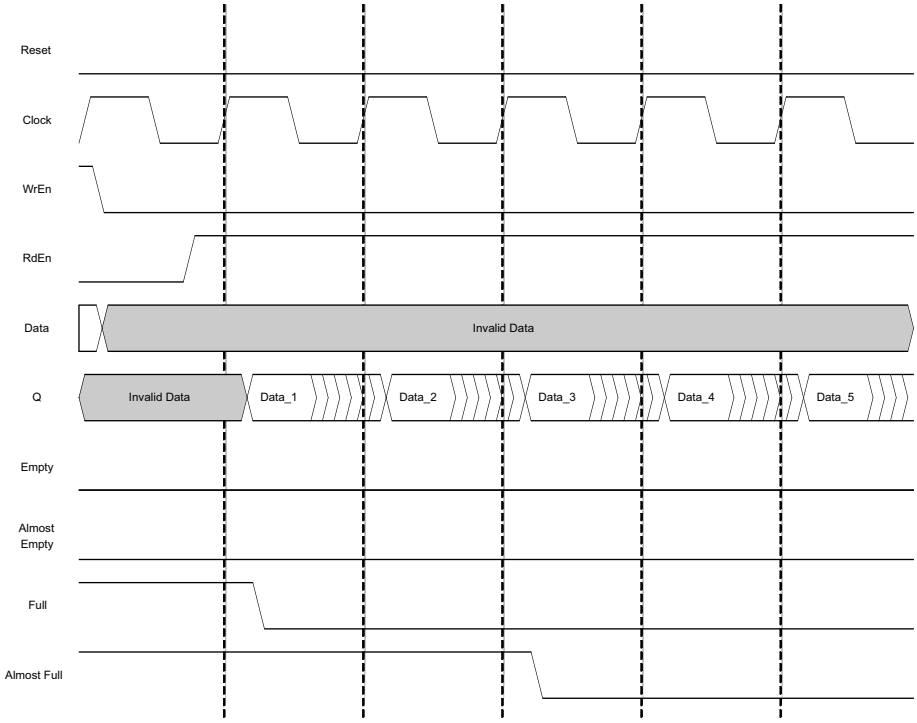


In this case, as seen above, the Almost Full flag is IN location 2 before the FIFO is filled. The Almost Full flag is asserted when N-2 location is written, and Full flag is asserted when the last word is written into the FIFO.

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

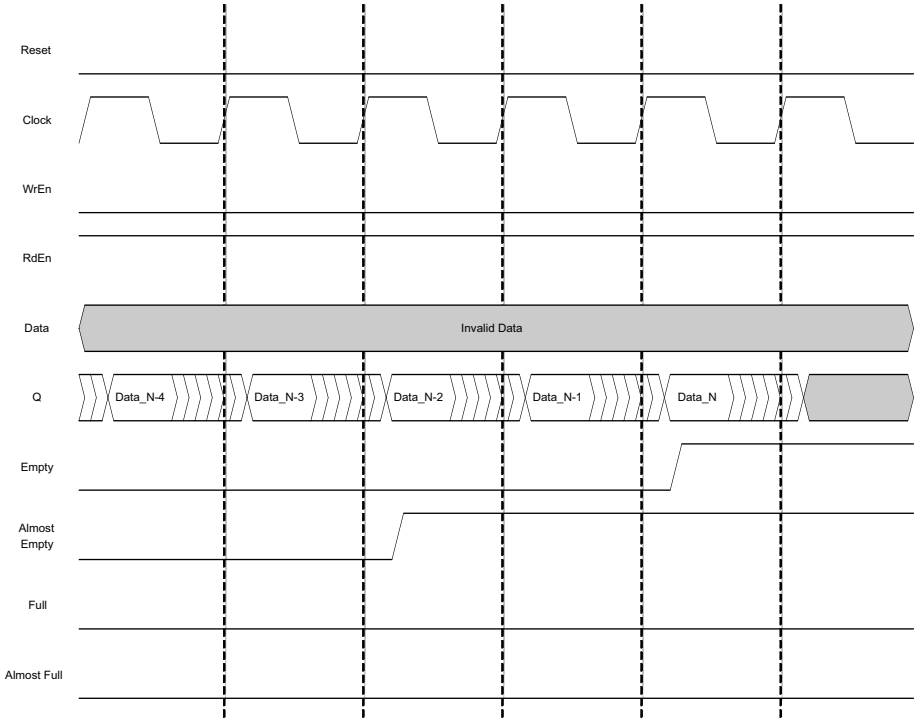
Now let us look at the waveforms when the contents of the FIFO are read out. Figure 9-35 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags gets de-asserted as shown.

Figure 9-35. 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. Below is the

Figure 9-36. FIFO Without Output Registers, End of Data Read Cycle



Figures 9-33 to 9-36 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 an extra option of output registers being enabled by RdEn signal.

Figures 9-37 to 9-40 show the similar waveforms for the FIFO with output register and without 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 gets delayed by one clock cycle.

Figure 9-37. FIFO with Output Registers, Start of Data Write Cycle

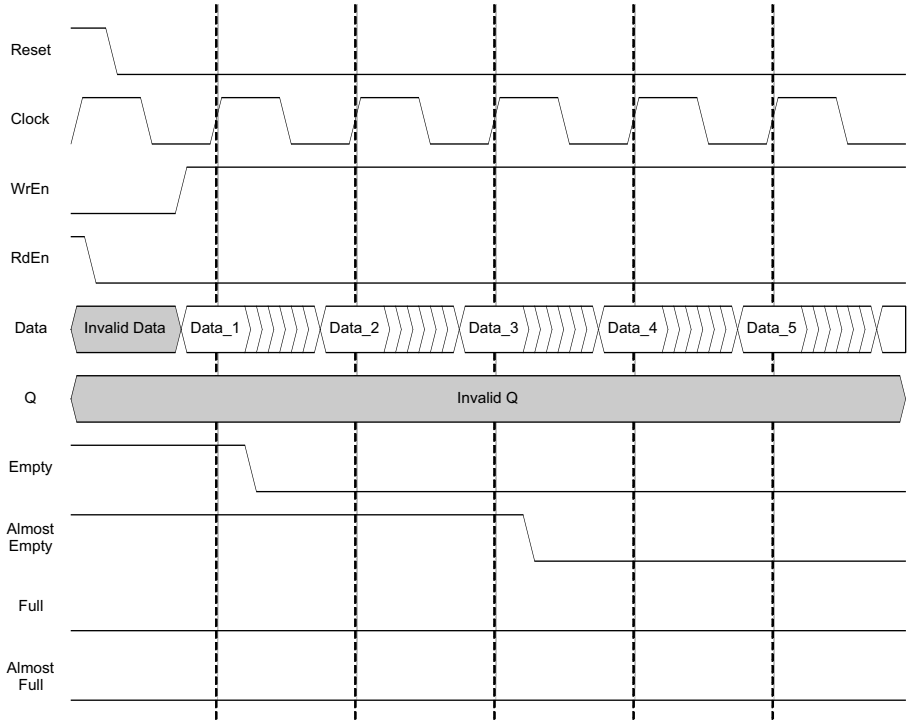


Figure 9-38. FIFO with Output Registers, End of Data Write Cycle

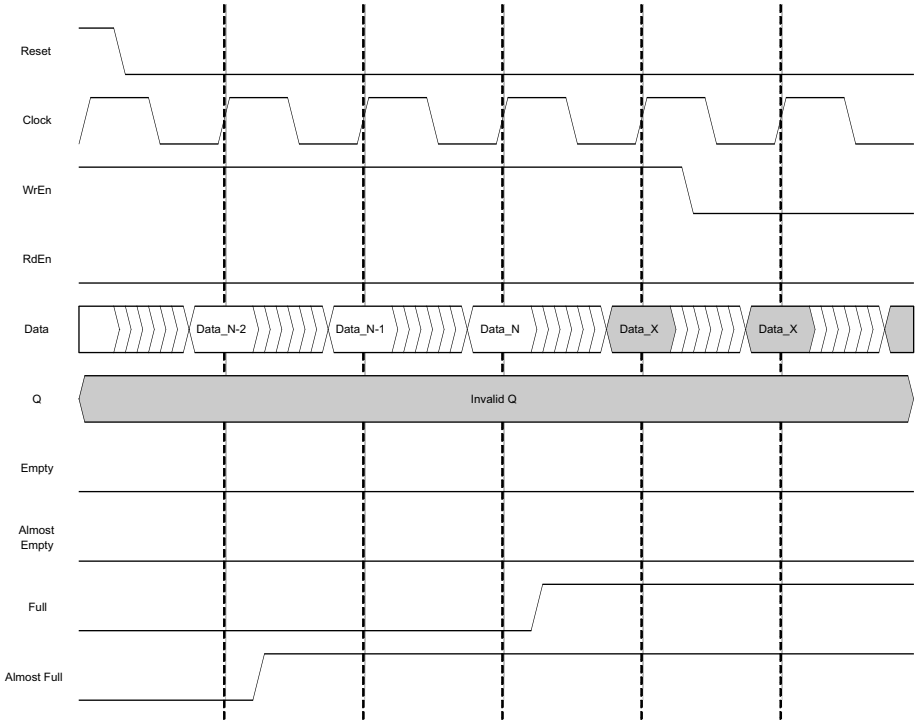


Figure 9-39. FIFO with Output Registers, Start of Data Read Cycle

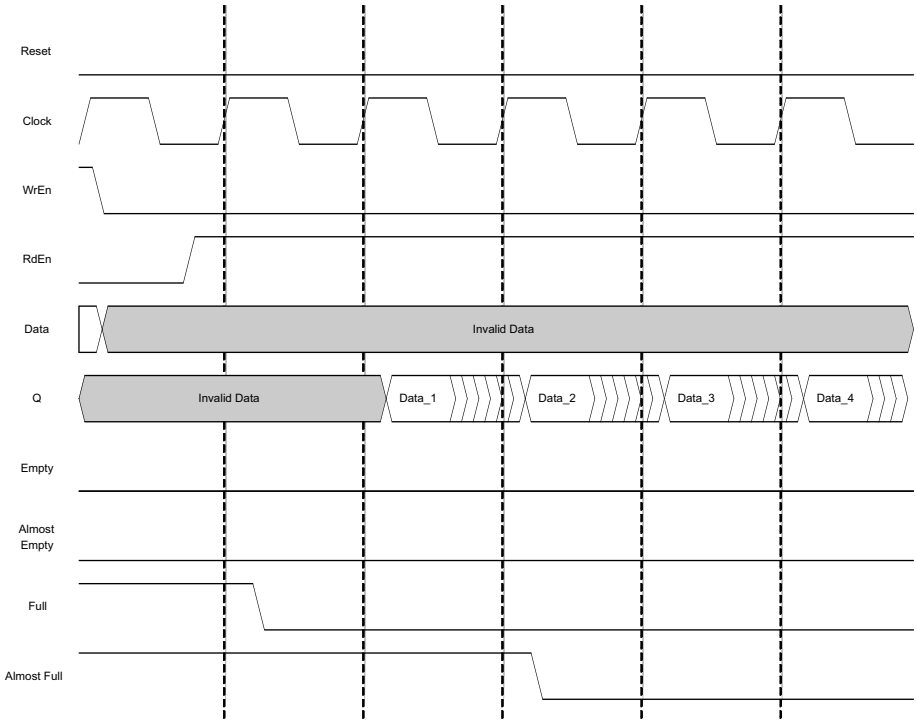
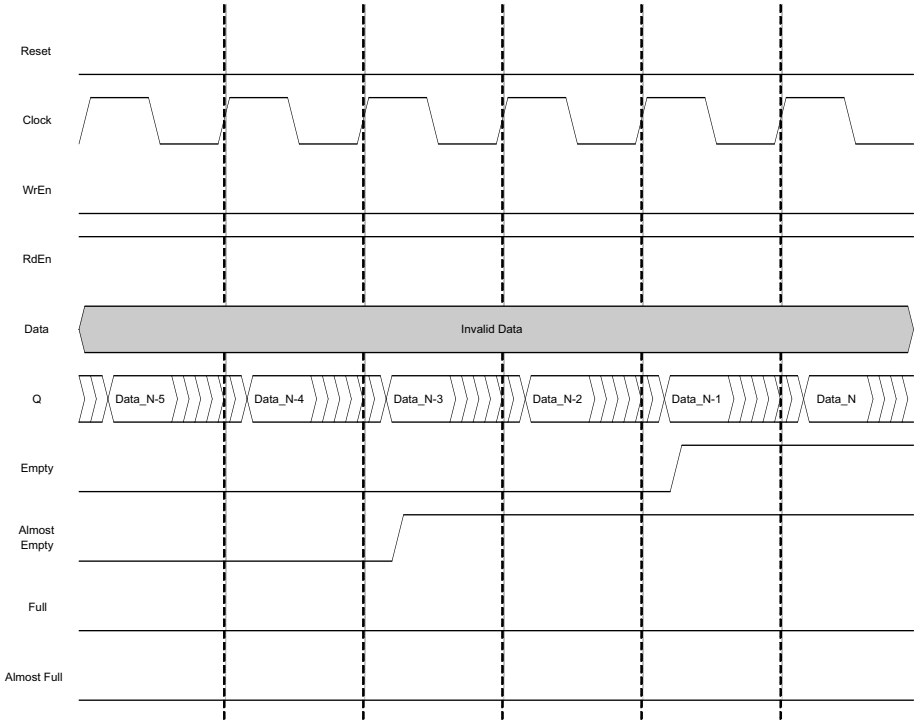
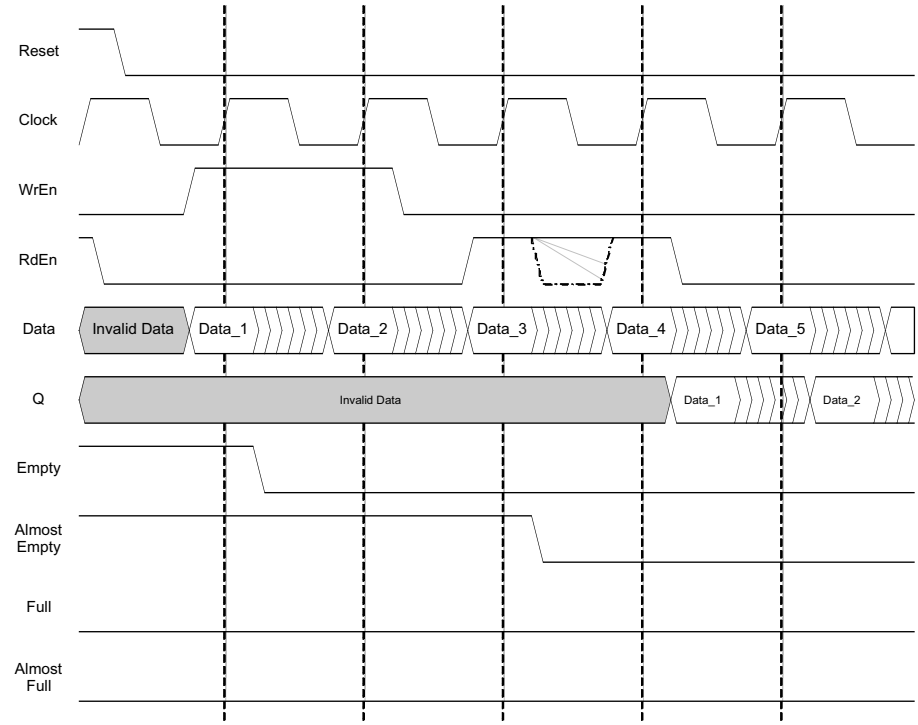


Figure 9-40. FIFO with Output Registers, End of Data Read Cycle



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), and the RdEn should be high also during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn goes true.

Figure 9-41. 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: 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 are required to change clock domains from read clock to write clock and vice versa. This adds latency to the flags either during assertion or during de-assertion. The latency can be avoided only in one of the cases (either assertion or de-assertion).

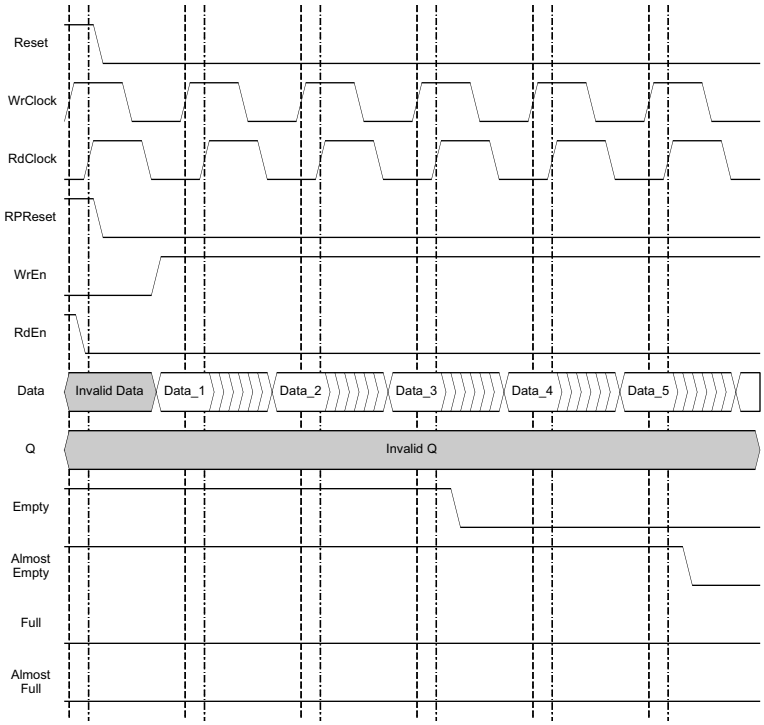
In the current emulated FIFO, there is no latency during assertion of these flags. Thus, when these flag go true, there is no latency. However this causes the latency during the de-assertion.

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 Empty and Almost Empty flags.

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

With this in mind, let us look at the FIFO_DC without output register waveforms. Figure 9-42 shows the operation of the FIFO_DC when it is empty and the data starts to get written into it.

Figure 9-42. FIFO_DC Without Output Registers, Start of Data Write Cycle

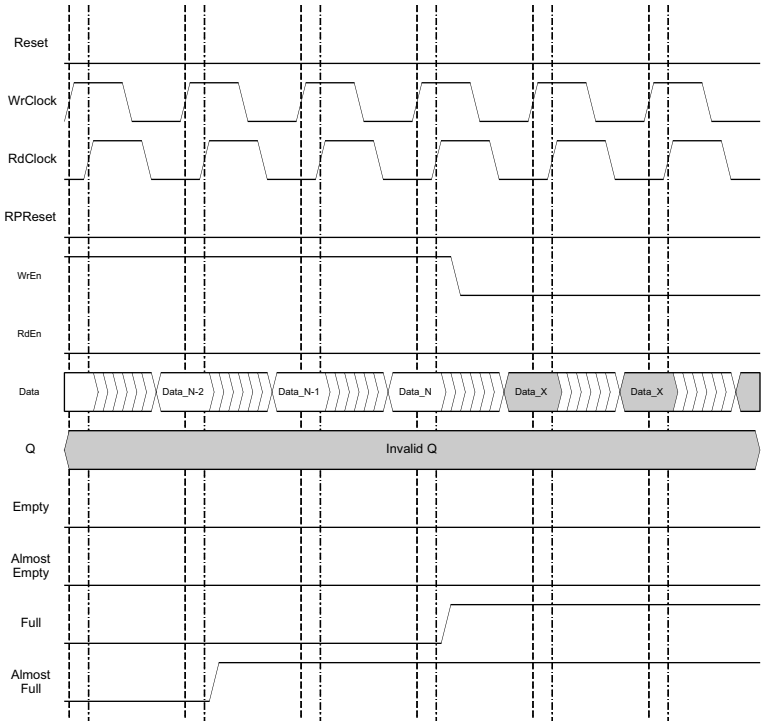


The WrEn signal has to be high to start writing into the FIFO_DC. The Empty and Almost Empty flags are high to begin and Full and Almost full are low.

When the first data gets 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 are assuming 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 gets filled.

Now let is 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 9-43 shows the behavior of these flags. In this figure we assume that FIFO_DC depth is ‘N’.

Figure 9-43. FIFO_DC Without Output Registers, End of Data Write Cycle



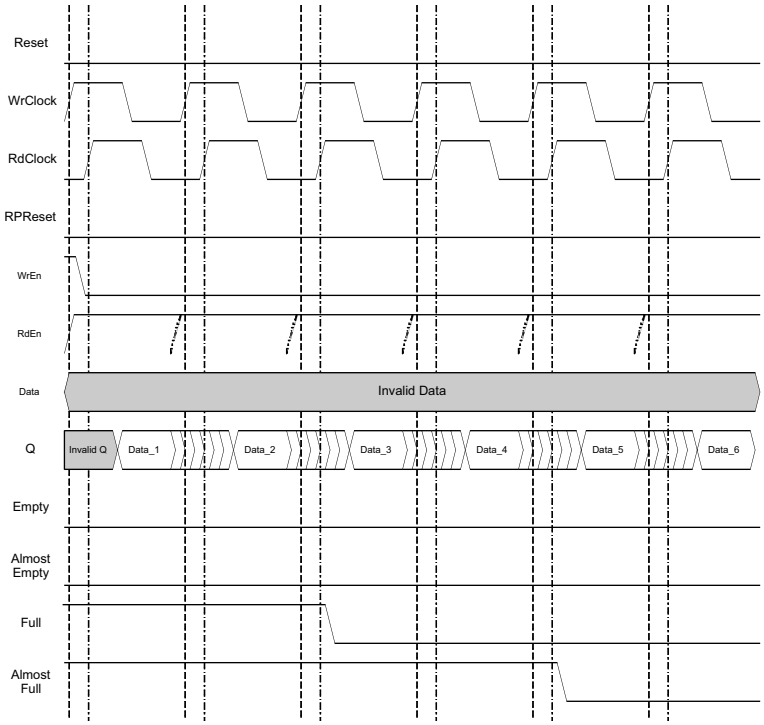
In this case, the Almost Full flag is in location 2 before the FIFO_DC is filled. The Almost Full flag is asserted when N-2 location is written, and Full flag is asserted when the last word is written into the FIFO_DC.

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

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

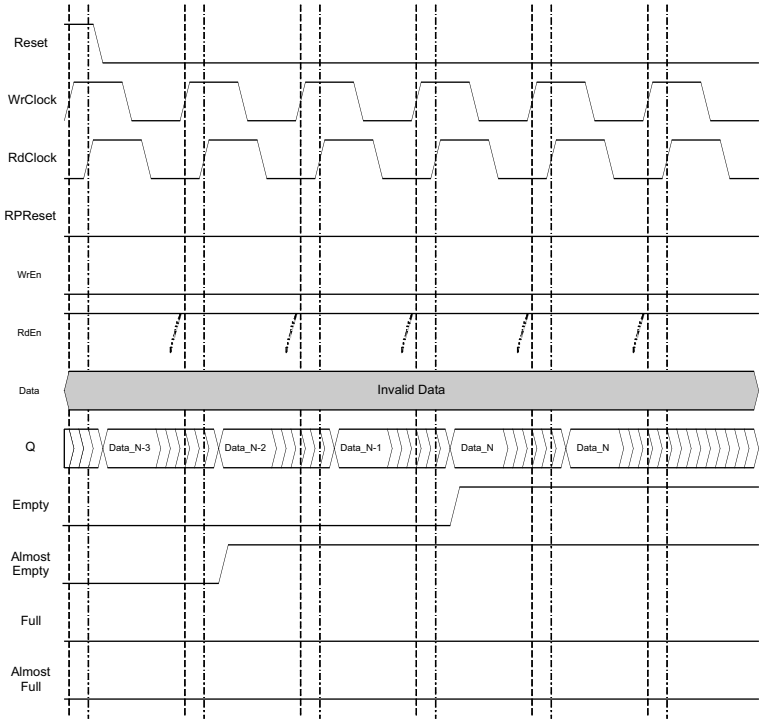
Now let us look at the waveforms when the contents of the FIFO_DC are read out. Figure 9-44 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags get de-asserted as shown. In this case, note that the de-assertion is delayed by two clock cycles.

Figure 9-44. 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. Below is the

Figure 9-45. FIFO_DC Without Output Registers, End of Data Read Cycle



Figures 9-42 to 9-45 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 output registers to be enabled by the RdEn signal.

Figures 9-46 to 9-49 show similar waveforms for the FIFO_DC with output register and without output register enable with RdEn. It should be noted 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 9-46. FIFO_DC With Output Registers, Start of Data Write Cycle

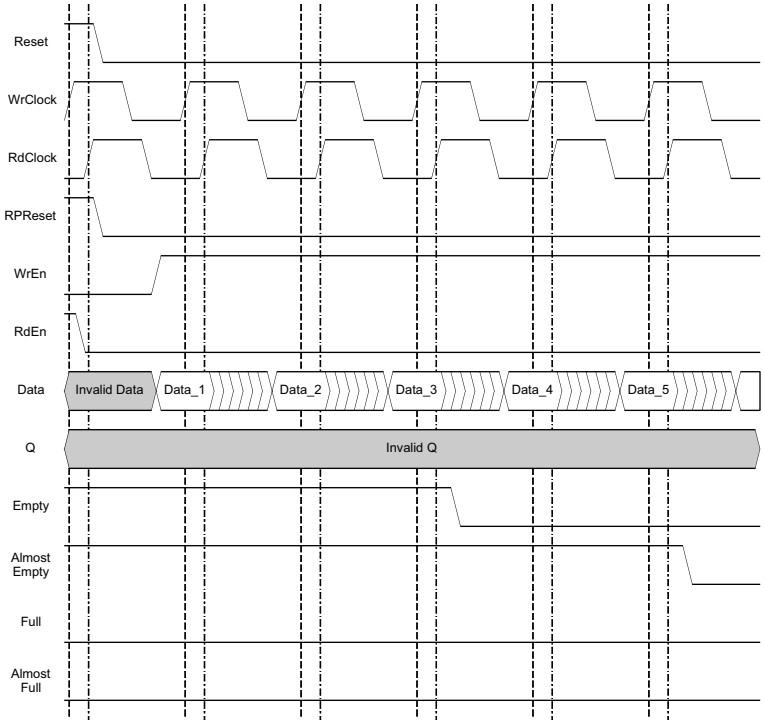


Figure 9-47. FIFO_DC With Output Registers, End of Data Write Cycle

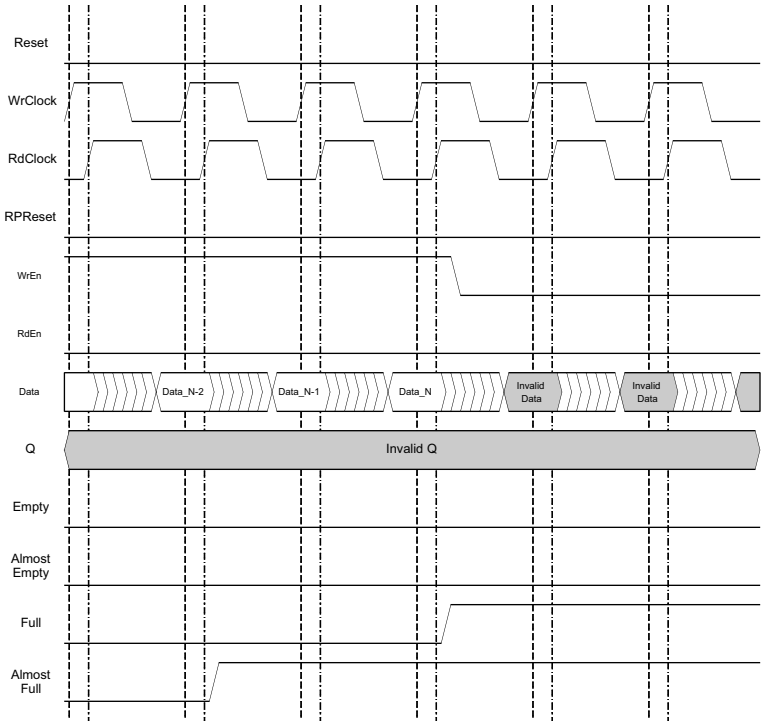


Figure 9-48. FIFO_DC With Output Registers, Start of Data Read Cycle

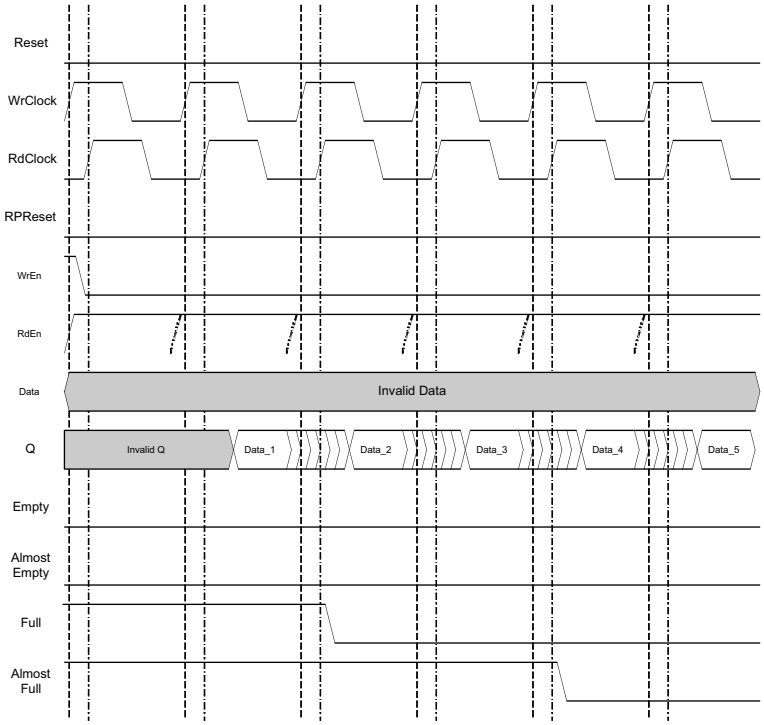
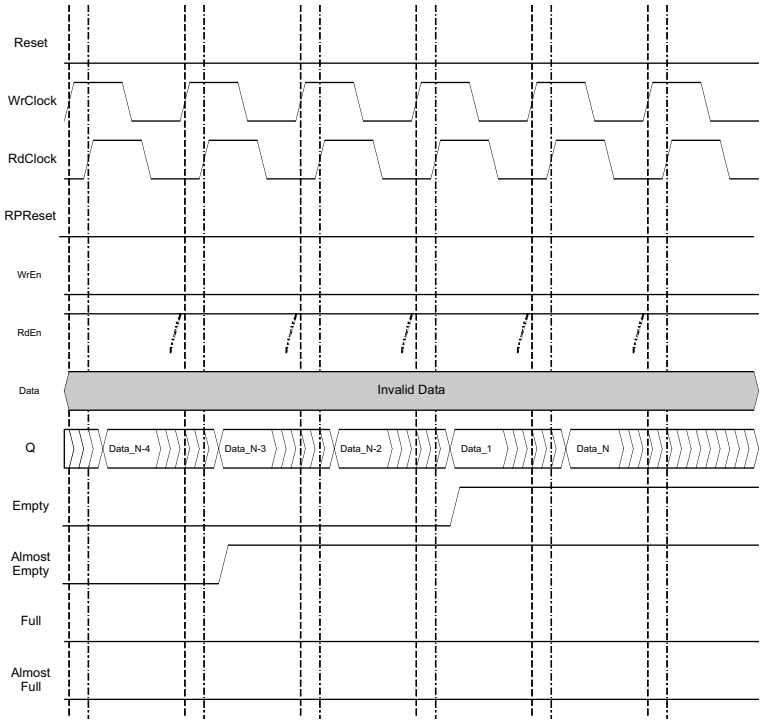
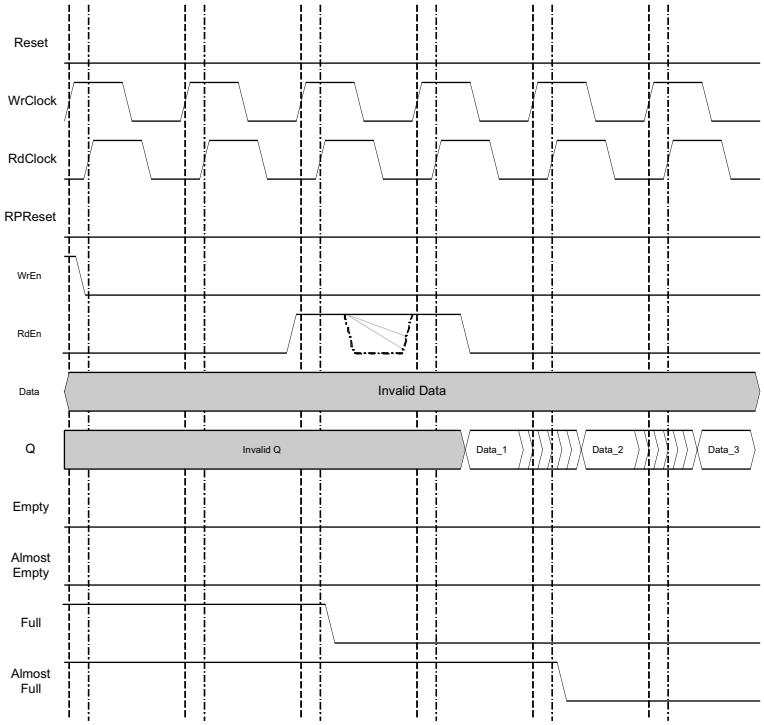


Figure 9-49. FIFO_DC With Output Registers, End of Data Read Cycle



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_DC), and the RdEn should be high also during that clock cycle. Otherwise the data takes an extra clock cycle when the RdEn is goes true.

Figure 9-50. FIFO_DC With Output Registers and RdEn on Output Registers

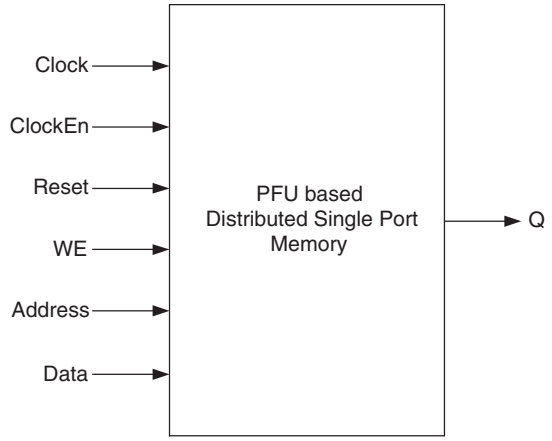


Distributed Single Port RAM (Distributed_SPRAM) – PFU Based

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. The memory's address and output registers are optional.

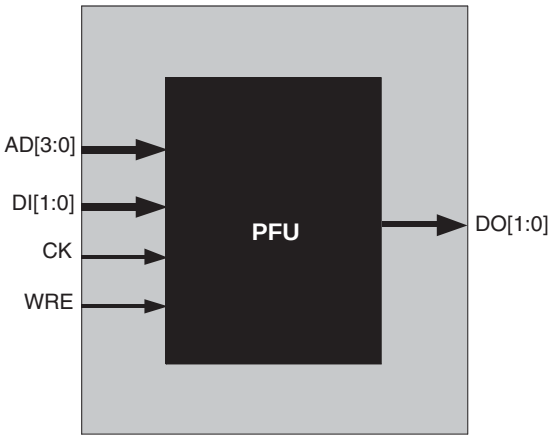
Figure 9-51 shows the Distributed Single Port RAM module as generated by the IPexpress.

Figure 9-51. Distributed Single Port RAM Module Generated by IPexpress



The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock, ClockEn and Reset is generated by utilizing the resources available in the PFU. The basic Distributed Single Port RAM primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-52.

Figure 9-52. Distributed Single Port RAM (Distributed_SPRAM) for LatticeECP/EC and LatticeXP Devices



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.

The various ports and their definitions for the memory are included in Table 9-14. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 9-14. PFU based Distributed Single port RAM Port Definitions

Port Name in Generated Module	Port Name in the EBR Block 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	—

Users have an option of enabling the output registers for Distributed Single Port RAM (Distributed_SPRAM). Figures 8-35 and 8-36 show the internal timing waveforms for the Distributed Single Port RAM (Distributed_SPRAM) with these options.

Figure 9-53. PFU Based Distributed Single Port RAM Timing Waveform - Without Output Registers

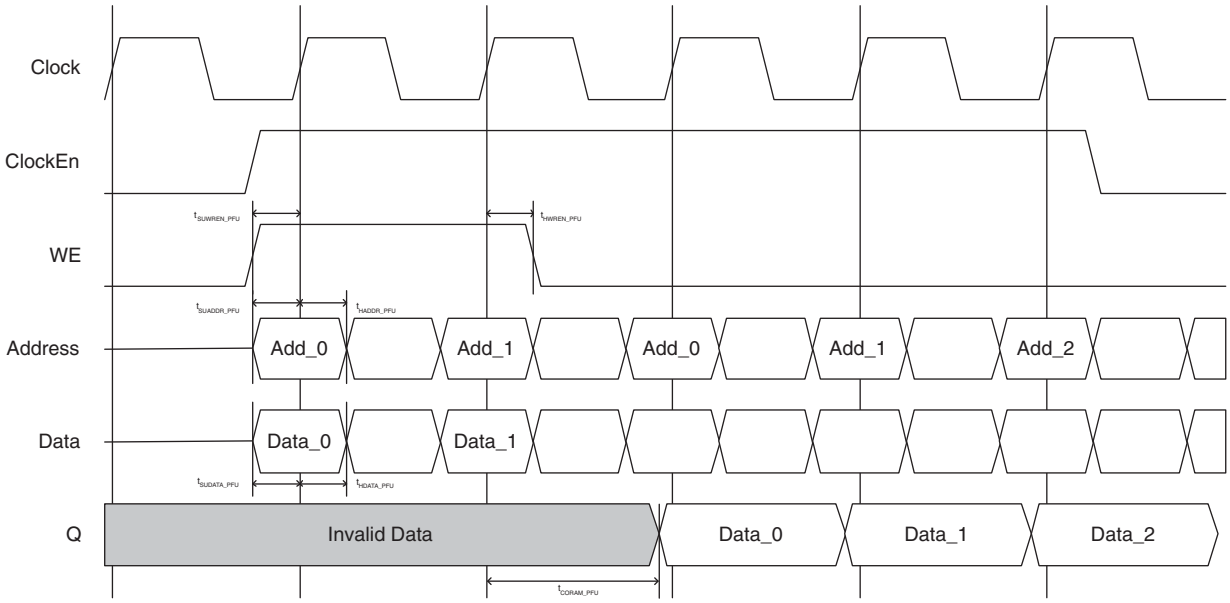
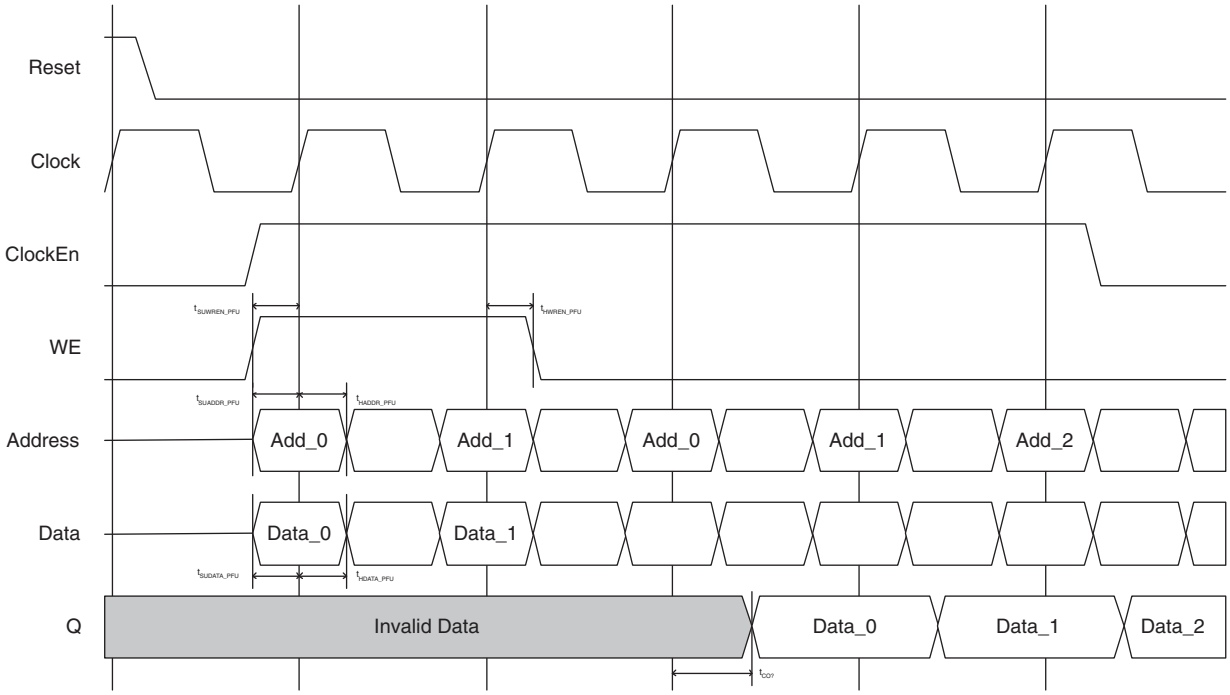


Figure 9-54. PFU Based Distributed Single Port RAM Timing Waveform – with Output Registers

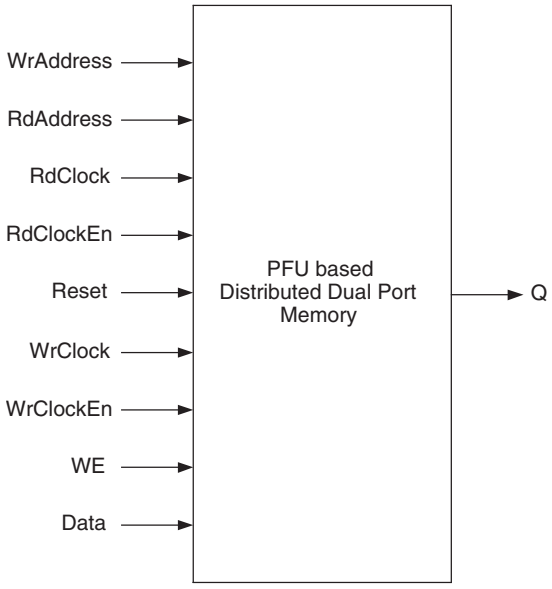


Distributed Dual Port RAM (Distributed_DPRAM) – PFU Based

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

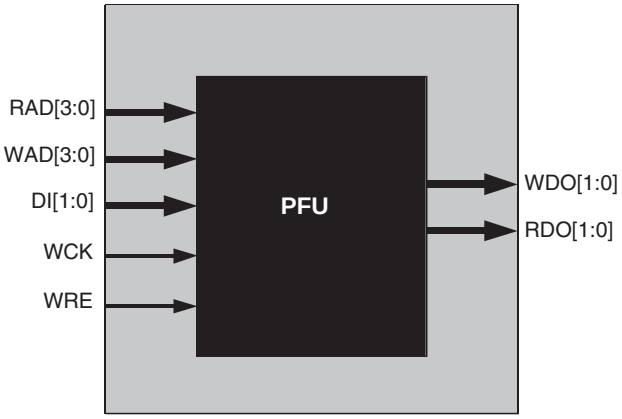
Figure 9-55 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 9-55. Distributed Dual Port RAM Module Generated by IPexpress



The generated module makes use of a 4-input LUT available in the PFU. Additional logic for Clocks, Clock Enables and Reset is generated by utilizing the resources available in the PFU. The basic Distributed Dual Port RAM primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-56.

Figure 9-56. PFU-based Distributed Dual Port RAM for LatticeECP/EC and LatticeXP Devices



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 the memory are included in Table 9-15. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 9-15. PFU-based Distributed Dual-Port RAM Port Definitions

Port Name in Generated Module	Port Name in EBR Block Primitive	Description	Active State
WrAddress	WAD[23:0]	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	—

Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed_DPRAM). Figures 8-39 and 8-40 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed_DPRAM) with these options.

Figure 9-57. PFU Based Distributed Dual Port RAM Timing Waveform - Without Output Registers (Non-Pipelined)

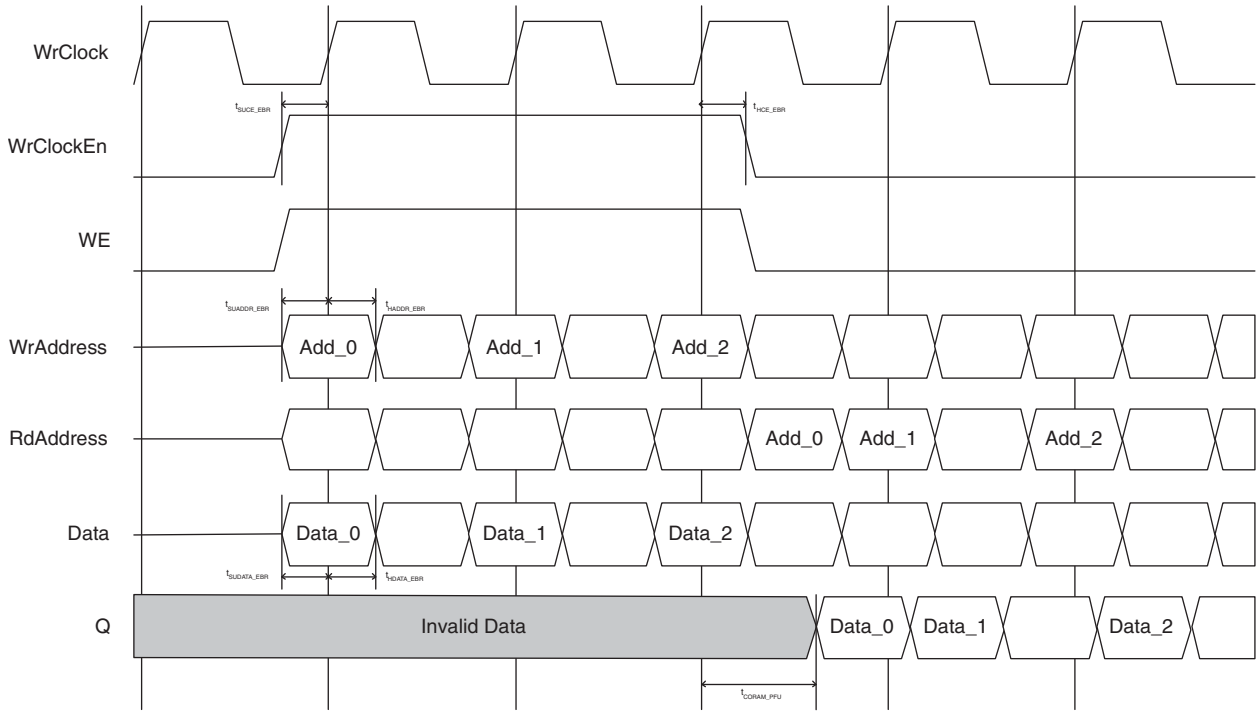
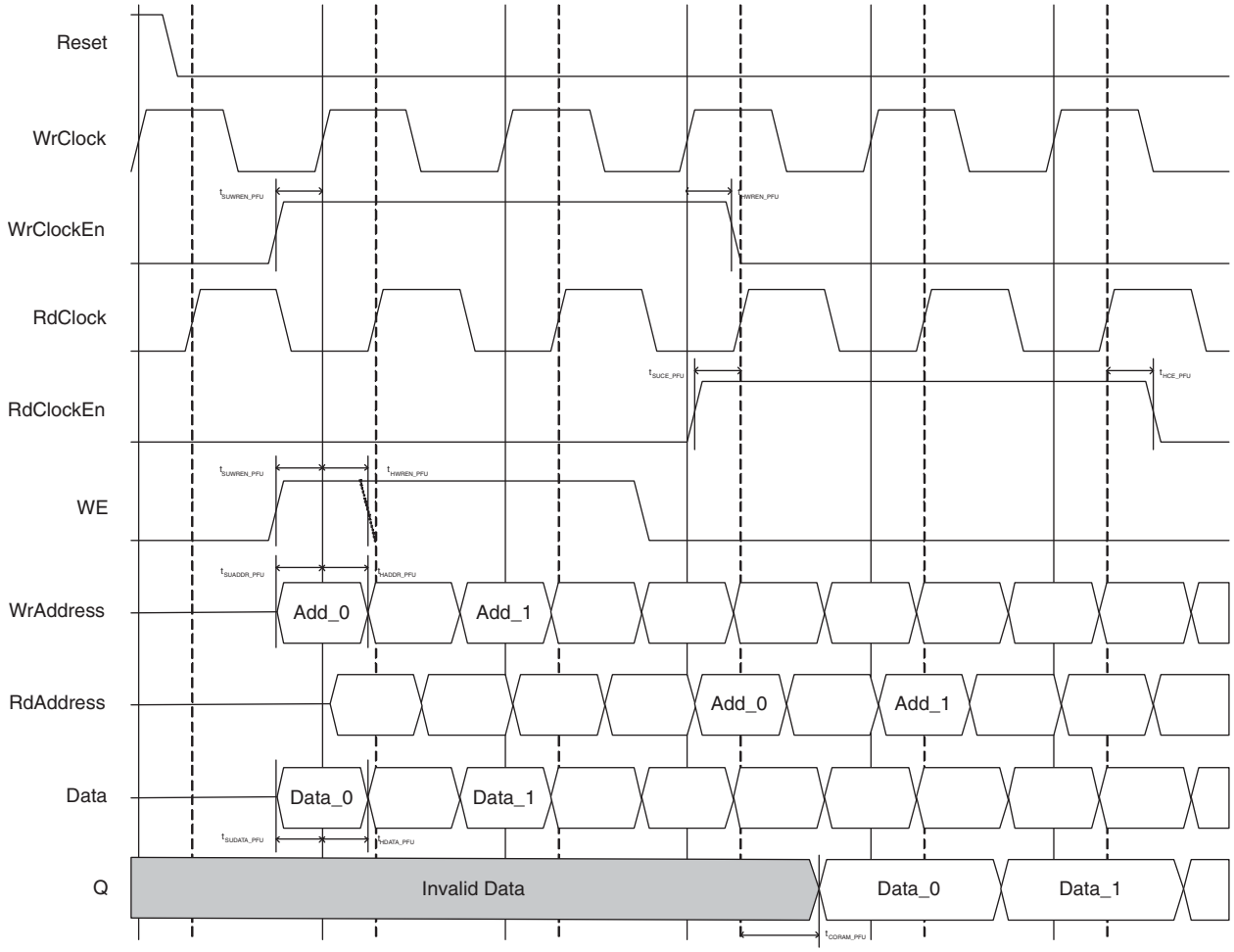


Figure 9-58. PFU Based Distributed Dual Port RAM Timing Waveform – With Output Registers

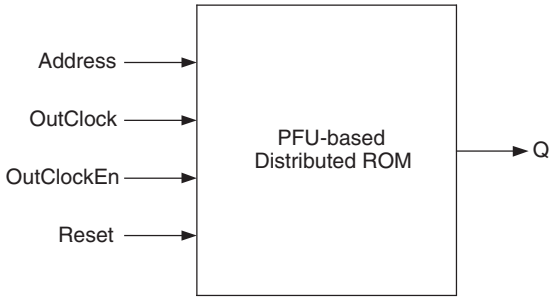


Distributed ROM (Distributed_ROM) – PFU Based

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.

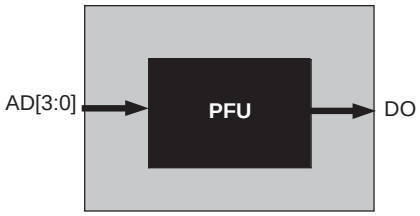
Figure 9-59 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 9-59. Distributed ROM Generated by IPexpress



The generated module makes use of the 4-input LUT available in the PFU. The basic Distributed Dual Port RAM primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-60.

Figure 9-60. PFU-based Distributed ROM (Sync_ROM) for LatticeECP/EC and LatticeXP Devices



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 the to enable the output registers in the IPexpress configuration.

The various ports and their definitions for the memory are included in Table 9-16. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 9-16. PFU-based Distributed ROM Port Definitions

Port Name in Generated Module	Port Name in the EBR 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 of enabling the output registers for Distributed ROM (Distributed_ROM). Figures 8-43 and 8-44 show the internal timing waveforms for the Distributed ROM with these options.

Figure 9-61. PFU Based ROM Timing Waveform – without Output Registers

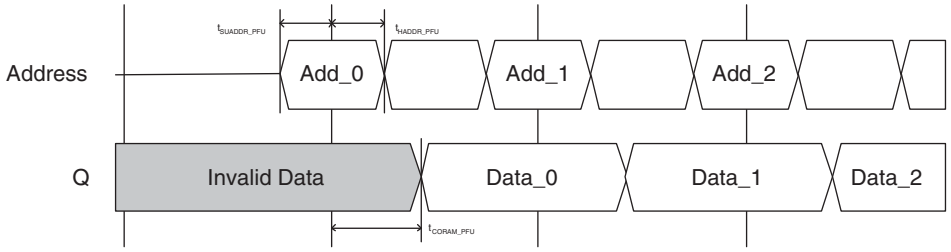
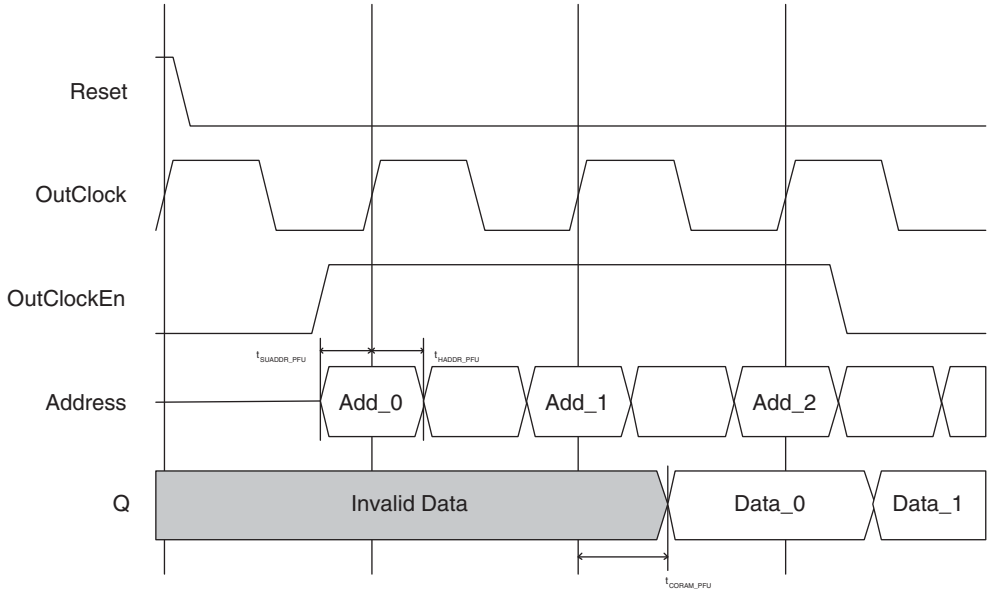


Figure 9-62. PFU Based ROM Timing Waveform – with Output Registers



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 a user can create or edit using any ASCII editor. IPexpress supports three types of memory file formats:

1. Binary file
2. Hex File
3. 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. 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 (ORCA)

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 a 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

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2004	01.0	Initial release.
July 2004	01.1	Minor updates only.
October 2004	01.2	Minor updates only.
February 2005	01.3	Replace LatticeEC™ and LatticeECP™ LatticeEC™, LatticeECP™ and LatticeXP™
		Replace LatticeECP/ECLatticeEC/ECP and LatticeXP
		Replace LatticeEC or LatticeECP LatticeEC, LatticeECP or LatticeXP
		Added Hardware related information for the LatticeEC/ECP/XP devices
		Update Figure 8-4
		Figure 8-40 Replace “_EBR” with “_PFU” in the figure’s timing parameters.
April 2005	01.4	Updated the Trual Dual Port RAM and Module Manager Flow sections.
October 2005	01.5	Updated the block diagrams of modules generated by the Module Manager. Added section for Module Manager flow example. Added section for PMI flow.
February 2006	01.6	Removed the PMI support section
April 2006	01.7	Updated the Initializing Memory section
October 2006	01.8	Updated the FIFO section. Added dual port memory access notes in Appendix A.
September 2012	01.9	Updated document with new corporate logo.

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 would form the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade 8 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, WRITETHROUGH, and READBEFOREWRITE 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.

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write address, during a write cycle. This mode is supported for x9, x18 and x36 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 Global Set/ Reset attribute is used to enable or disable the global set/reset for RAM element.