LatticeECP3 Memory User Guide

Technical Note

FPGA-TN-02188-2.0

June 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
CS	Chip Select
EBR	Embedded Block RAM
ECC	Error Check Code
EDIF	Electronic Design Interchange Format
FPGA	Field Programmable Gate Array
FIFO	First In First Out
GUI	Graphical User Interface
IP	Intellectual Property
LSB	Least Significant Bit
LUT	Look Up Table
MSB	Most Significant Bit
PFF	Programmable Functional Unit without RAM
PFU	Programmable Functional Unit
PLL	Phase-Locked Loops
PMI	Parameterizable Module Inferencing
RAM	Random-Access Memory
ROM	Read-Only Memory
RST	Reset
VHDL	VHSIC (Very High Speed Integrated Circuit) Hardware Description Language

1. Introduction

This technical note discusses memory usage for the LatticeECP3™ family of FPGA devices. It is intended to be used by design engineers as a guide to integrating the EBR- and PFU-based memories for this device family in the Lattice Diamond® software.

The LatticeECP3 architecture provides many resources for memory-intensive applications. The sysMEM™ Embedded Block RAM (EBR) compliments its distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM and ROM memories can be constructed using the EBR. The LUTs and PFUs 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 in this document. You can utilize the memory primitives in several ways:

- **Through IPexpress™** – The IPexpress GUI allows you to specify the memory type and size required. IPexpress uses this information to construct a netlist to implement the desired memory by using one or more of the memory primitives.
- **Through Parameterizable Module Inferencing (PMI)** – PMI allows experienced users to skip the graphical interface and utilize the configurable memory primitives on-the-fly from the Lattice Diamond software Project Navigator. The parameters and the control signals needed either in Verilog or VHDL can be set. The top-level design has the parameters defined and signals declared so the interface can automatically generate the black box during synthesis.
- **Through the Instantiation of Memory Primitives** – Memory primitives are called directly by the top-level module and instantiated in your design. This is an advanced method and requires a thorough understanding of memory hook-ups and design interfaces.

The remainder of this document discusses these approaches.

2. Utilizing IPexpress

You can utilize IPexpress to easily specify a variety of memories in your designs. These modules are constructed using one or more memory primitives along with general purpose routing and Look-Up Tables (LUTs) as required. The primitives include:

- 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

2.1. IPexpress Flow

For generating any of these memories, create or open a project for the LatticeECP3 devices.

From the Project Navigator, select **Tools** > **IPexpress**. Alternatively, you can also click the IPexpress button in the toolbar when LatticeECP3 devices are targeted in the project.

This opens the IPexpress window, as shown in Figure 2.1.

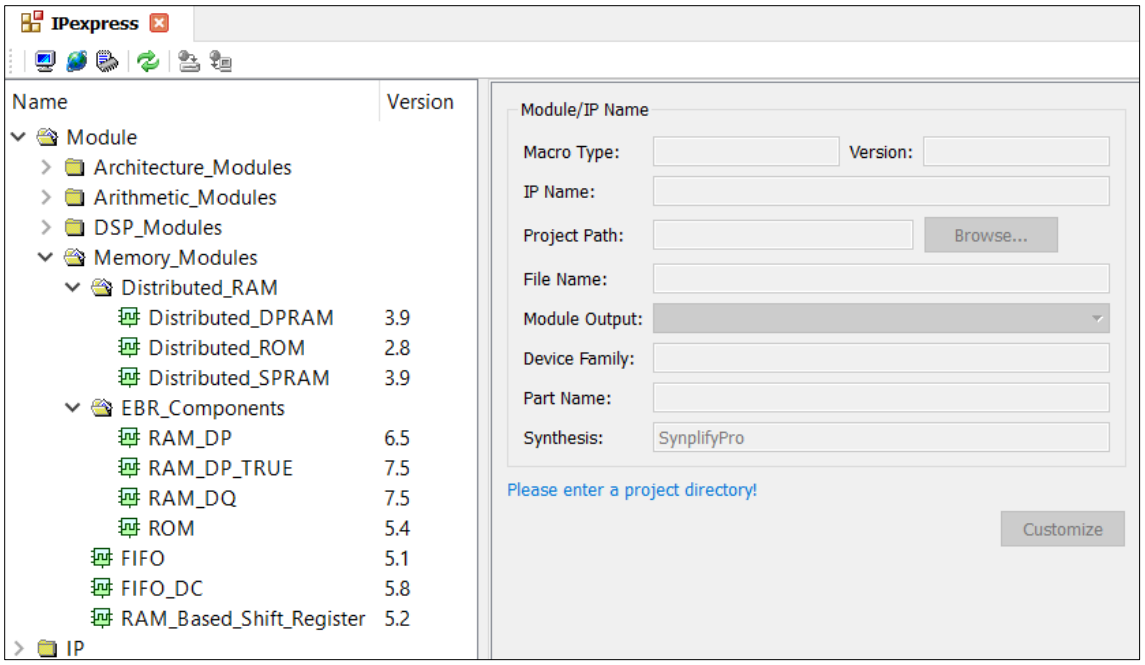

Figure 2.1. IPexpress Main Window

The left pane of this window includes the Module Tree. The EBR-based Memory Modules are under the **EBR_Components** directory and the PFU-based Distributed Memory Modules are under the **Distributed_RAM** directory, as shown in Figure 2.1.

As an example, consider generating an EBR-based Pseudo Dual-Port RAM of size 512x18. Select **RAM_DP** under **EBR_Components**. The right pane changes, as shown in Figure 2.2.

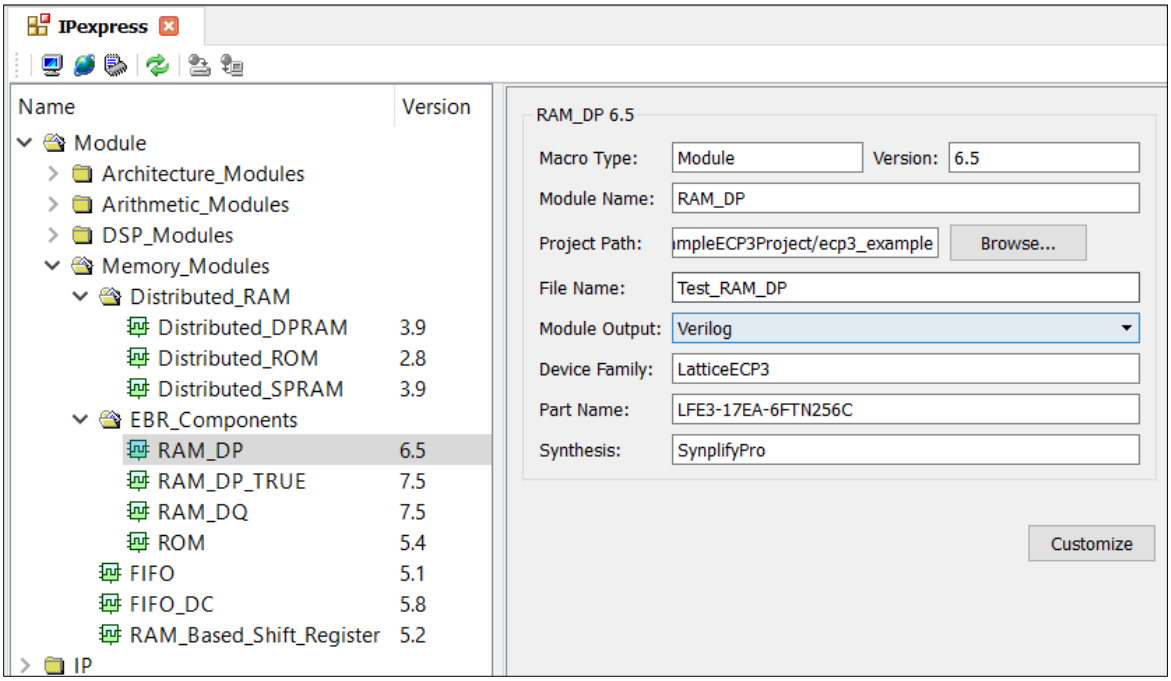

Figure 2.2. Example: Generating Pseudo Dual-Port RAM (RAM_DP) Using IPexpress

In the right pane, options including **Macro Type**, **Version**, **Module Name**, **Device Family**, **Part Name**, and **Synthesis** are device and selected module dependent. These cannot be changed in IPexpress.

You can change the directory where the generated module files are placed by clicking the **Browse** button in the **Project Path** field.

The entity name for the module to be generated can be specified in the **File Name** text box. You must provide this entity name. For **Module Output**, select either **Verilog** or **VHDL**.

By clicking the **Customize** button, another window opens where you can customize the RAM (Figure 2.3).

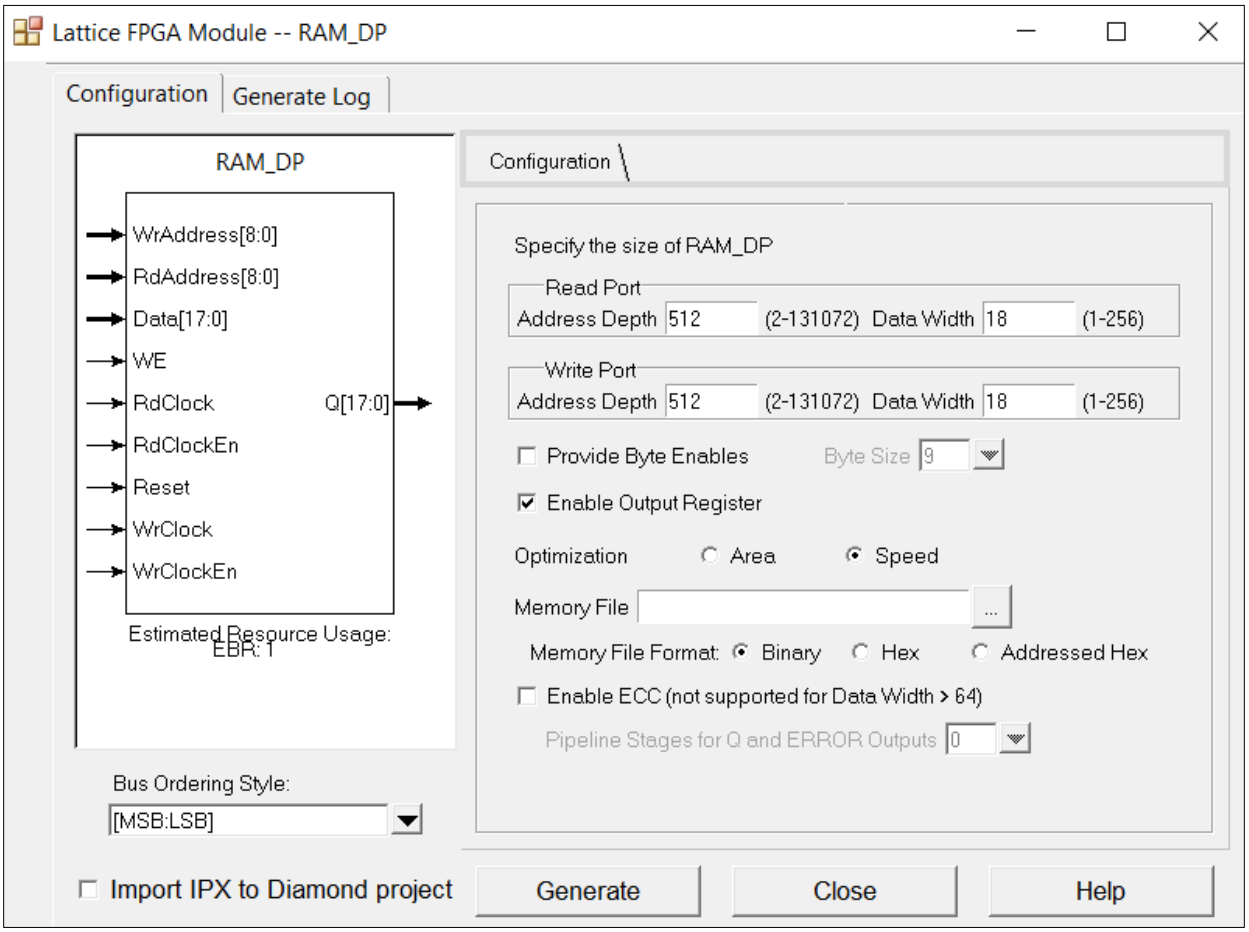

Figure 2.3. Example: Generating Pseudo Dual-Port RAM (RAM_DP) Module Customization – General Options

The left side of this window shows the block diagram of the module. The right side includes the **Configuration** tab where you can select options to customize the RAM_DP, for example, specifying the address port sizes and data widths.

You can specify the address depth and data width for the **Read Port** and the **Write Port** in the text boxes provided. In this example, a Pseudo Dual-Port RAM of size 512 x 18 is generated. You 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 checkbox **Enable Output Registers** inserts the output registers in the Read Data Port. Output registers are optional for EBR-based RAMs.

You can also pre-initialize your 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 Formats section of this document.

At this point, you can click the **Generate** button to generate the module that you have customized. A VHDL or Verilog netlist is then generated and placed in the specified location. You can incorporate this netlist in your designs.

3. Utilizing the PMI

Parameterizable Module Inferencing allows experienced users to skip the graphical interface and utilize the configurable memory modules on-the-fly from the Lattice Diamond Project Navigator. The parameters and control signals needed can be set in either Verilog or VHDL. The top-level design includes the defined memory parameters and declared signals. The interface can then automatically generate the black box during synthesis and the Lattice Diamond software can generate the netlist on-the-fly. Lattice memories are the same as industry standard memories, so you can get the parameters for each module from any memory-related guide, which is available through the online help system.

To do this, create a Verilog or VHDL behavior code for the memory and the synthesis tool automatically identifies it as memory and synthesizes it as a distributed or EBR memory. Memory sizes smaller than 2K bits are automatically mapped to the Distributed mode and those larger than 2K bits are implemented using EBRs. This default option can be over-ridden using the RAM_STYLE attribute in Synplify. Refer to Appendix A. Attribute Definitions for the example code.

ECC in Memory Modules

IPexpress allows you to implement Error Check Codes (ECC) in the EBR-based memory modules. There is a checkbox to enable ECC in the configuration tab for the module.

If you choose to use ECC, you have a 2-bit error signal and the error codes are as below.

- Error[1:0]=00 – Indicates there is no error.
- Error[0]=1 – Indicates there is a 1-bit error which has been fixed.
- Error[1]=1 – Indicates there is a 2-bit error which cannot be corrected.

4. Memory Modules

4.1. Single-Port RAM (RAM_DQ) – EBR-Based

The EBR blocks in LatticeECP3 devices can be configured as Single-Port RAM, RAM_DQ. IPexpress allows you to generate the Verilog-HDL or VHDL along with the EDIF netlist for the memory size per design requirements.

IPexpress generates the memory module, as shown in Figure 4.1.

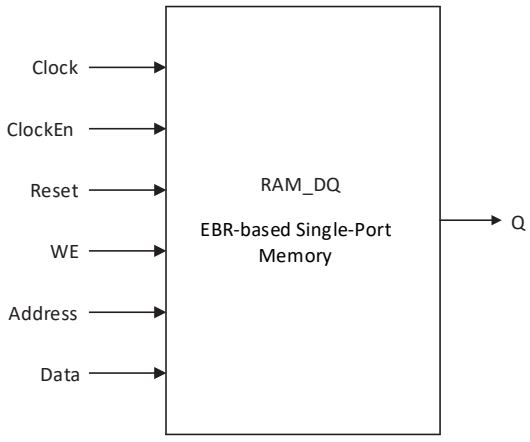

Figure 4.1. Single-Port Memory Module Generated by IPexpress

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 that you specify in the IPexpress GUI. For sizes smaller than an EBR block, the module is created in one EBR block and only one memory can be realized in the 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 inputs data and address 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 4.1. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DQ primitive.

Table 4.1. EBR-Based Single-Port Memory Port Definitions

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
Clock	CLK	Clock
ClockEn	CE	Clock Enable
Address	AD[x:0]	Address Bus
Data	DI[y:0]	Data In
Q	DO[y:0]	Data Out
WE	WE	Write Enable
Reset	RST	Reset
—	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 (or CS) is useful when the 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. If the memory size you specify requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU and external to the EBR blocks.

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) of each EBR block are listed in Table 4.2.

Table 4.2. Single-Port Memory Sizes for 18K Memories in LatticeECP3 Devices

Single-Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
16,384 x 1	DI	DO	AD[13:0]
8,192 x 2	DI[1:0]	DO[1:0]	AD[12:0]
4,096 x 4	DI[3:0]	DO[3:0]	AD[11:0]
2,048 x 9	DI[8:0]	DO[8:0]	AD[10:0]
1,024 x 18	DI[17:0]	DO[17:0]	AD[9:0]
512 x 36	DI[35:0]	DO[35:0]	AD[8:0]

Table 4.3 shows the various attributes available for the Single-Port Memory (RAM_DQ). Some of these attributes are user-selectable through the IPexpress GUI.

Table 4.3. Single-Port RAM Attributes of LatticeECP3 Devices

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
Memory File Format	—	BINARY, HEX, ADDRESSED HEX	—	Yes
Write Mode	Read/Write Mode for Write Port	NORMAL, WRITE-THROUGH, READBEFOREWRITE[1]	NORMAL	Yes
Chip Select Decode	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Init Value	Initialization value	0x00000000000000000000 00000000000000000000 00000000000000000000 0000000000000. 0xFFFF FFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFF	0x000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000	No

Note: READBEFOREWRITE Mode is available only for LatticeECP3-XXEA devices.

The Single-Port RAM (RAM_DQ) can be configured as NORMAL, WRITE THROUGH or READBEFOREWRITE 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, you can choose to enable the output registers for RAM_DQ. Figure 2.3 to Figure 4.2 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 the Lattice Diamond software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

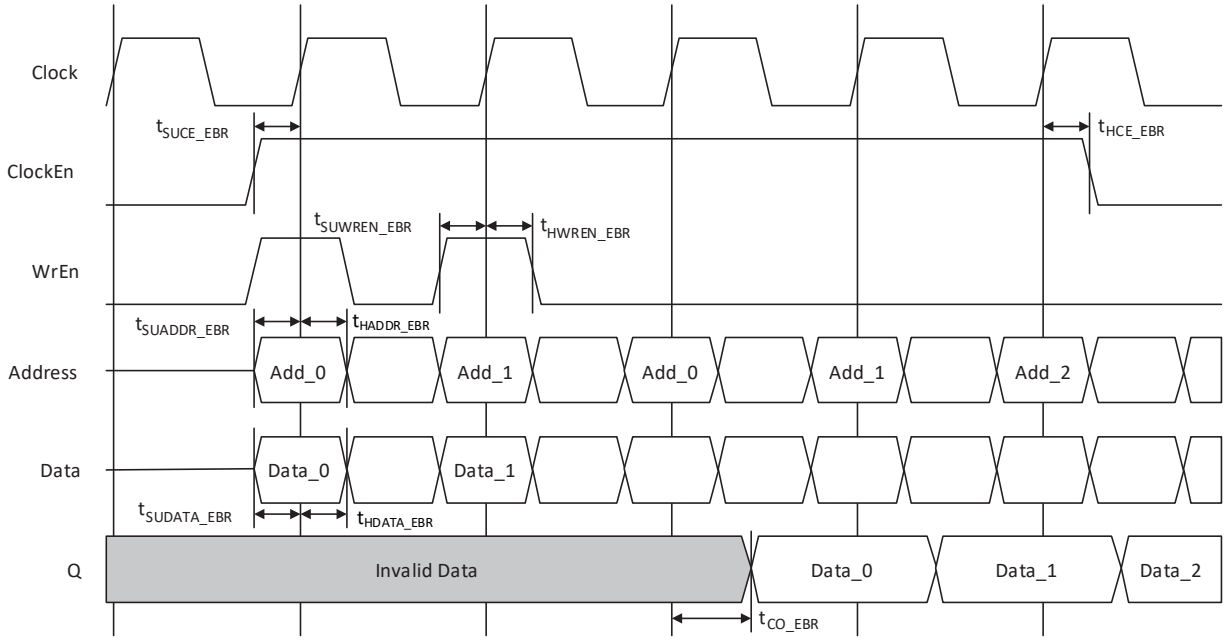

Figure 4.2. Single-Port RAM Timing Waveform in Normal (NORMAL) Mode – without Output Registers

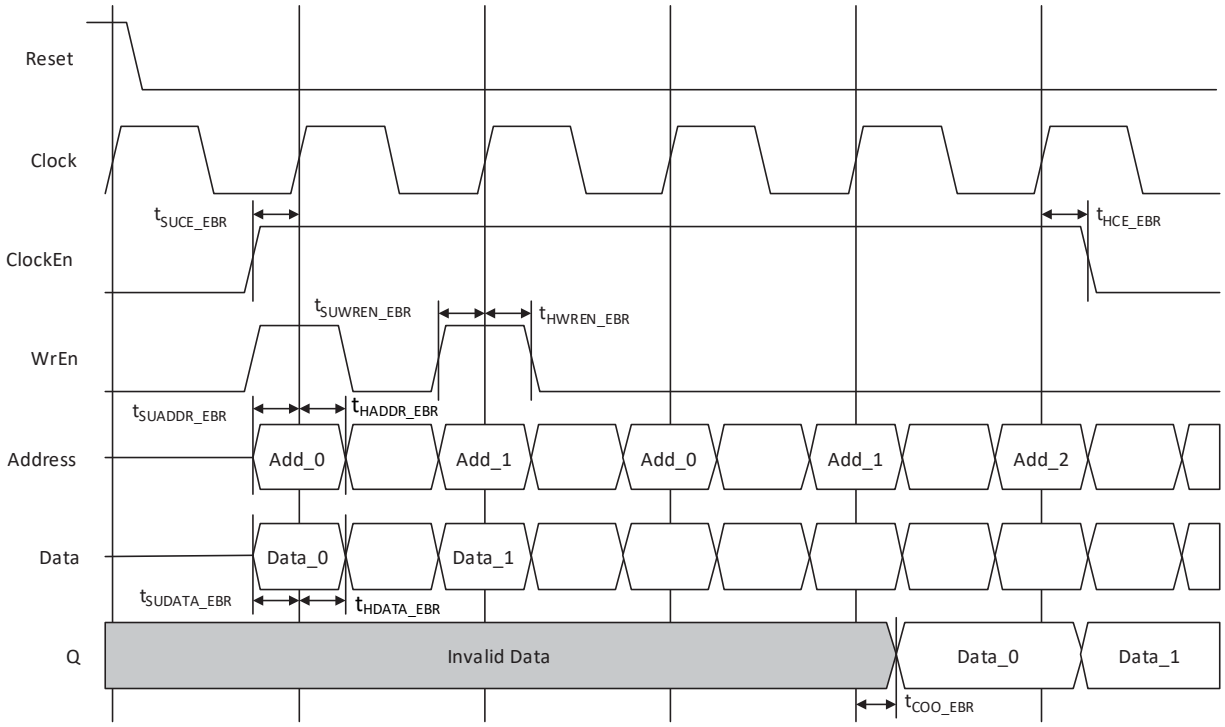

Figure 4.3. Single-Port RAM Timing Waveform in Normal (NORMAL) Mode – with Output Registers

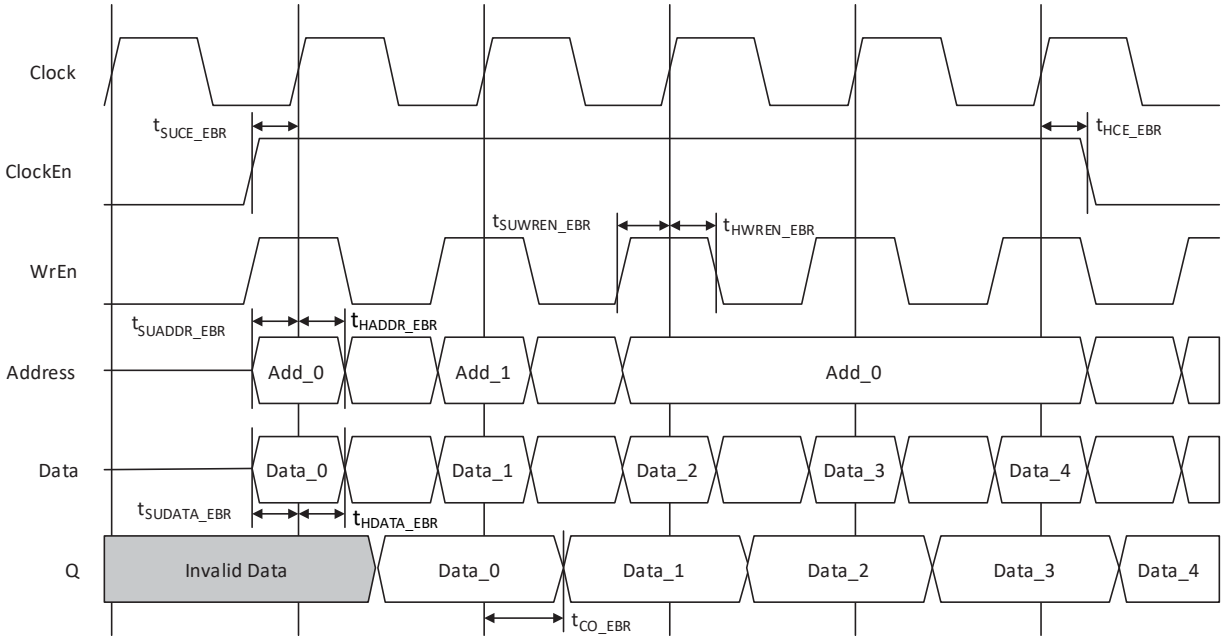

Figure 4.4. Single-Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode – without Output Registers

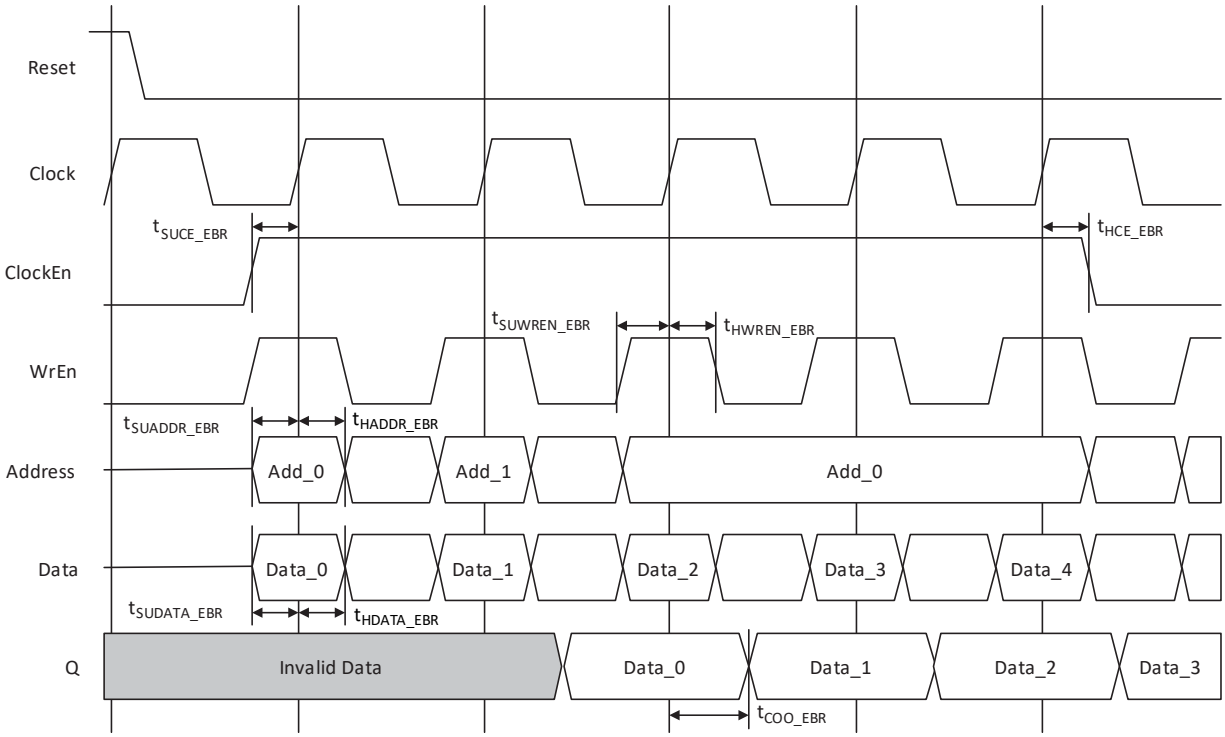

Figure 4.5. Single-Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode – with Output Registers

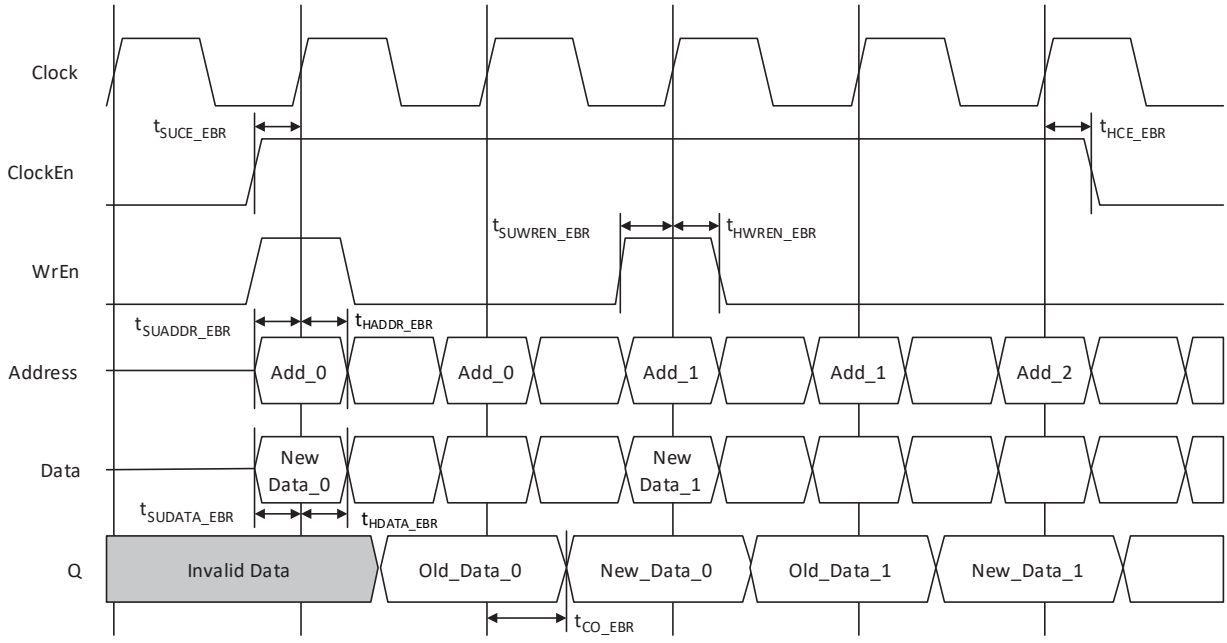

Figure 4.6. Single-Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode – without Output Registers

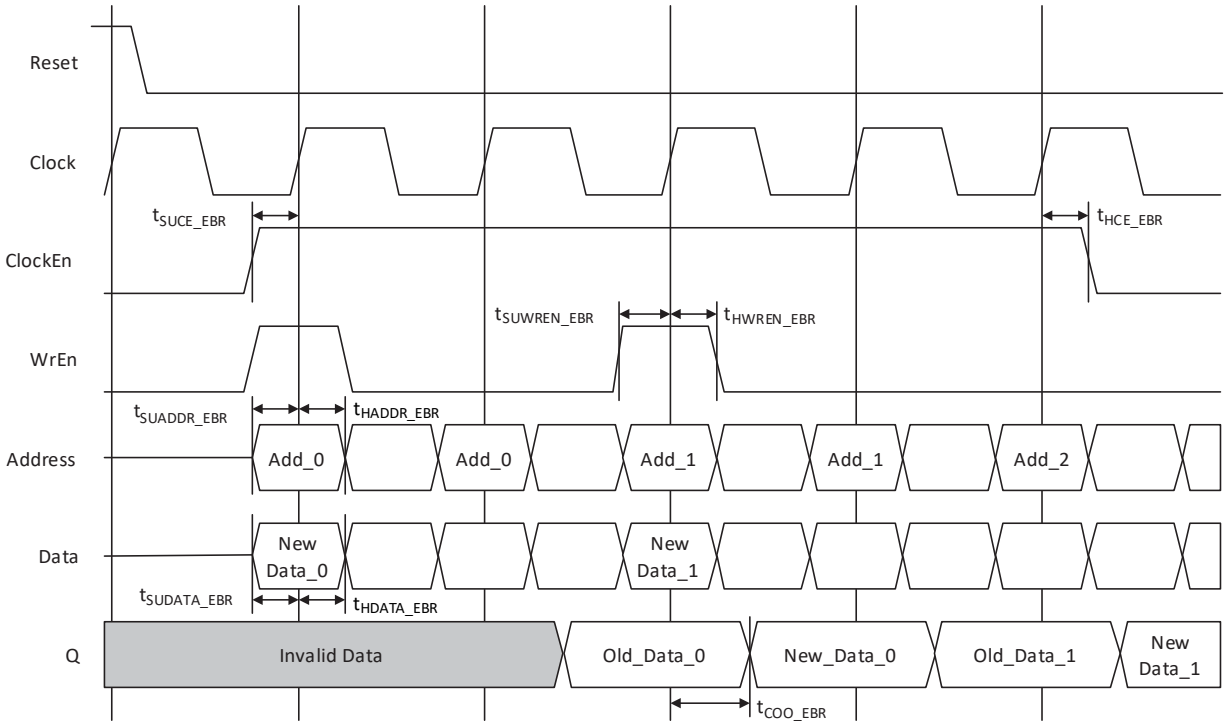

Figure 4.7. Single-Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode – with Output Registers

4.2. True Dual-Port RAM (RAM_DP_TRUE) – EBR-Based

The EBR blocks in the LatticeECP3 devices can be configured as True Dual-Port RAM, RAM_DP_TRUE. IPexpress allows you 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 4.8.

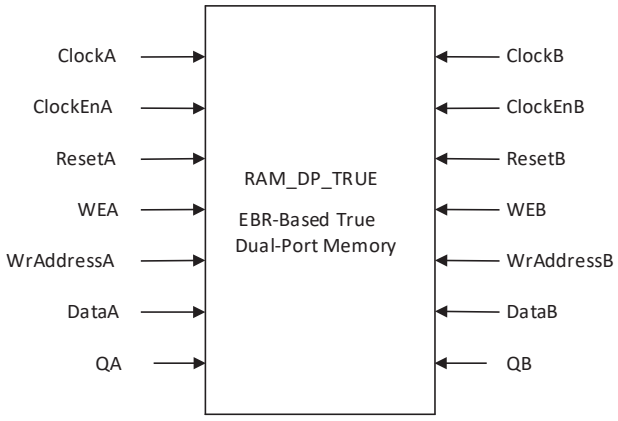

Figure 4.8. True Dual-Port Memory Module Generated by IPexpress

The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than one EBR block, the module is created in one EBR block. 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.

In True Dual-Port RAM mode, the inputs data and address 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 True Dual-Port RAM are listed in Table 4.4. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP_TRUE primitive.

Table 4.4. EBR-Based True Dual-Port Memory Port Definitions

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
ClockA, ClockB	CLKA, CLKB	Clock for Port A and Port B
ClockEnA, ClockEnB	CEA, CEB	Clock Enables for Port CLKA and CLKB
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
ResetA, ResetB	RSTA, RSTB	Reset for Port A and Port B
—	CSA[2:0], CSB[2:0]	Chip Selects for each port

CS (or Chip Select) is useful when the 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 you specify 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 18,432 bits of RAM. The values for x (address) and y (data) of each EBR block are listed in Table 4.5.

Table 4.5. Dual-Port Memory Sizes for 18K Memory in LatticeECP3 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]
16,384 x 1	DIA	DIB	DOA	DOB	ADA[13:0]	ADB[13:0]
8,192 x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[12:0]	ADB[12:0]
4,096 x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[11:0]	ADB[11:0]
2,048 x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[10:0]	ADB[10:0]
1,024 x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	ADA[9:0]	ADB[9:0]

Table 4.6 shows the various attributes available for True Dual-Port Memory (RAM_DQ). Some of these attributes are user-selectable through the IPexpress GUI.

Table 4.6. True Dual-Port RAM Attributes for LatticeECP3 Devices

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
Memory File Format	—	BINARY, HEX, ADDRESSED HEX	—	Yes
Port A Write Mode	Read/Write Mode for Port A	NORMAL, WRITE- THROUGH, READBEFOREWRITE1	NORMAL	Yes

Attribute	Description	Values	Default Value	User Selectable through IPexpress
Port B Write Mode	Read/Write Mode for Port B	NORMAL, WRITE- THROUGH, READBEFOREWRITE1	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	0x00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000 0xF FFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF	0x000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000	No

Note: READBEFOREWRITE Mode is available only for LatticeECP3-XXEA devices.

The True Dual-Port RAM (RAM_DP_TRUE) can be configured as NORMAL, WRITE THROUGH, or READBEFOREWRITE 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. Attribute Definitions.

Additionally, you can select to enable the output registers for RAM_DP_TRUE. Figure 4.9 to Figure 4.12 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 the Lattice Diamond software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

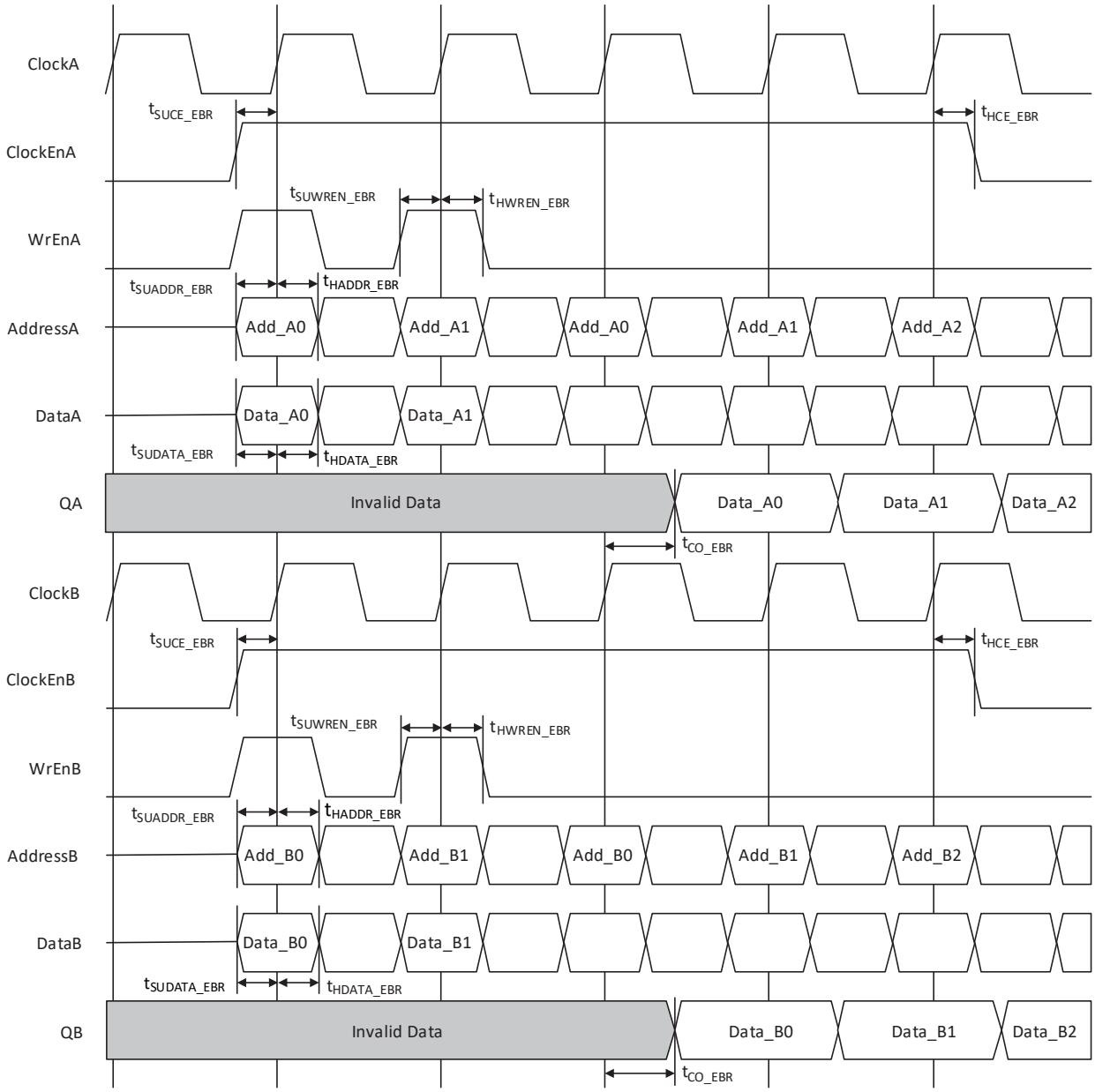

Figure 4.9. True Dual-Port RAM Timing Waveform in Normal (NORMAL) Mode – without Output Registers

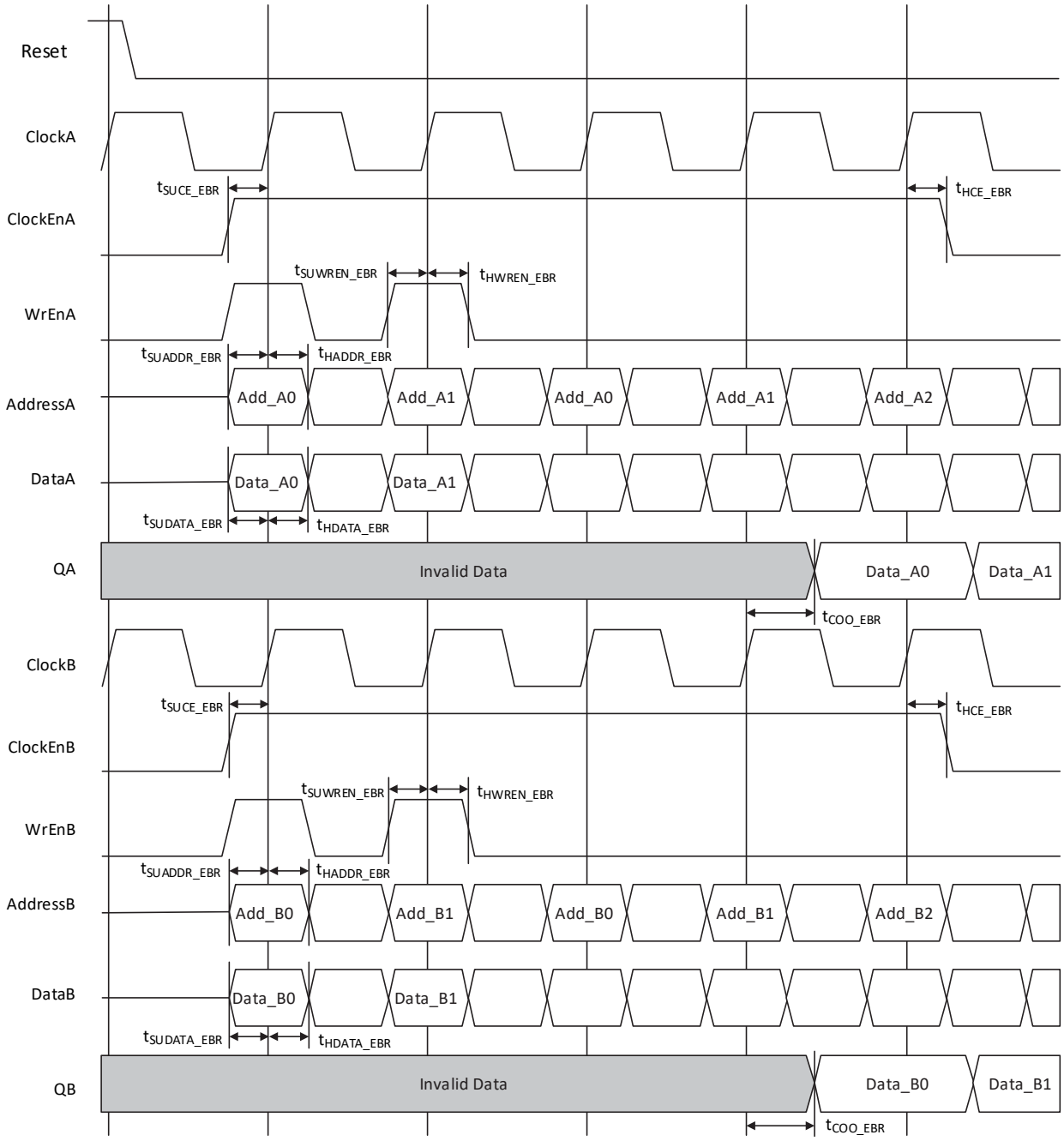

Figure 4.10. True Dual-Port RAM Timing Waveform in Normal (NORMAL) Mode – with Output Registers

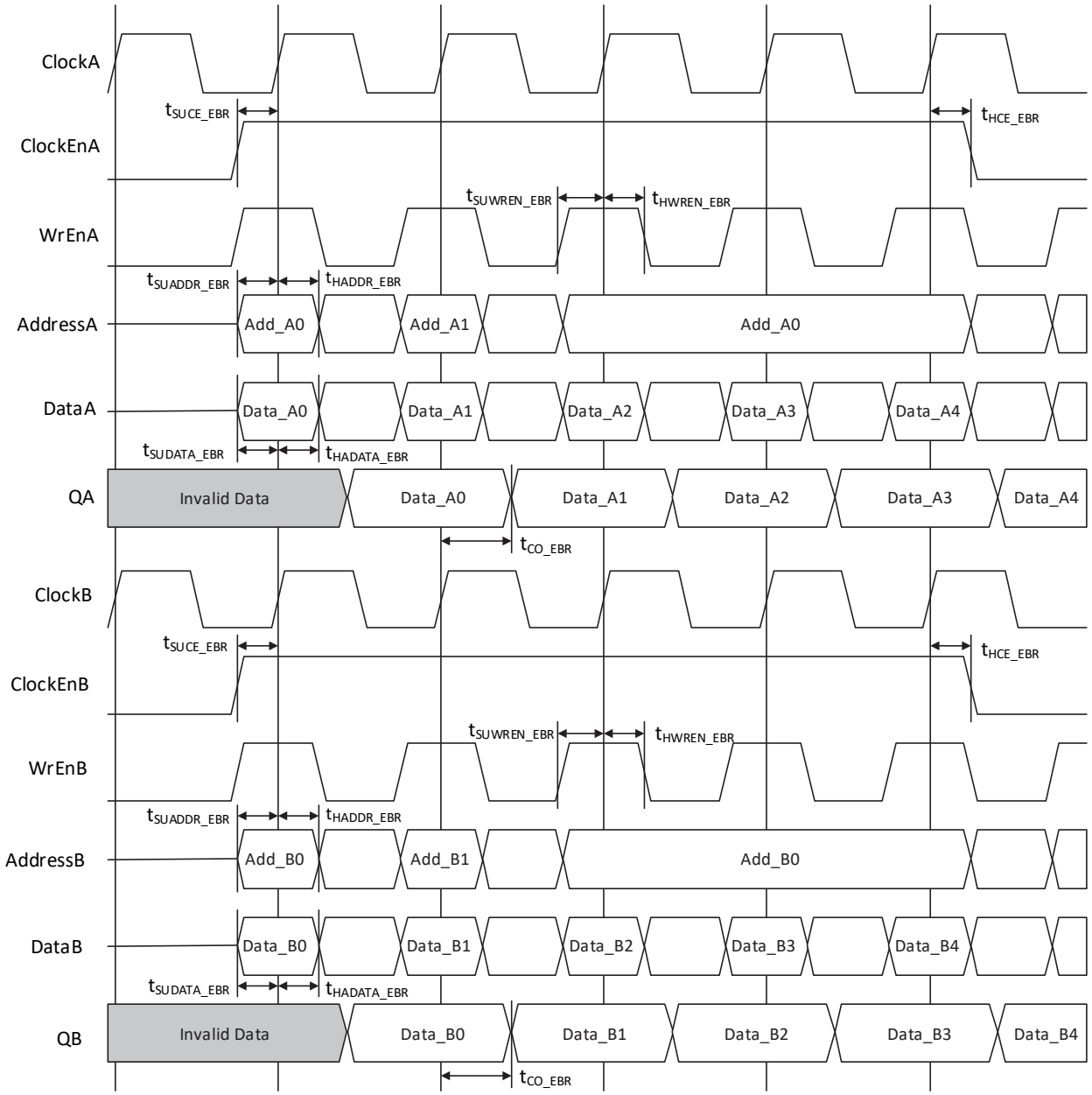

Figure 4.11. True Dual-Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode – without Output Registers

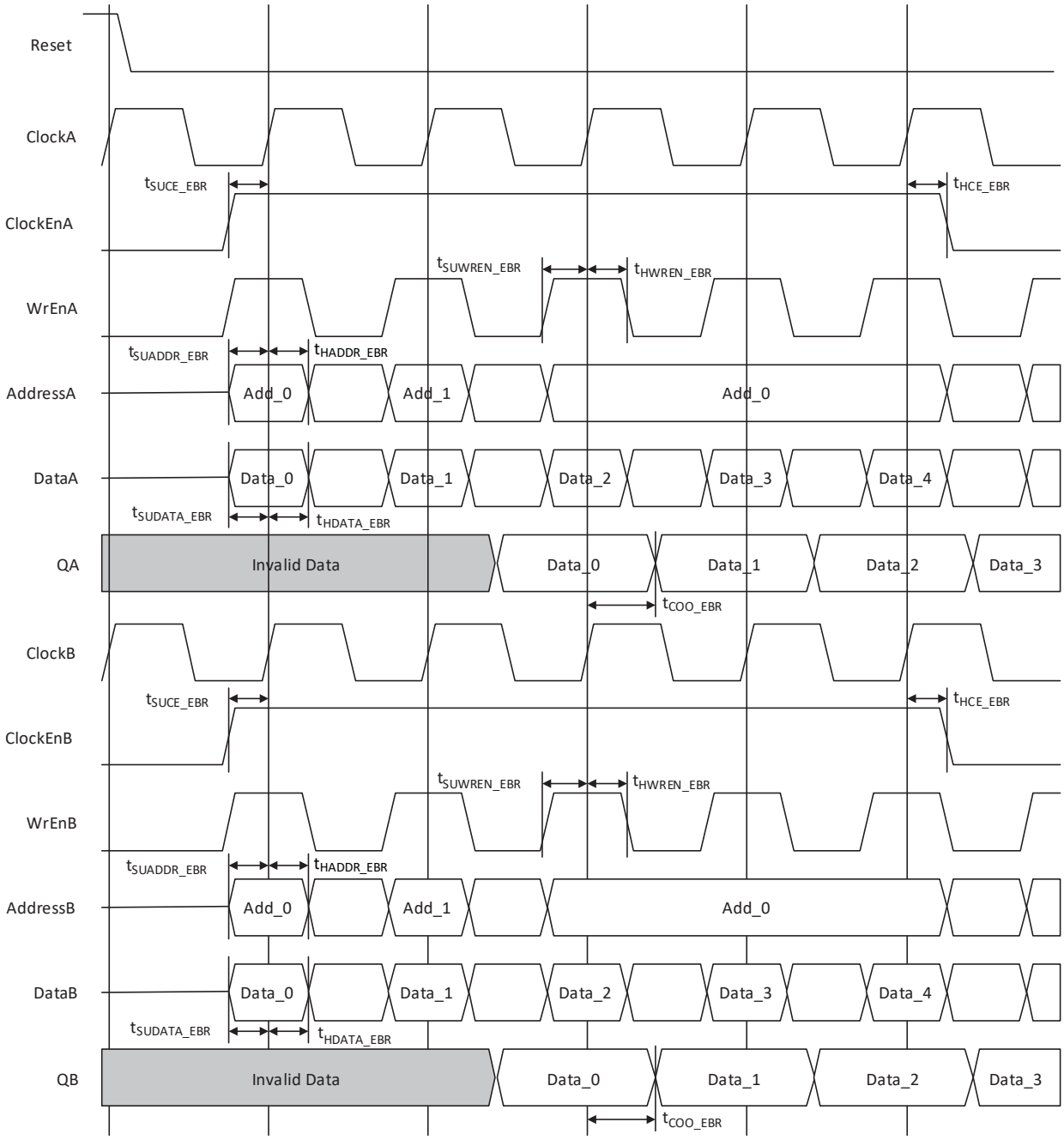

Figure 4.12. True Dual-Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode – with Output Registers

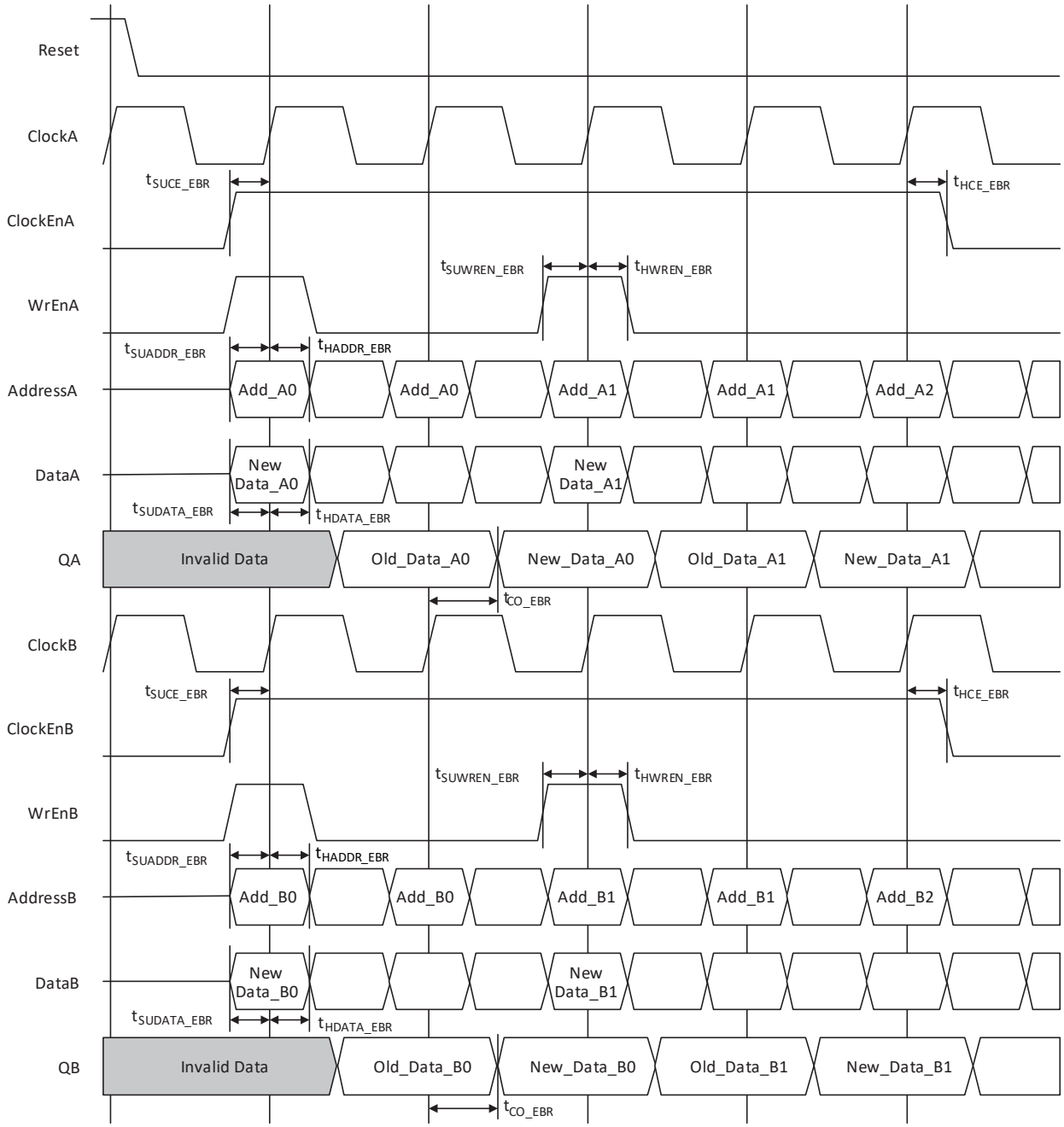

Figure 4.13. True Dual-Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode – without Output Registers

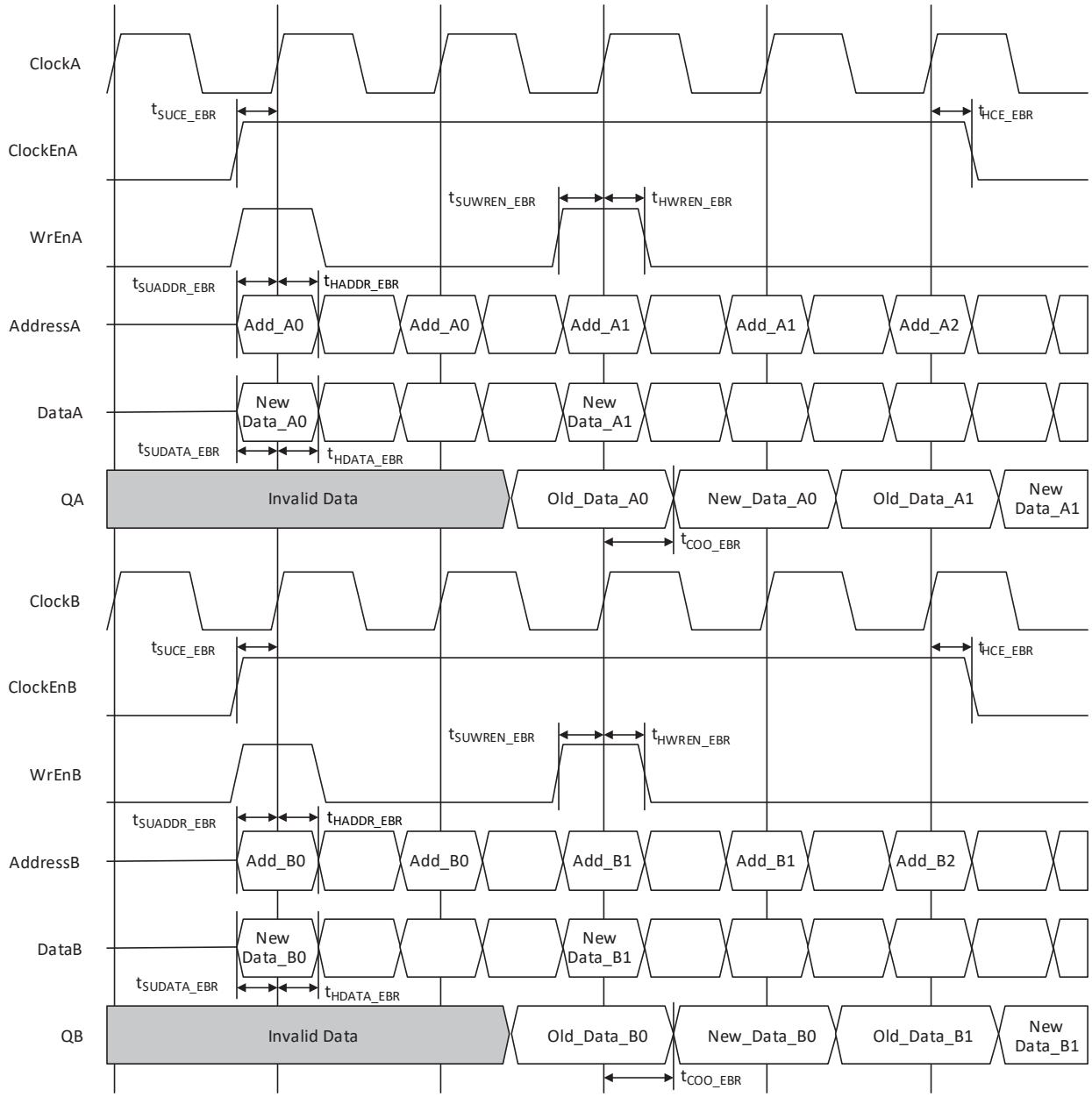

Figure 4.14. True Dual-Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode – with Output Registers

4.3. Pseudo Dual-Port RAM (RAM_DP) – EBR-Based

The EBR blocks in LatticeECP3 devices can be configured as Pseudo Dual-Port RAM or RAM_DP. IPexpress allows you to generate the Verilog-HDL or VHDL along with EDIF netlists for the memory size as per design requirements.

IPexpress generates the memory module shown in Figure 4.15.

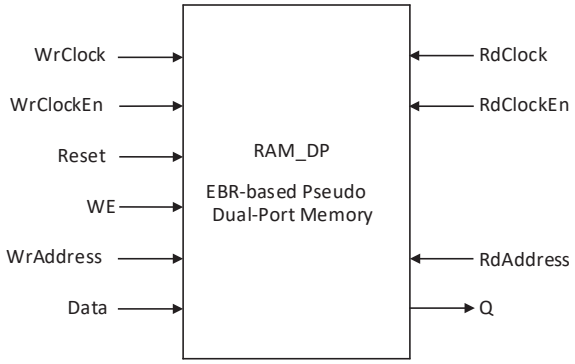

Figure 4.15. Pseudo Dual-Port Memory Module Generated by IPexpress

The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than one EBR block, the module is created in one EBR block. 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.

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 Pseudo Dual-Port memory are listed in Table 4.7. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP primitive.

Table 4.7. EBR-Based Pseudo Dual-Port Memory Port Definitions

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
RdAddress	ADR[x:0]	Read Address
WrAddress	ADW[x:0]	Write Address
RdClock	CLKR	Read Clock
WrClock	CLKW	Write Clock
RdClockEn	CER	Read Clock Enable
WrClockEn	CEW	Write Clock Enable
Q	DO[y:0]	Read Data
Data	DI[y:0]	Write Data
WE	WE	Write Enable
Reset	RST	Reset
—	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.

CS, or Chip Select, 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 you specify 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 18,432 bits of RAM. The values for x (address) and y (data) of each EBR block are listed in Table 4.8.

Table 4.8. Pseudo Dual-Port Memory Sizes for 18K Memory in LatticeECP3 Devices

Pseudo Dual-Port Memory Size	Input Data Port B	Output Data Port A	Read Address Port A [MSB:LSB]	Write Address Port B [MSB:LSB]
16,384 x 1	DIB	DOA	RAD[13:0]	WAD[13:0]
8,192 x 2	DIB[1:0]	DOA[1:0]	RAD[12:0]	WAD[12:0]
4,096 x 4	DIB[3:0]	DOA[3:0]	RAD[11:0]	WAD[11:0]
2,048 x 9	DIB[8:0]	DOA[8:0]	RAD[10:0]	WAD[10:0]
1,024 x 18	DIB[17:0]	DOA[17:0]	RAD[9:0]	WAD[9:0]
512 x 36	DIB[35:0]	DOA[35:0]	RAD[8:0]	WAD[8:0]

Table 4.9 shows the various attributes available for the Pseudo Dual-Port Memory (RAM_DP). Some of these attributes are user-selectable through the IPexpress GUI.

Table 4.9. Pseudo Dual-Port RAM Attributes for LatticeECP3 Devices

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
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	0x00000000000000000000000000000000000000000000000000000000000000000000000000000000…………..0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0x00000000000000000000000000000000000000000000000000000000000000000000000000000000	No

You have the option to enable the output registers for Pseudo Dual-Port RAM (RAM_DP). Figure 4.16 and Figure 4.17 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 the Lattice Diamond software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

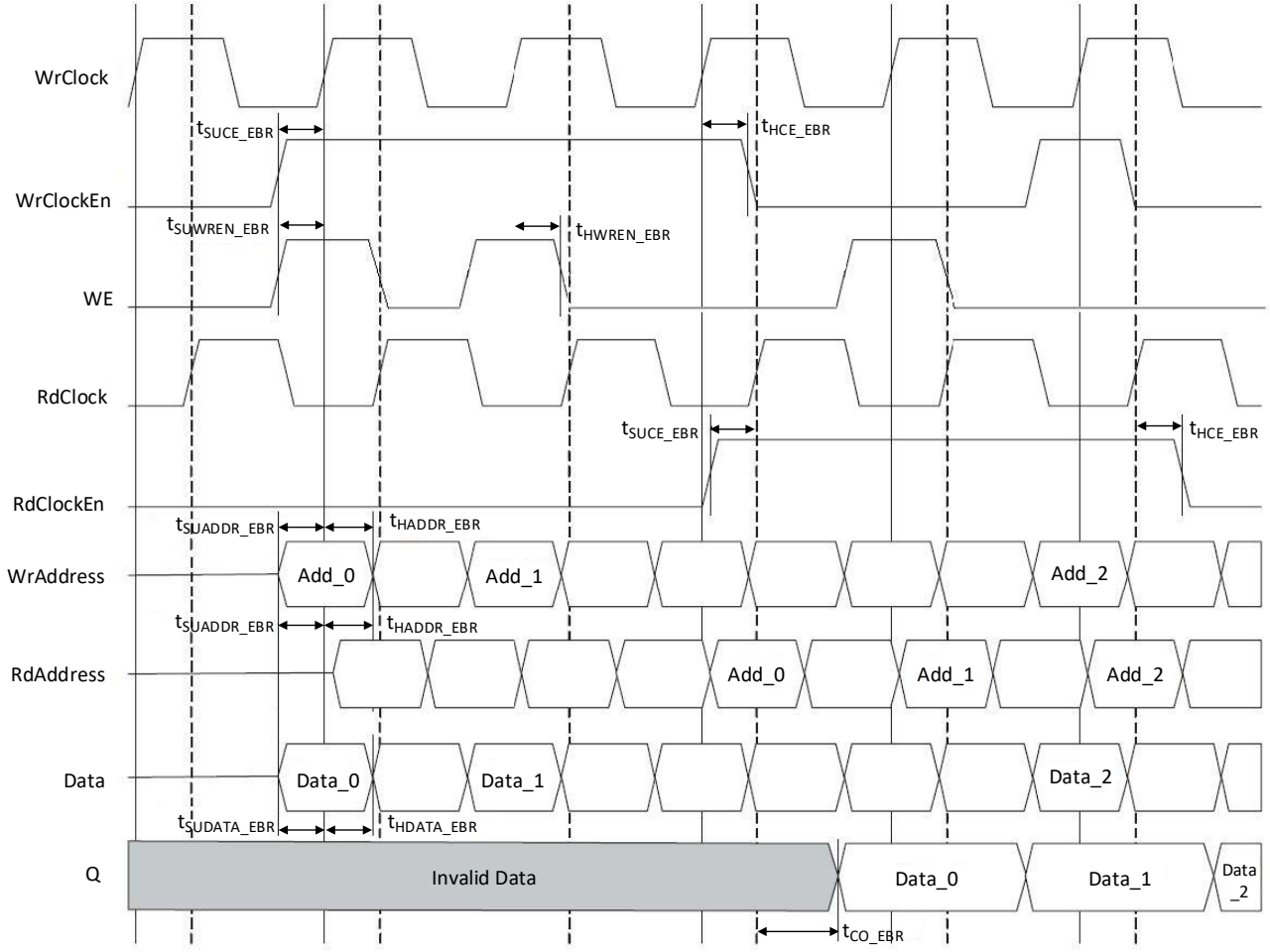

Figure 4.16. Pseudo Dual-Port RAM Timing Diagram – without Output Registers

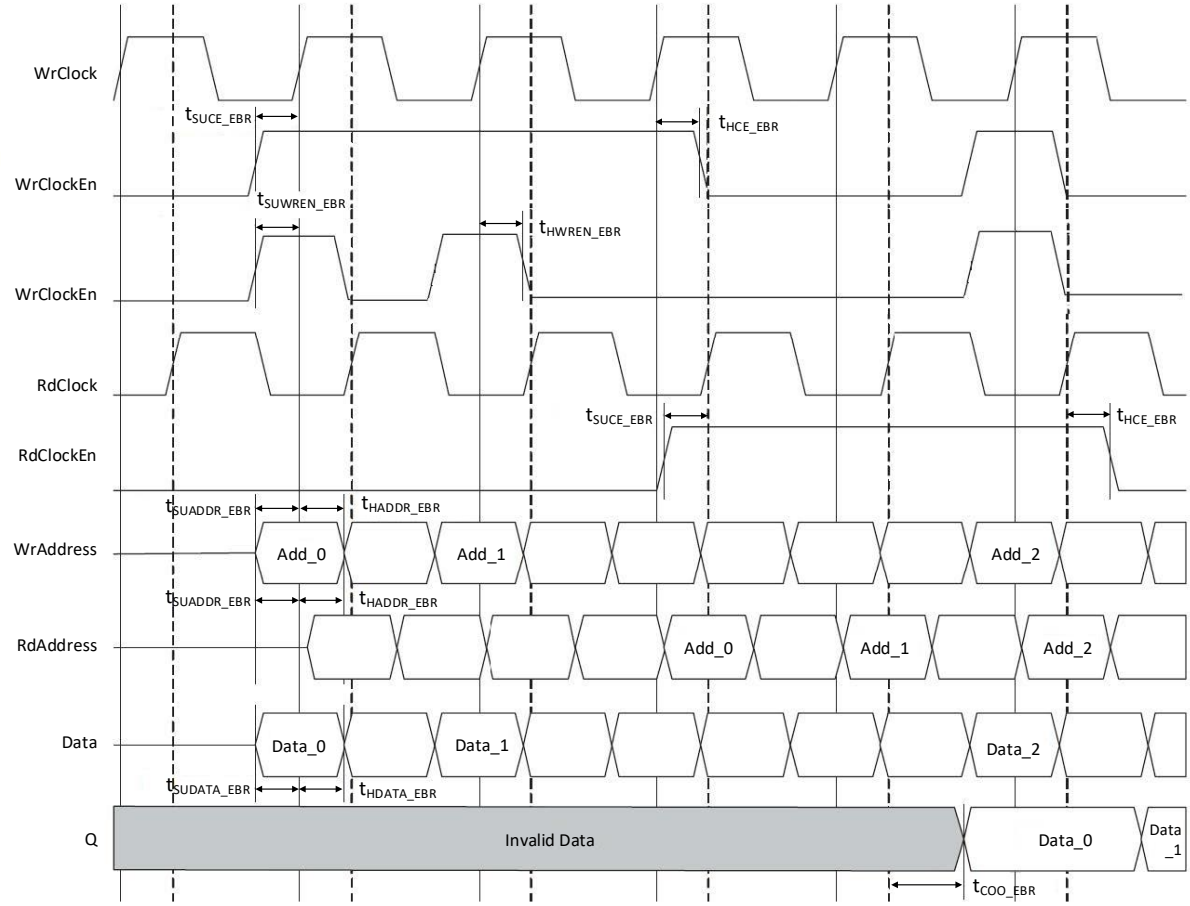

Figure 4.17. Pseudo Dual-Port RAM Timing Diagram – with Output Registers

4.4. Read-Only Memory (ROM) – EBR-Based

LatticeECP3 FPGAs support all the features of the ROM IPexpress or ROM. IPexpress allows you to generate Verilog-HDL, or VHDL, along with the EDIF netlist for the memory size required by your design. You are required to provide the ROM memory content in the form of an initialization file.

IPexpress generates the memory module shown in Figure 4.18.

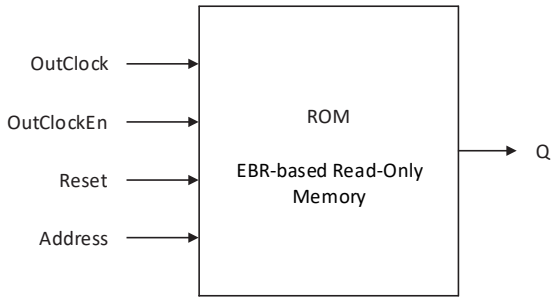

Figure 4.18. ROM – Read-Only Memory Module Generated by IPexpress

The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than one EBR block, the module is created in one EBR block. 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.

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 are listed in Table 4.10. The table lists the corresponding ports for the module generated by IPexpress and for the ROM primitive.

Table 4.10. EBR-Based ROM Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description
Address	AD[x:0]	Read Address
OutClock	CLK	Clock
OutClockEn	CE	Clock Enable
Reset	RST	Reset
—	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 you specified requires more than eight EBR blocks, the Lattice Diamond software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

When generating ROM using IPexpress, you 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 addressed hex formats. The initialization files are discussed in detail in the Initialization File Formats section of this document.

You have the option of enabling the output registers for Read-Only Memory (ROM). Figure 4.19 and Figure 4.20 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 (address) and y's (data) for each EBR block for the devices included in Table 4.11.

Table 4.11. ROM Memory Sizes for 16K Memory for LatticeECP3

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]

LatticeECP3 FPGAs have 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 can be emulated to be built around these RAMs. The IPexpress point tool in the Lattice Diamond software allows you to build a FIFO and FIFO_DC around Pseudo Dual-Port RAM (or DP_RAM).

Table 4.12 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. Attribute Definitions.

Table 4.12. ROM Attributes for LatticeECP3

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
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

You have the option to enable the output registers for Read-Only Memory (ROM). Figure 4.19 and Figure 4.20 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 the Lattice Diamond software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

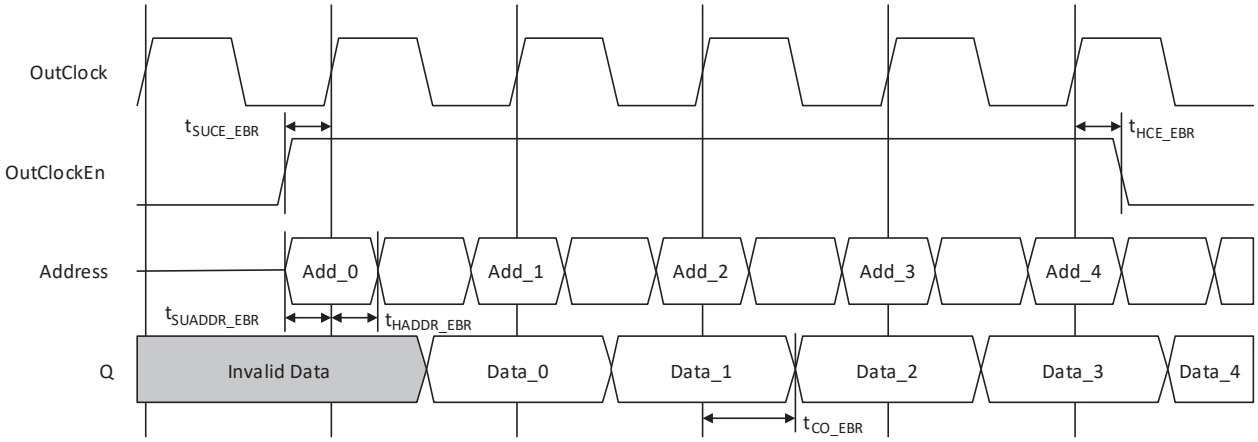

Figure 4.19. ROM Timing Waveform – without Output Registers

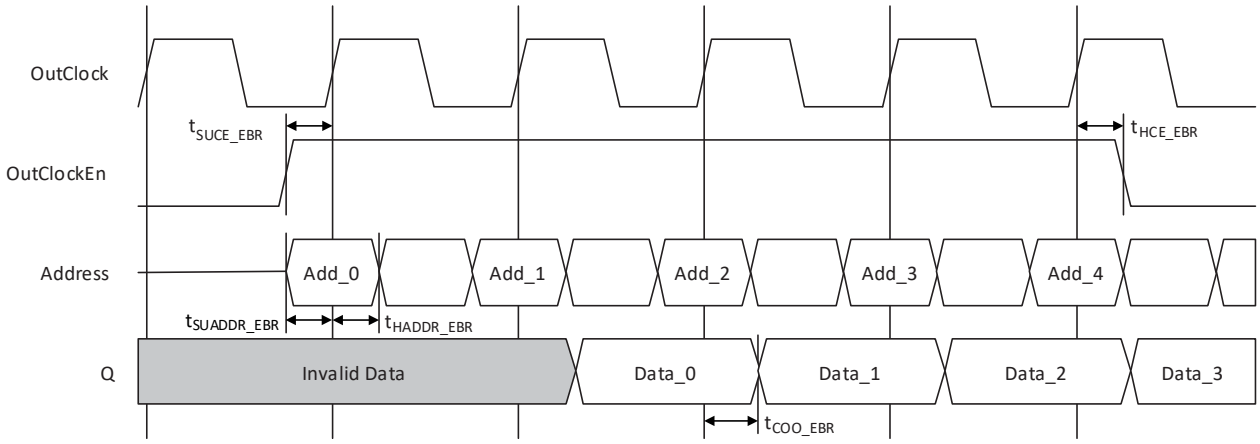

Figure 4.20. ROM Timing Waveform – with Output Registers

5. First In First Out (FIFO) Memory

The FIFO, or the single-clock FIFO, is an emulated FIFO. The address logic and 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

The non-pipelined or the FIFO without output registers is discussed first. Figure 5.1 shows the operation of the FIFO when it is empty and the data begins to be written into it.

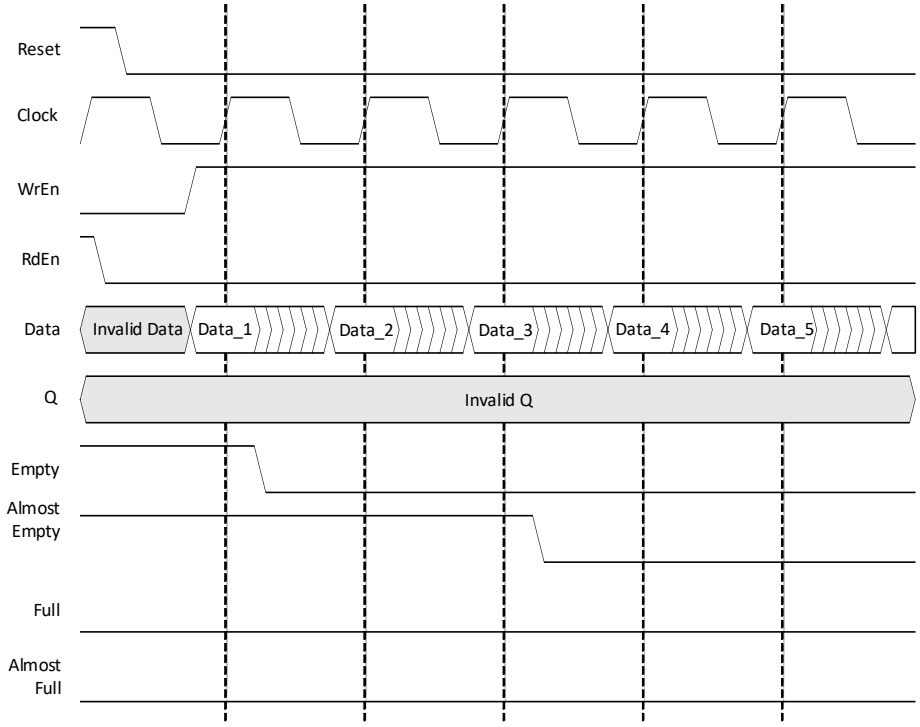

Figure 5.1. FIFO Without Output Registers, Start of data Write Cycle

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 since the FIFO is no longer empty. In this figure it is assumed that the Almost Empty flag setting is three, address location three. So, the Almost Empty flag is de-asserted when the third address location is filled.

Assume continuing to write into the FIFO to fill it. When the FIFO is filled, the Almost Full and Full flags are asserted. Figure 5.2 shows the behavior of these flags. In this figure it is assumed that the FIFO depth is N.

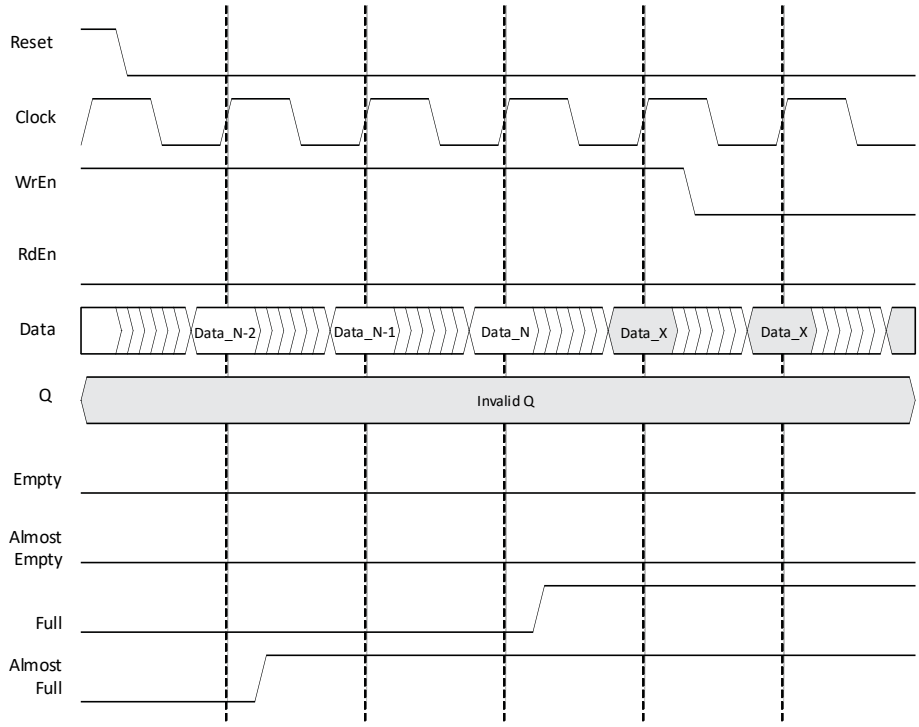

Figure 5.2. FIFO Without Output Registers, End of Data Write Cycle

In Figure 5.2, the Almost Full flag is two locations 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 are not written since the FIFO is full, that is, the Full flag is high.

Next, consider the waveforms when the contents of the FIFO are read out. Figure 5.3 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 Figure 5.3.

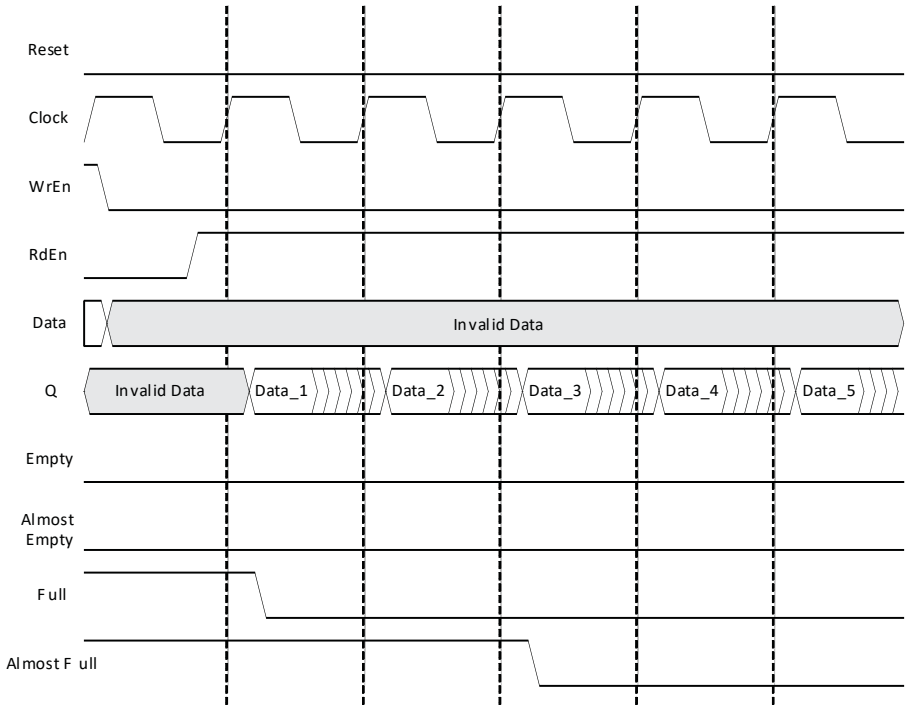

Figure 5.3. FIFO Without Output Registers, Start of Data Read Cycle

Similarly, as the data is read out and FIFO is emptied, the Almost Empty and Empty flags are asserted (see Figure 5.4).

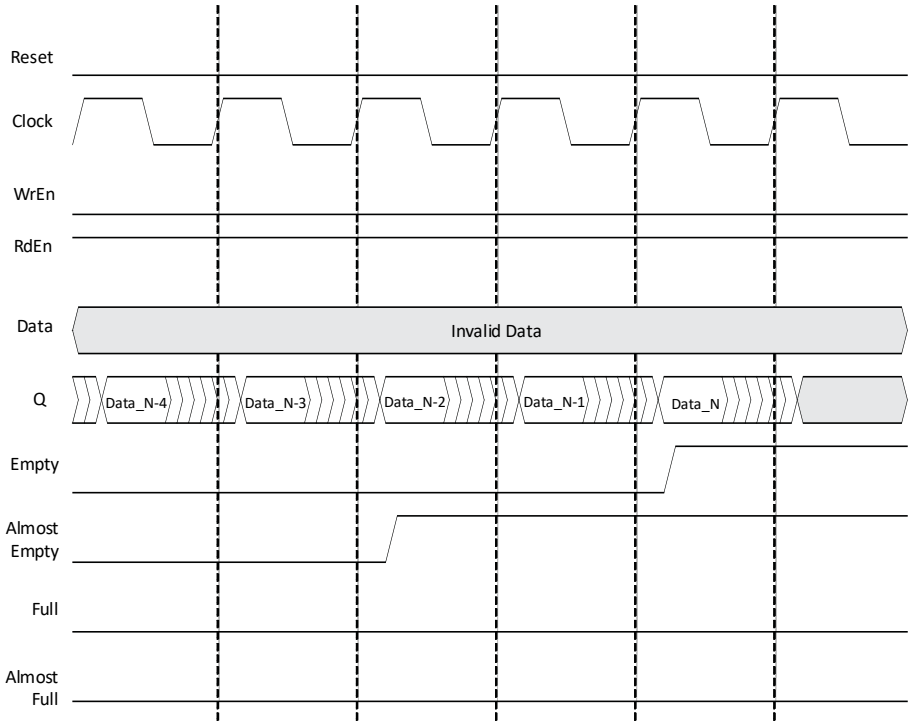

Figure 5.4. FIFO Without Output Registers, End of Data Read Cycle

Figure 5.1 to Figure 5.4 show the behavior of non-pipelined FIFO or FIFO without output registers. When the registers are pipelined, the output data is delayed by one clock cycle. There is also an option for output registers to be enabled by the RdEn signal.

Figure 5.5 to Figure 5.8 show similar waveforms for the FIFO with an output register and an output register enable with RdEn. Note that flags are asserted and de-asserted with similar timing to the FIFO without output registers. Only the data out Q gets delayed by one clock cycle.

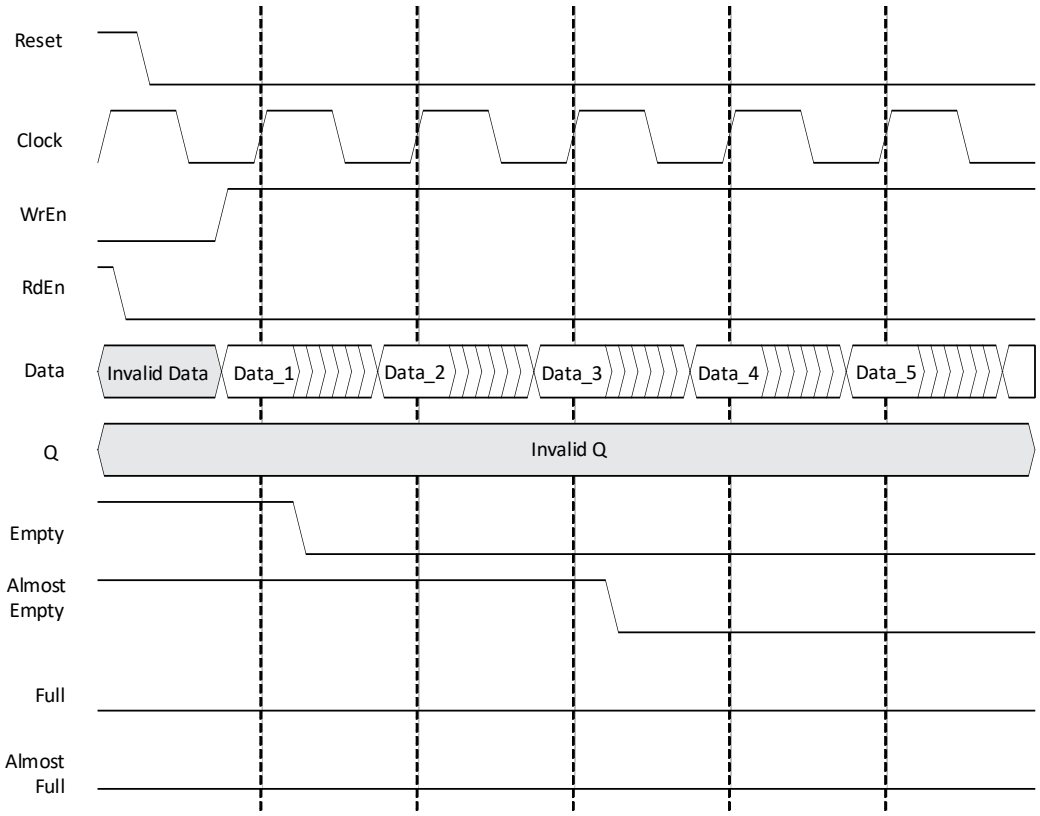

Figure 5.5. FIFO with Output Registers, Start of Data Write Cycle

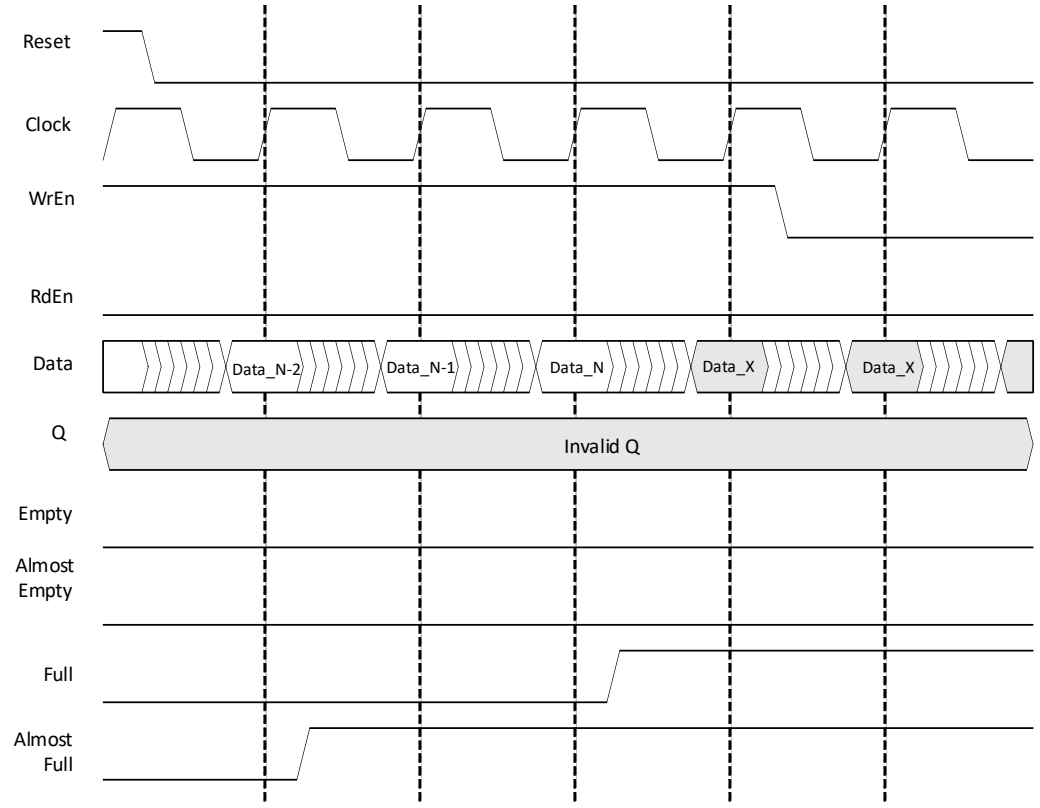

Figure 5.6. FIFO with Output Registers, End of Data Write Cycle

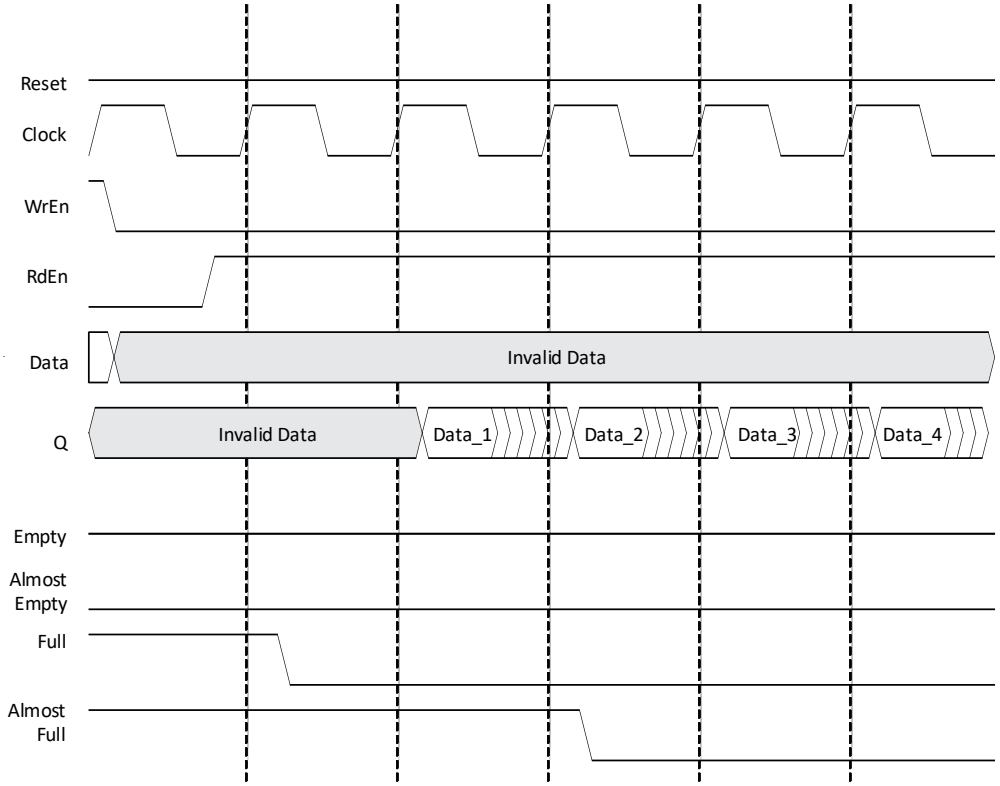

Figure 5.7. FIFO with Output Registers, Start of Data Read Cycle

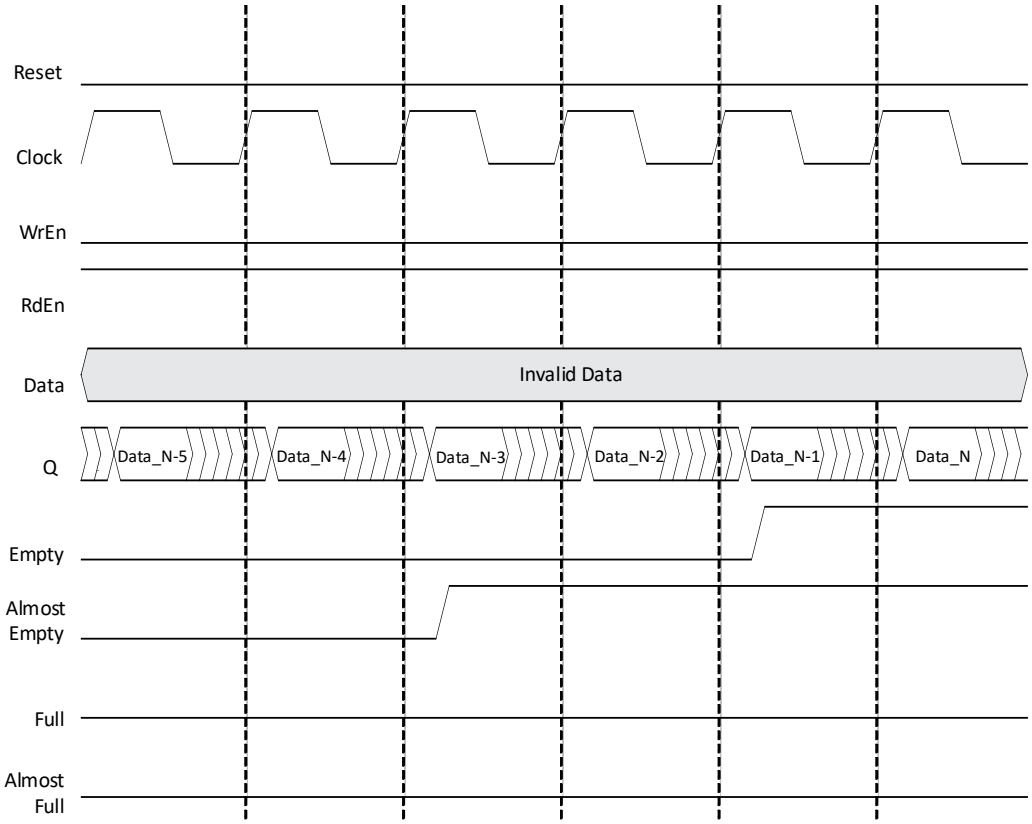

Figure 5.8. FIFO with Output Registers, End of Data Read Cycle

If the option enable output register with RdEn is selected, 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.

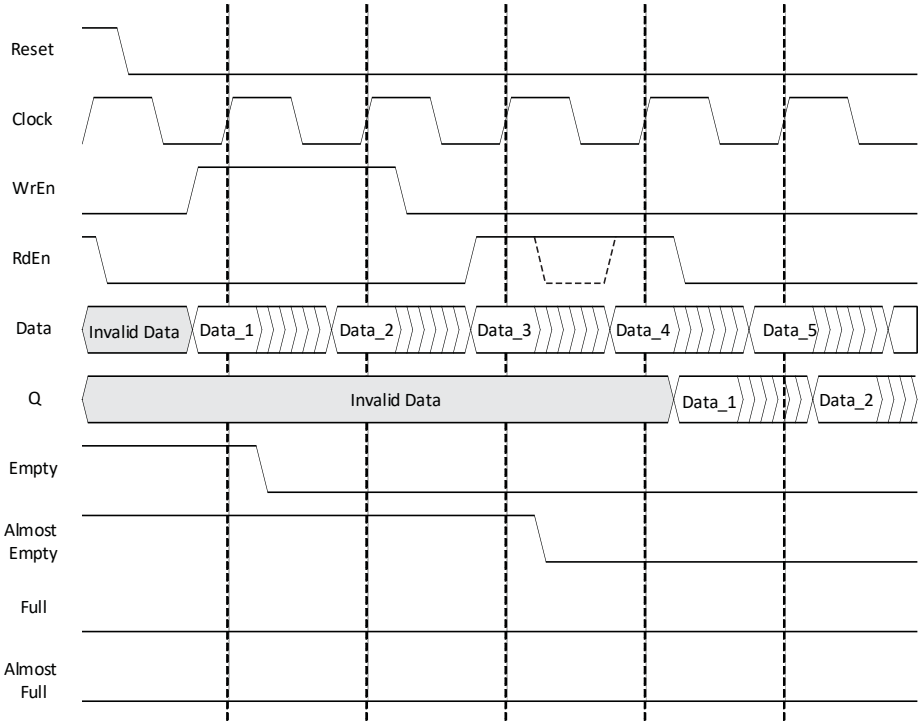

Figure 5.9. FIFO with Output Registers and RdEn on Output Registers

6. Dual-Clock First-In-First-Out (FIFO_DC) Memory

The FIFO_DC, or dual-clock FIFO, is also an emulated FIFO. The address logic and flag logic are implemented in the FPGA fabric around the RAM.

The ports available on the FIFO_DC include:

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

6.1. FIFO_DC Flags

As an emulated FIFO, FIFO_DC requires 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 de-assertion. Latency can be avoided only in one of the cases, either assertion or de-assertion, or distributed among the two.

In the emulated FIFO_DC, there is no latency during assertion of the flags. Thus, when the flag goes true, there is no latency. However, due to the design of the flag logic running on two clock domains, there is latency during deassertion.

The FULL and ALMOST_FULL flags operating in the write-clock domain wait for three cycles before de-assertion because of a read in the read-clock domain. However, no latency is required for assertion as a write operation operates in the same write-clock domain. Additionally, this implementation is the same for the EMPTY and ALMOST_EMPTY flags which operate in the read-clock domain. There is a three-cycle delay before de-assertion, but no additional latency for the flag assertion as the read operates in the clock domain.

Assume that you 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 the de-assertion of the Empty and Almost Empty flags.

De-assertion of Full and Almost Full although result of reading out the data from the FIFO_DC, takes extra WrClock cycles after reading the data for these flags to come out.

With this in mind, look at the waveforms for FIFO_DC without output registers. Figure 6.1 shows the operation of the FIFO_DC when it is empty and the data begins to be written into it.

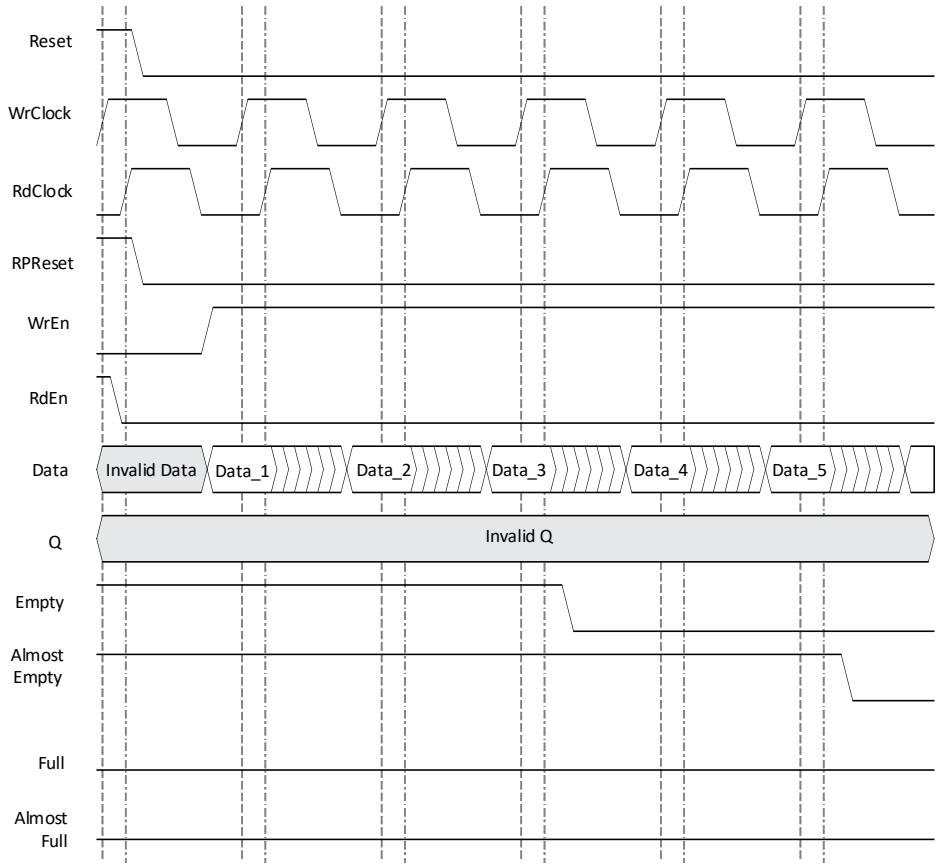

Figure 6.1. FIFO_DC Without Output Registers, Start of Data Write Cycle

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 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, it is assumed that the Almost Empty setting flag setting is 3, address location 3. The Almost Empty flag is de-asserted when the third address location is filled.

Now assume that you 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 6.2 shows the behavior of these flags. In this figure, it is assumed that FIFO_DC depth is N.

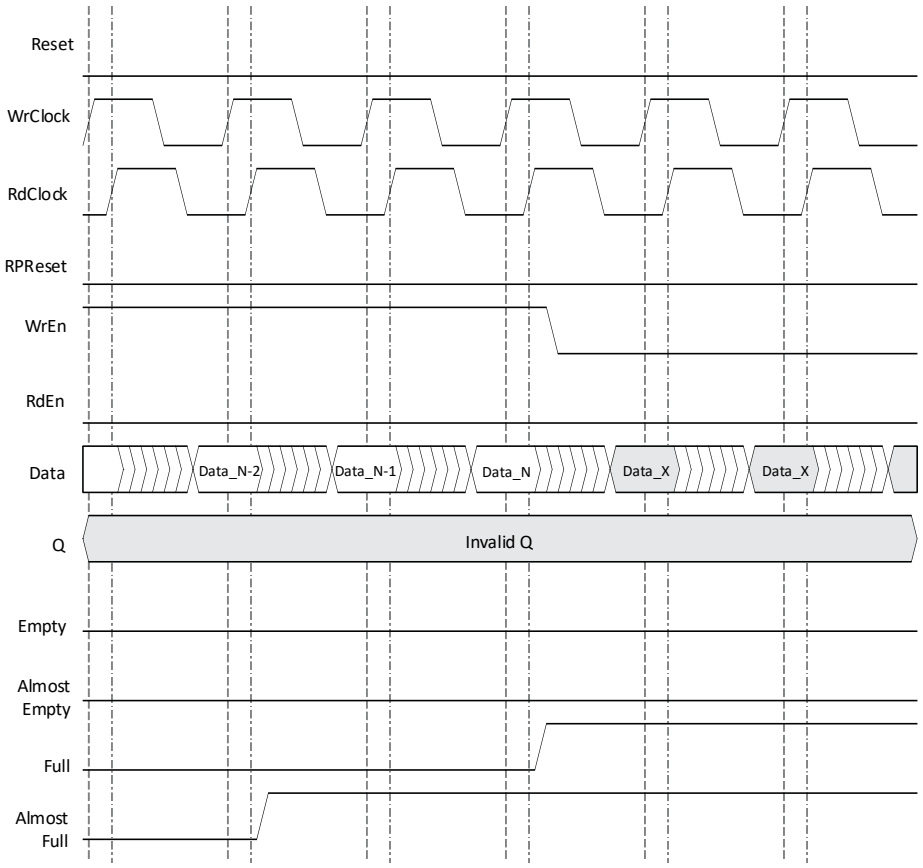

Figure 6.2. FIFO_DC Without Output Registers, End of Data Write Cycle

In Figure 6.2, the Almost Full flag is two locations 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 are not written since the FIFO_DC is full, that is, the Full flag is high.

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

Now look at the waveforms when the contents of the FIFO_DC are read out. Figure 6.3 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. Note that the de-assertion is delayed by two clock cycles.

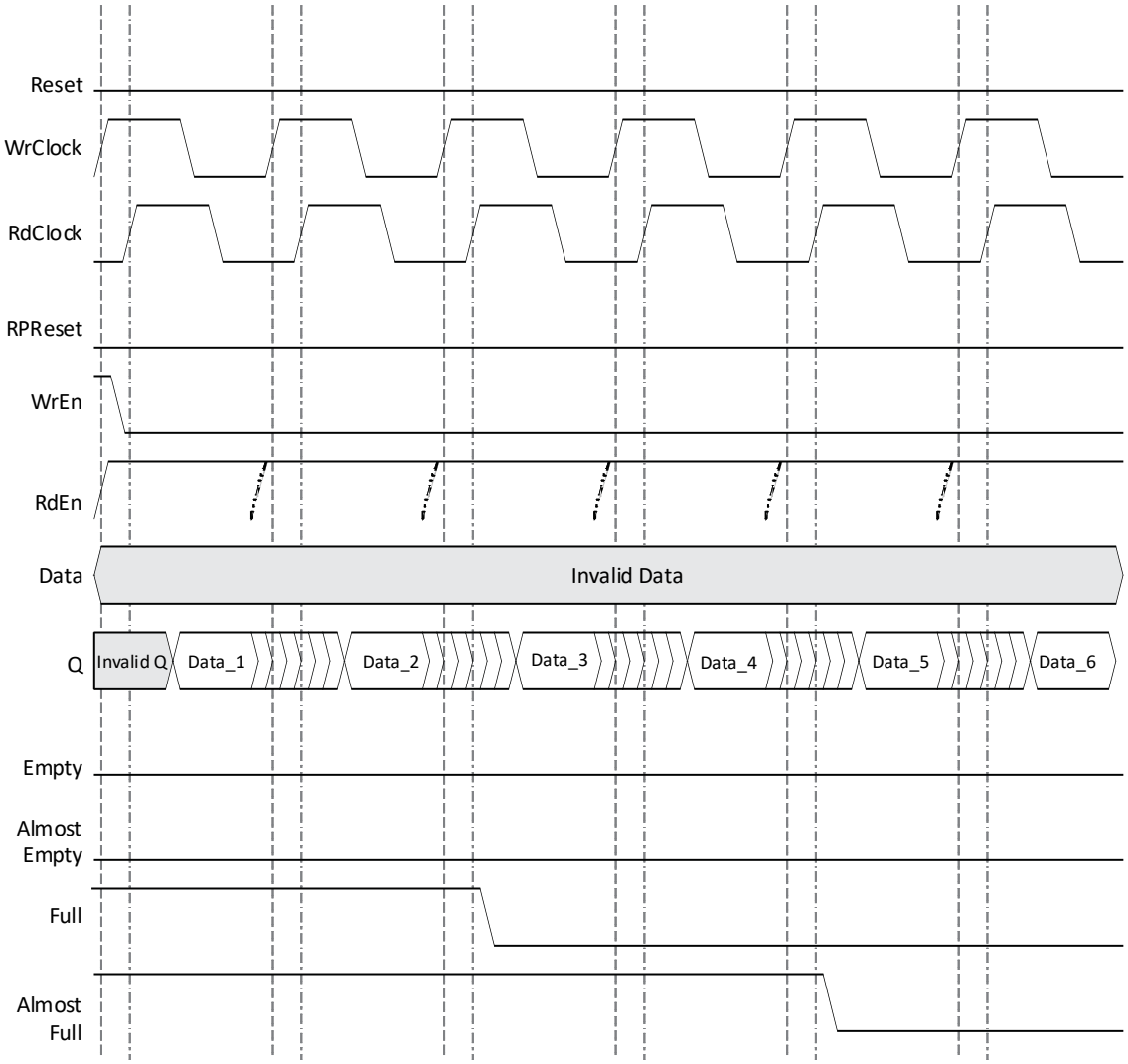

Figure 6.3. FIFO_DC Without Output Registers, Start of Data Read Cycle

Similarly, as the data is read out and FIFO_DC is emptied, the Almost Empty and Empty flags are asserted (see Figure 6.4).

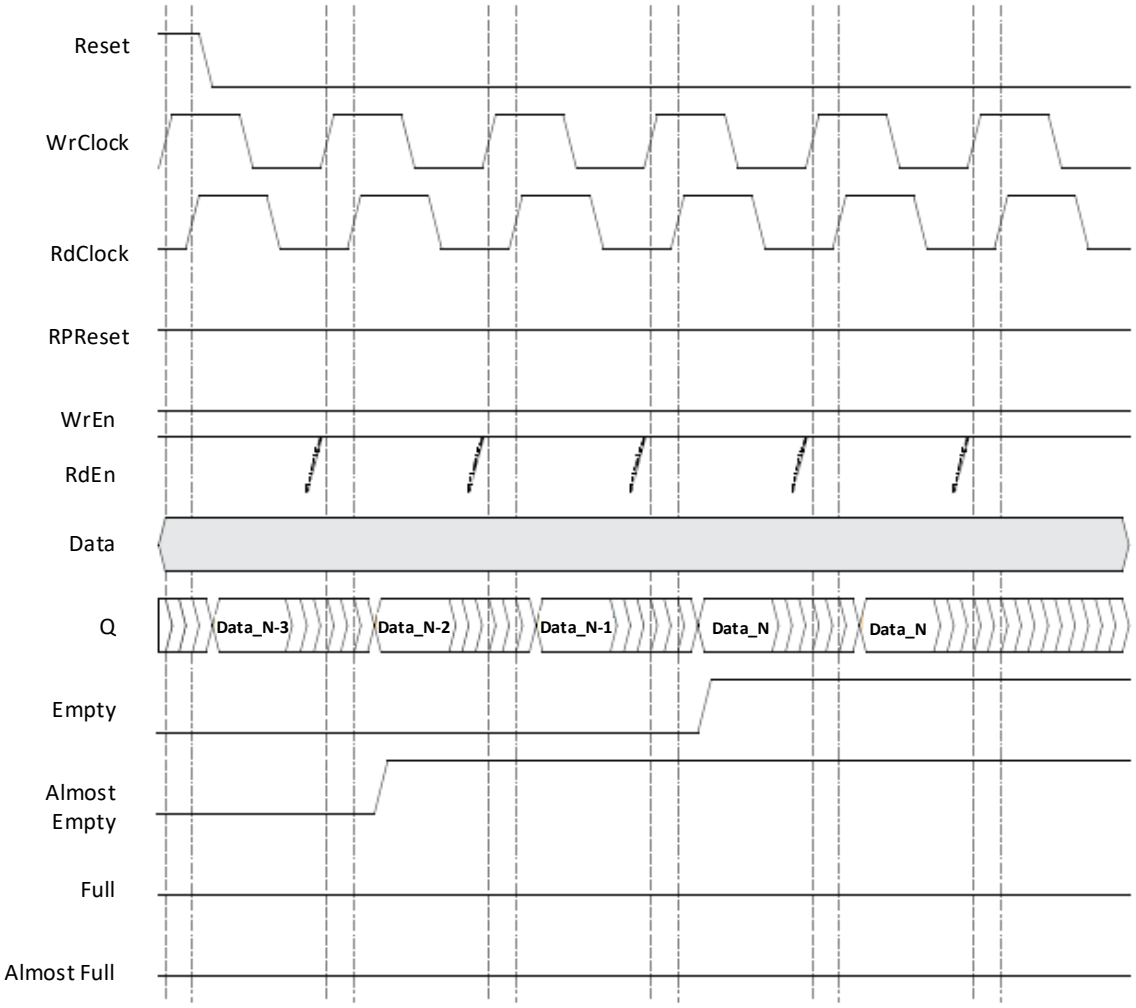

Figure 6.4. FIFO_DC Without Output Registers, End of Data Read Cycle

Figure 6.1 to Figure 6.4 show the behavior of the non-pipelined FIFO_DC or FIFO_DC without output registers. When you 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.

Figure 6.5 to Figure 6.8 show 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 timing similar to the FIFO_DC without output registers. However, only the data out Q is delayed by one clock cycle.

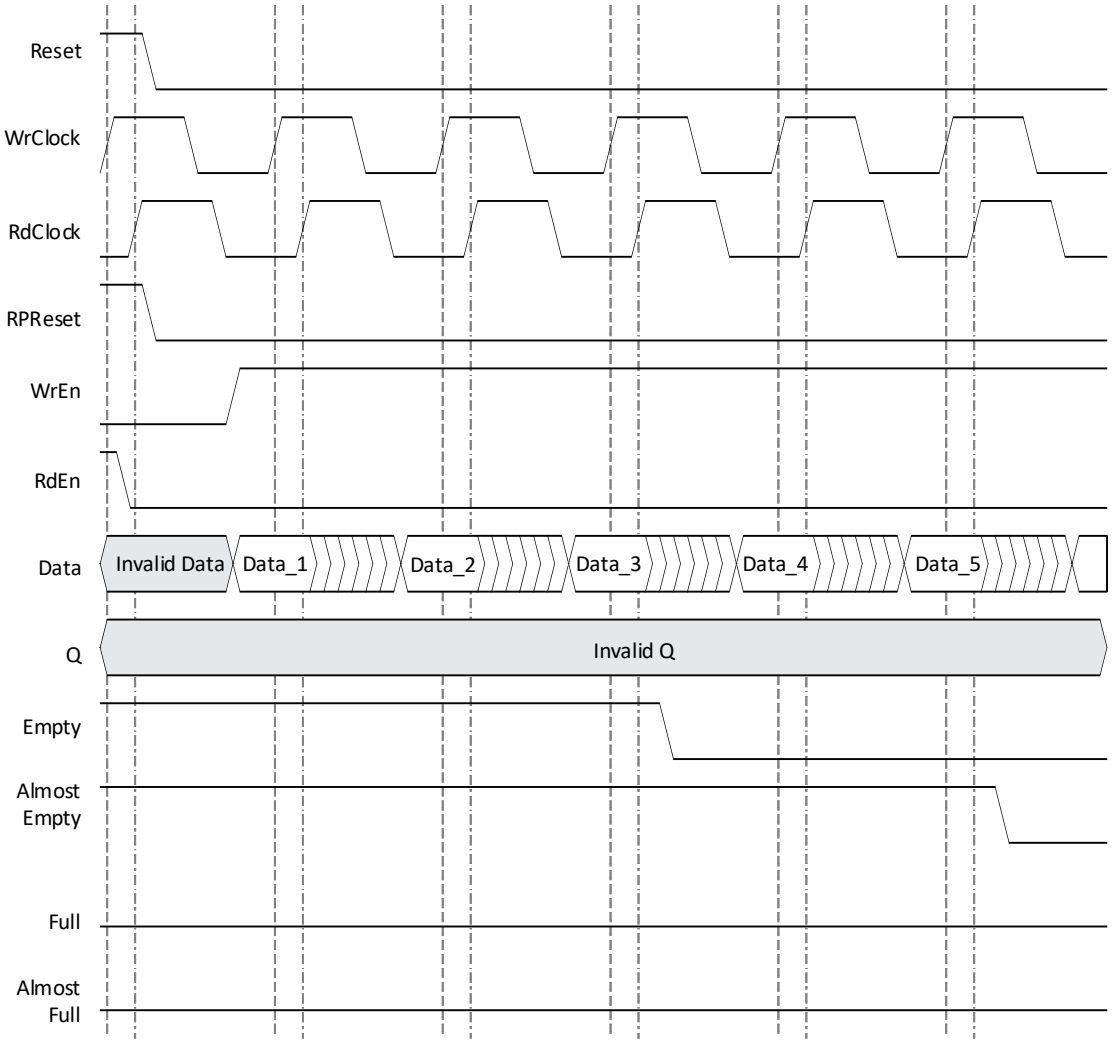

Figure 6.5. FIFO_DC with Output Registers, Start of Data Write Cycle

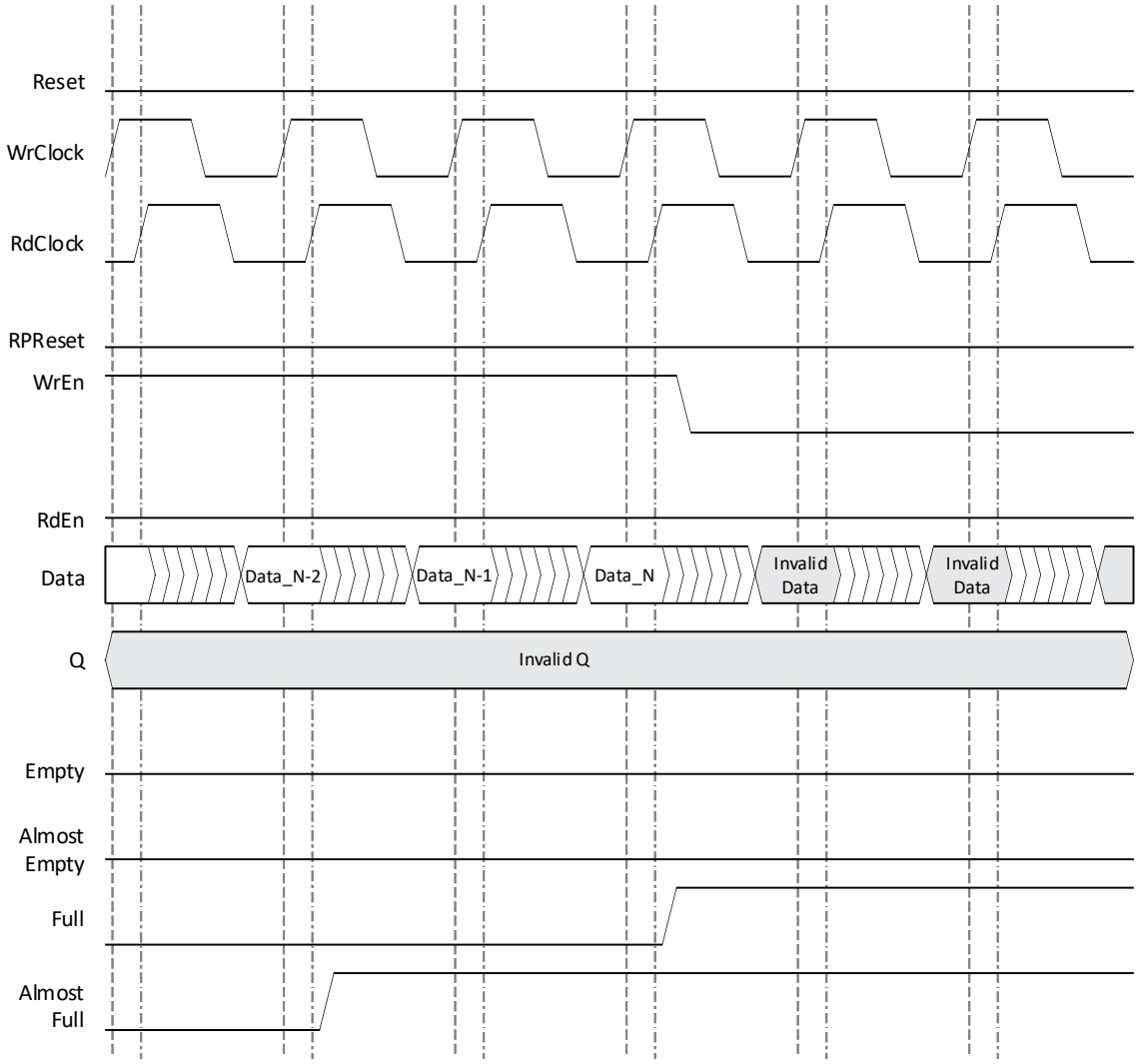

Figure 6.6. FIFO_DC with Output Registers, End of Data Write Cycle

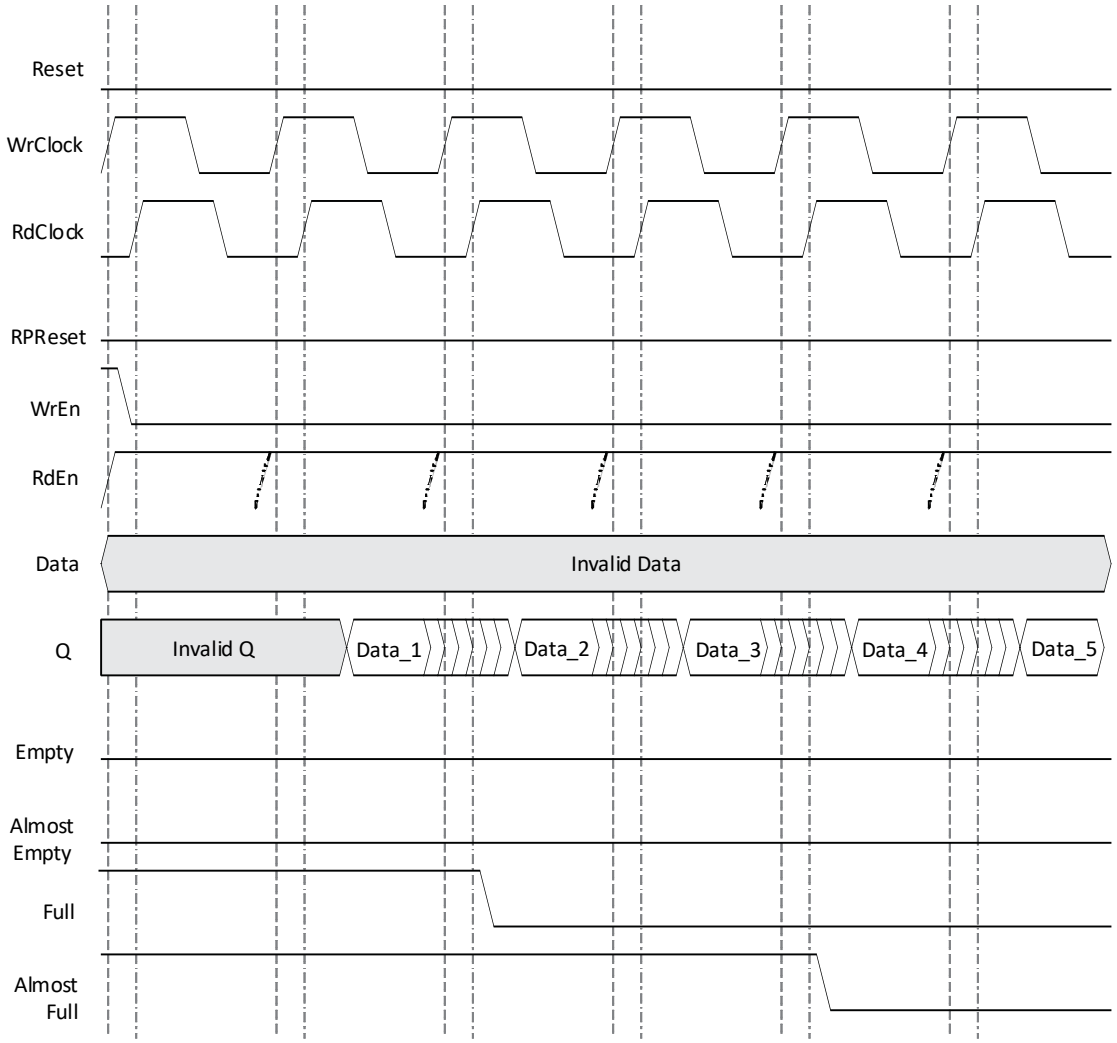

Figure 6.7. FIFO_DC with Output Registers, Start of Data Read Cycle

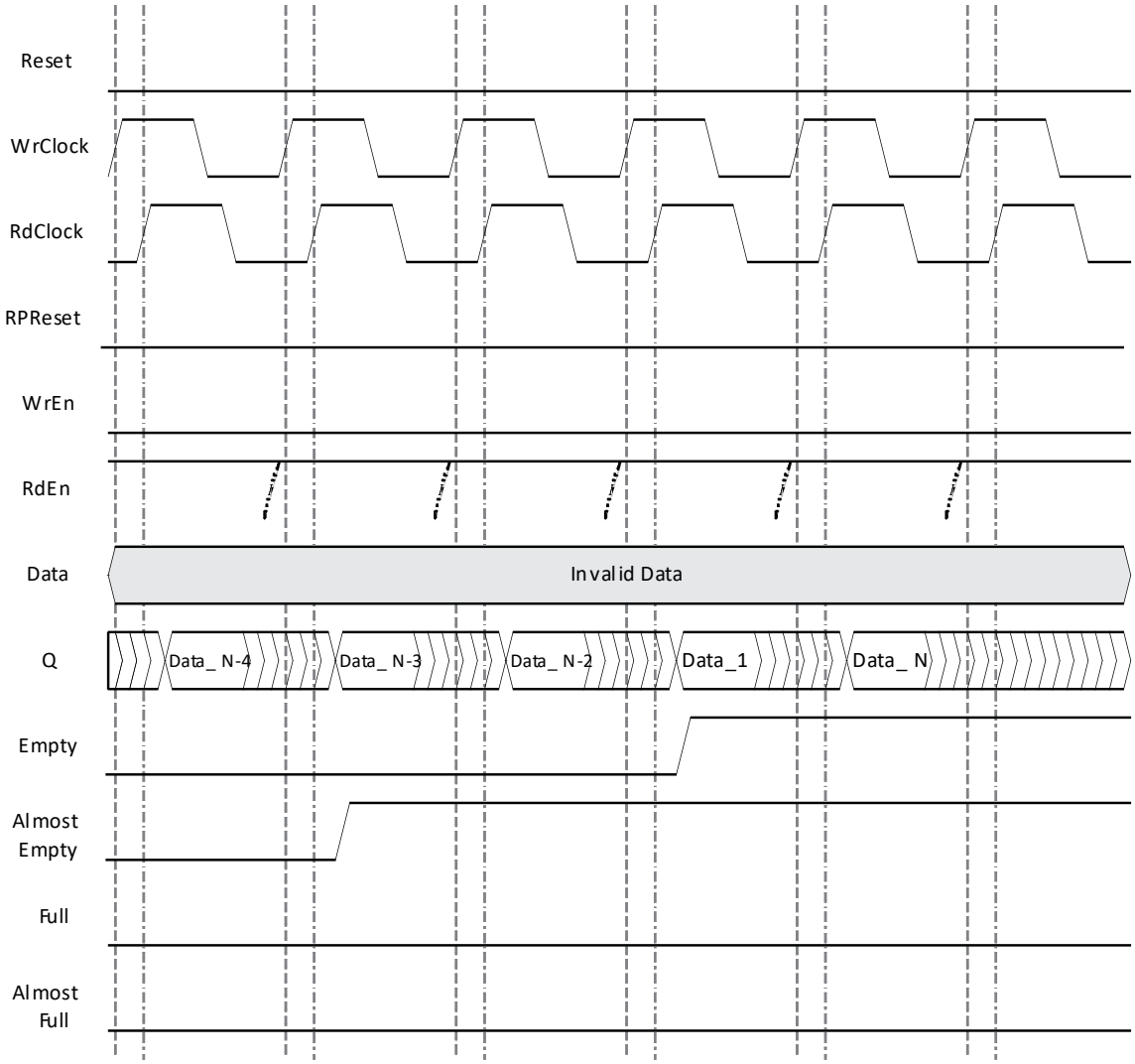

Figure 6.8. FIFO_DC with Output Registers, End of Data Read Cycle

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

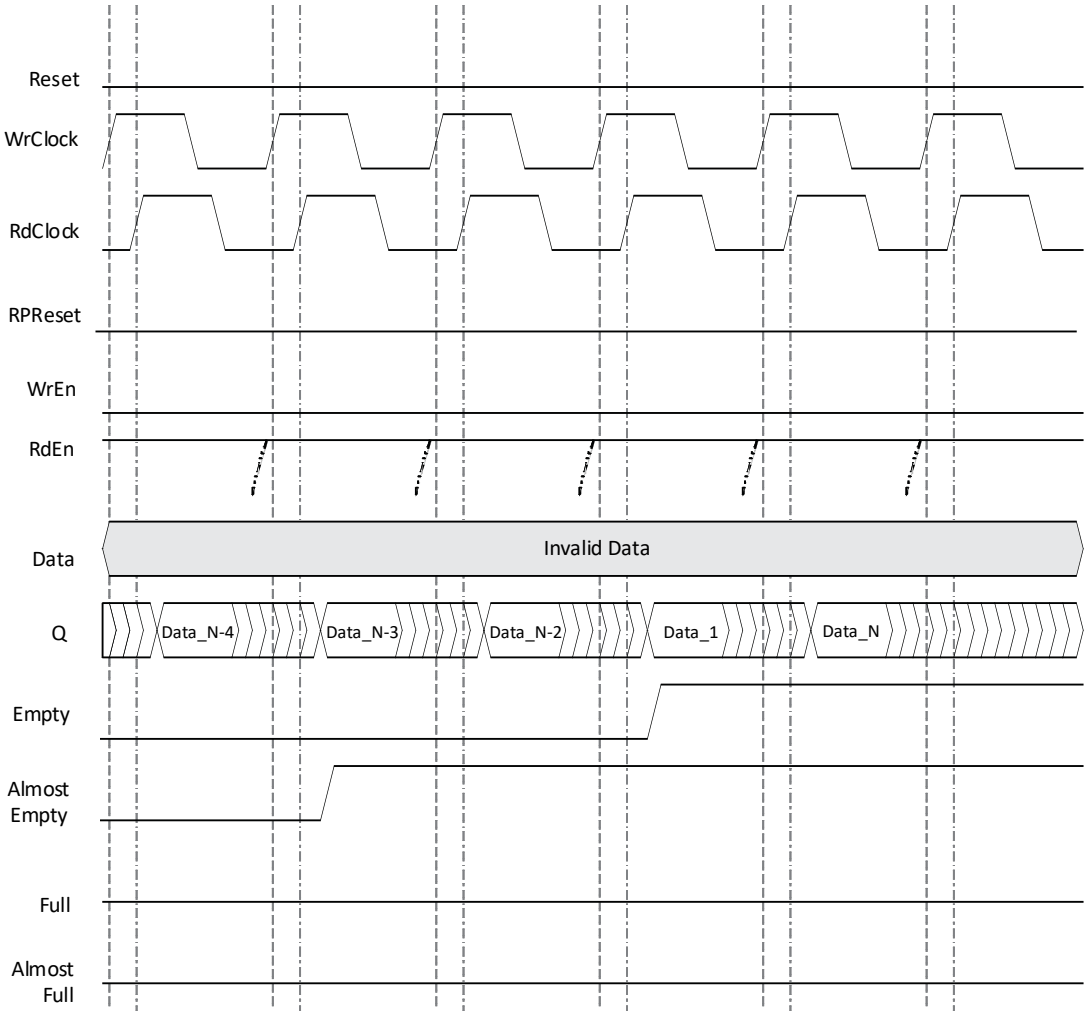

Figure 6.9. FIFO_DC with Output Registers and RdEn on Output Registers

7. Distributed Single-Port RAM (Distributed_SPRAM) – PFU-Based

PFU-based Distributed Single-Port RAM is created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 7.1 shows the Distributed Single-Port RAM module as generated by IPexpress.

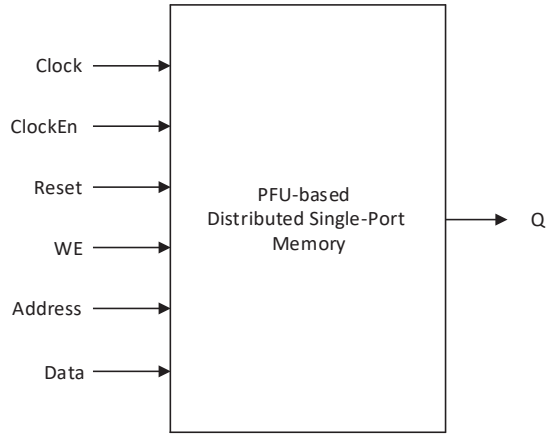

Figure 7.1. Distributed Single-Port RAM Module Generated by IPexpress

The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or 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 you want to enable the output registers in the IPexpress configuration.

The various ports and their definitions are listed in Table 7.1. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 7.1. PFU-Based Distributed Single-Port RAM Port Definitions

Port Name in Generated Module	Port Name in the EBR block Primitive	Description
Clock	CK	Clock
ClockEn	—	Clock Enable
Reset	—	Reset
WE	WRE	Write Enable
Address	AD[3:0]	Address
Data	DI[1:0]	Data In
Q	DO[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 their IPexpress configuration.

The various ports and their definitions for the memory are included in Table 4.11. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

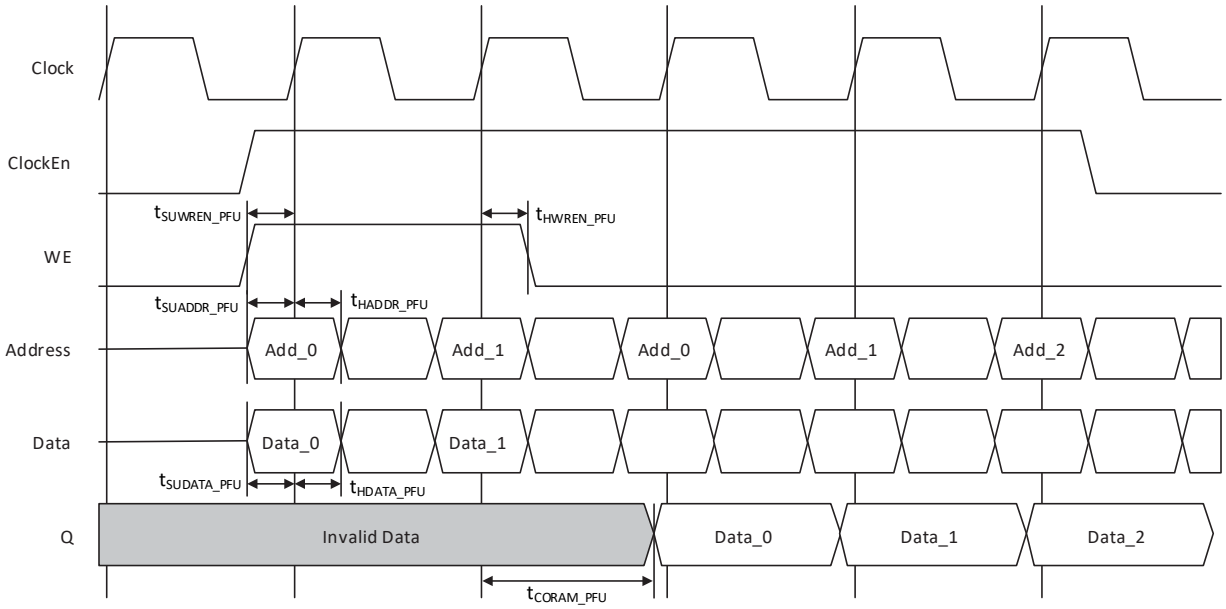

Figure 7.2. PFU-Based Distributed Single-Port RAM Timing Waveform – without Output Registers

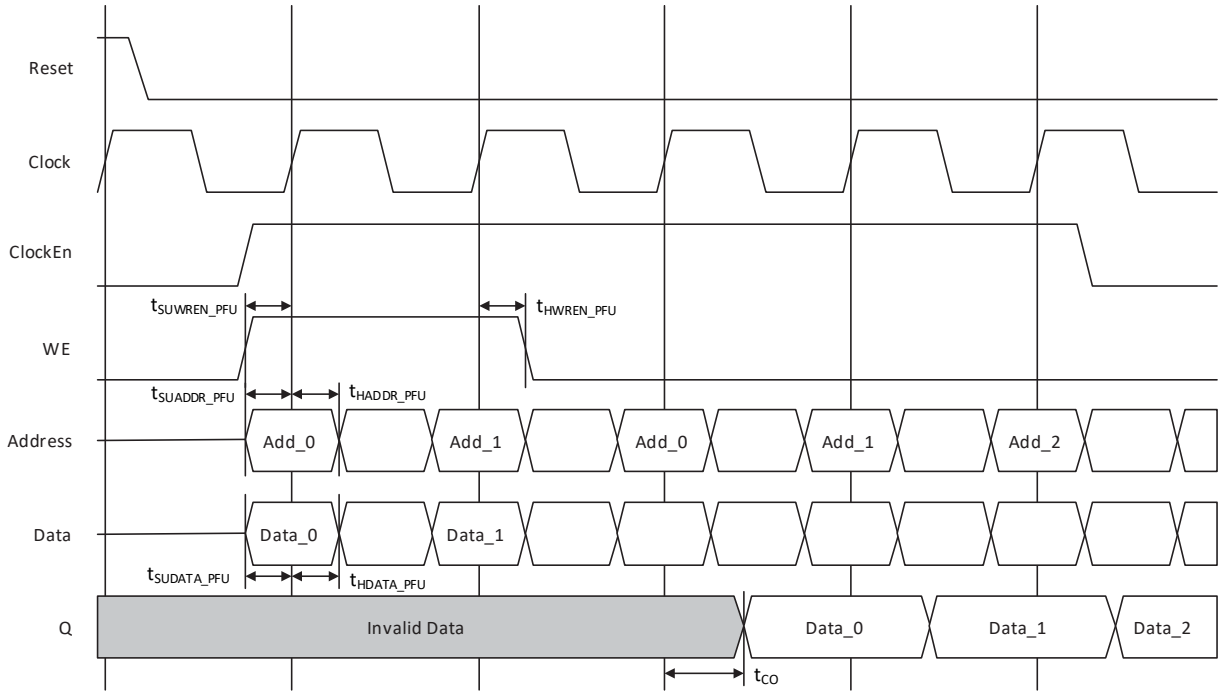

Figure 7.3. PFU-Based Distributed Single-Port RAM Timing Waveform – with Output Registers

8. Distributed Dual-Port RAM (Distributed_DPRAM) – PFU-Based

PFU-based Distributed Dual-Port RAM is also created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 8.1 shows the Distributed Single-Port RAM module as generated by IPexpress.

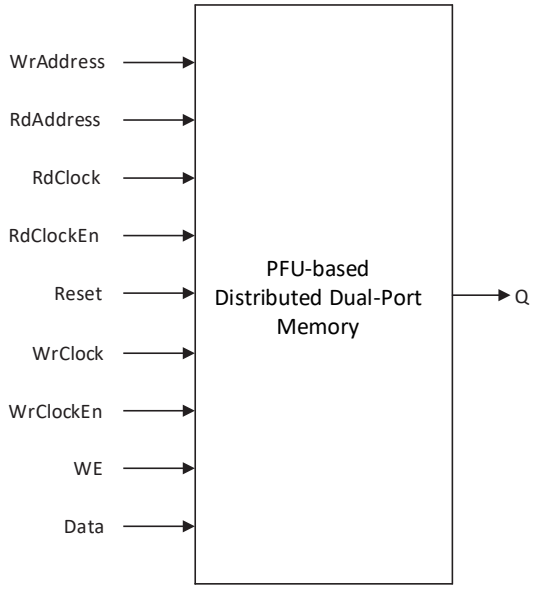

Figure 8.1. Distributed Dual-Port RAM Module Generated by IPexpress

The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or reset is generated by utilizing the resources available in the PFU.

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

The various ports and their definitions are listed in Table 8.1. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 8.1. PFU-Based Distributed Dual-Port RAM Port Definitions

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
WrAddress	WAD[23:0]	Write Address
RdAddress	RAD[3:0]	Read Address
RdClock	—	Read Clock
RdClockEn	—	Read Clock Enable
WrClock	WCK	Write Clock
WrClockEn	—	Write Clock Enable
WE	WRE	Write Enable
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 you want to enable the output registers in the IPexpress configuration.

You have the option of enabling the output registers for Distributed Dual-Port RAM (Distributed_DPRAM). Figure 8.2 and Figure 8.3 show the internal timing waveforms for the Distributed Dual-Port RAM (Distributed_DPRAM) with these options.

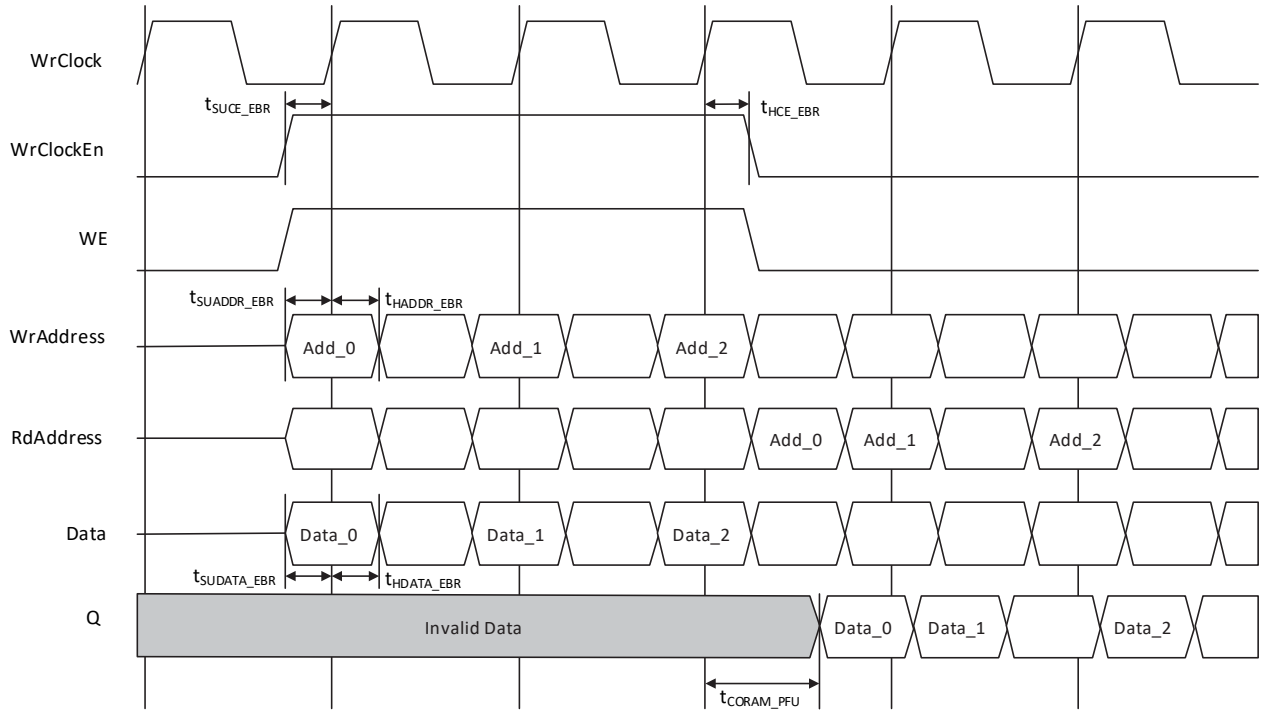

Figure 8.2. PFU-Based Distributed Dual-Port RAM Timing Waveform – without Output Registers

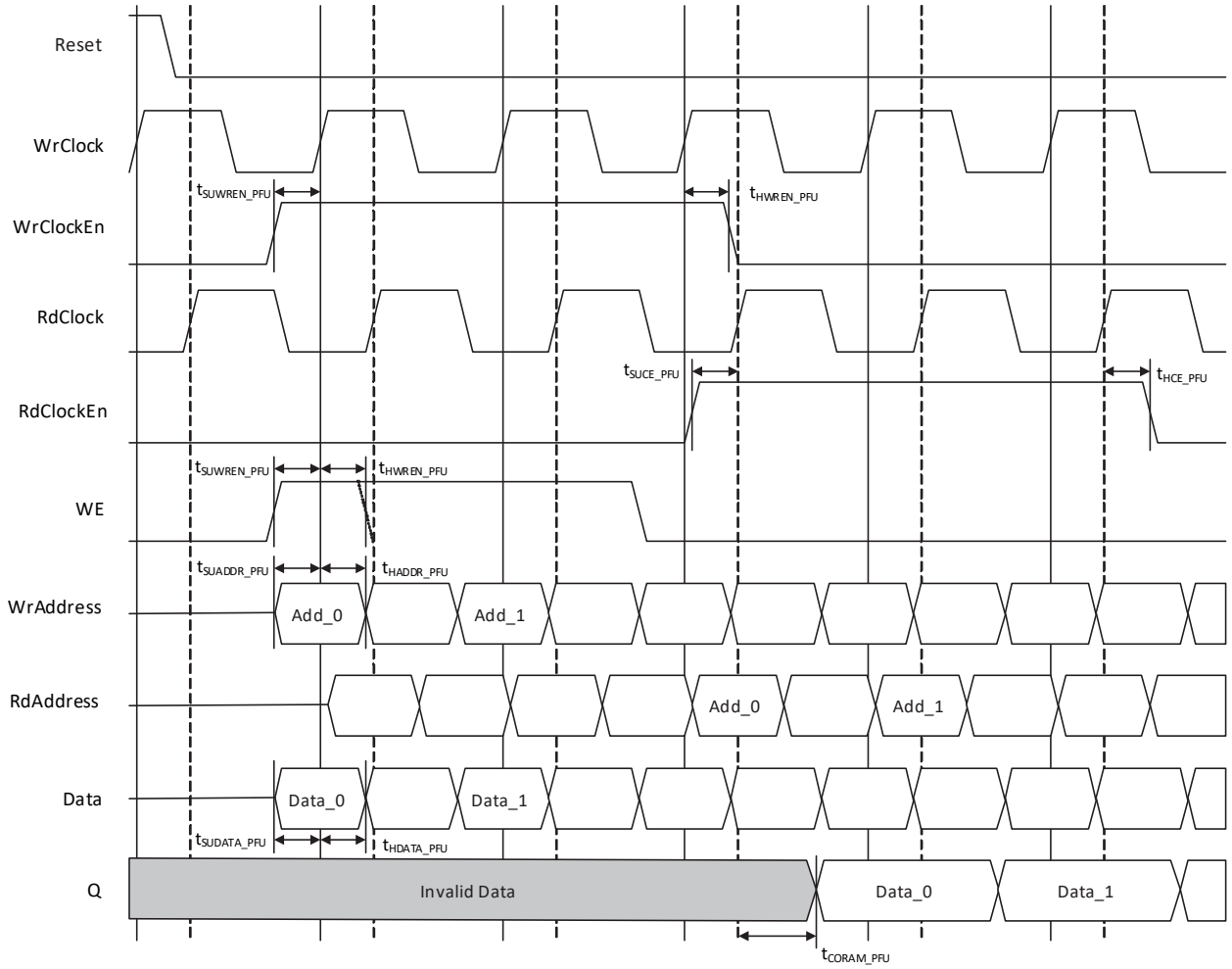

Figure 8.3. PFU-Based Distributed Dual-Port RAM Timing Waveform – with Output Registers

9. Distributed ROM (Distributed_ROM) – PFU-Based

PFU-based Distributed ROM is also created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 9.1 shows the Distributed ROM module generated by IPexpress.

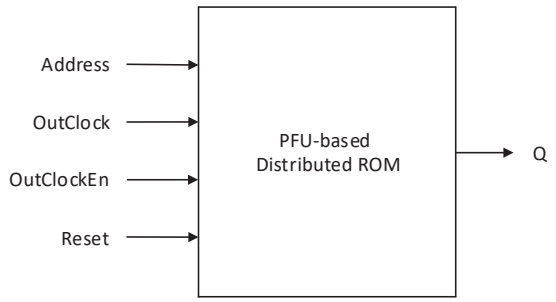

Figure 9.1. Distributed ROM Generated by IPexpress

The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or 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 you want to enable the output registers in the IPexpress configuration.

The various ports and their definitions are listed in Table 9.1. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 9.1. PFU-Based Distributed ROM Port Definitions

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
Address	AD[3:0]	Address
OutClock	—	Out Clock
OutClockEn	—	Out Clock Enable
Reset	—	Reset
Q	DO	Data Out

You have the option to enable the output registers for Distributed ROM (Distributed_ROM). Figure 9.2 and Figure 9.3 show the internal timing waveforms for the Distributed ROM with these options.

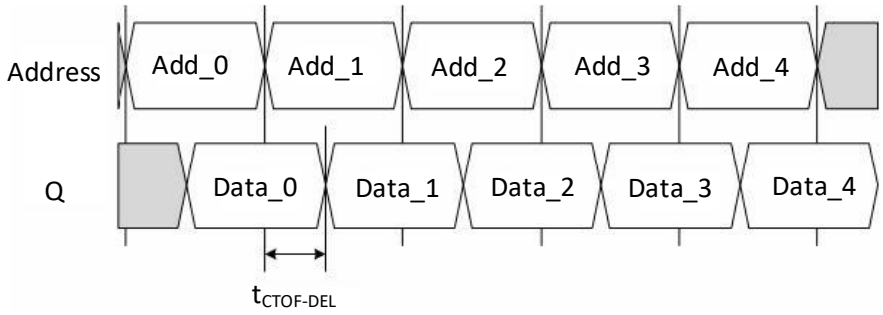

Figure 9.2. PFU-Based Distributed ROM Timing Waveform – without Output Registers

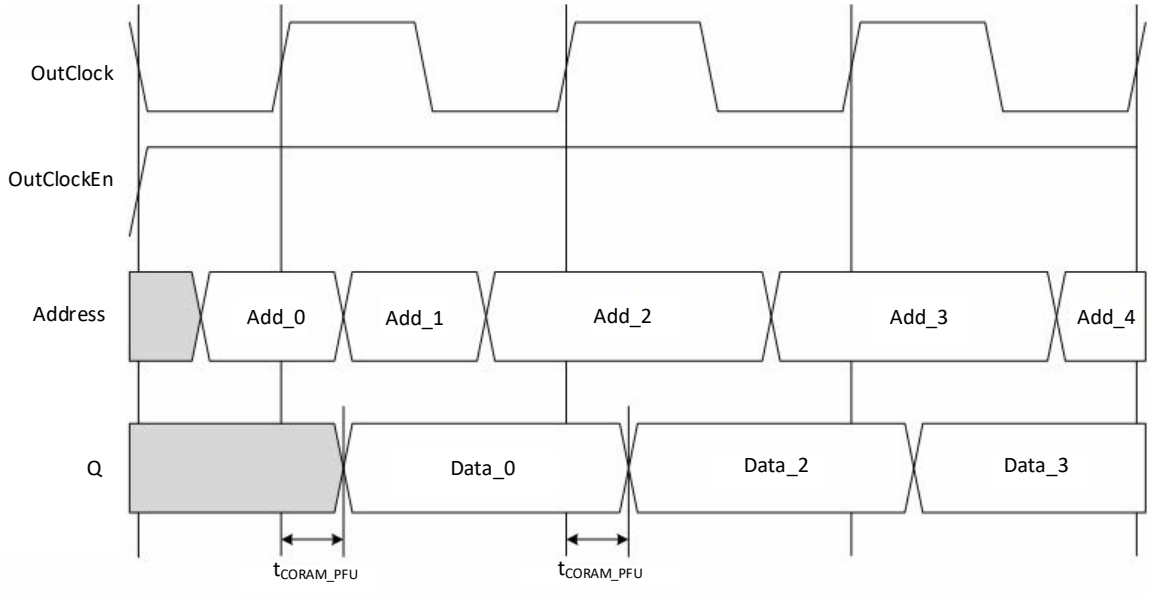

Figure 9.3. PFU-Based Distributed ROM Timing Waveform – with Output Registers

10. Initializing Memory

In each memory mode, it is possible to specify the power-on state of each bit in the memory array. This allows the memory to be used as ROM if desired. Each bit in the memory array can have a value of 0 or 1.

10.1. Initialization File Formats

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

- Binary file
- Hex File
- Addressed Hex

The file name for the memory initialization file is *.mem (<file_name>.mem). Each row includes 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, respectively.

The initialization file is primarily used for configuring the ROMs. RAMs can also use the initialization file to preload memory contents.

Binary File

The binary file is a text file of 0s and 1s. The rows indicate the number of words and the 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
00010010000010010000100100010010
00010011000100110001001100010011

Hex File

The hex file is a text file of hexadecimal characters arranged in a similar row-column arrangement. The number of rows in the file is the 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 addresses and data. Each line starts with an address, followed by a colon, and any number of data. The format of the file is “address: data data data data” where the address and data are hexadecimal numbers.

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

In the example above, the first line shows 03 at address A0, F3 at address A1, 3E at address A2,and 4F at address A3. The second line shows 3B at address B2 and 9F at address B3.

There is no limitation on the address and data values. 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. It is not necessary to specify data at all address locations. If data is not specified at a certain address, the data at that location is initialized to 0. SCUBA makes memory initialization possible both through the synthesis and simulation flows.

Appendix A. Attribute Definitions

DATA_WIDTH

Data width is associated with the RAM and FIFO elements. The DATA_WIDTH attribute defines the number of bits in each word. It uses the values 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.

CSDECODE

CSDECODE or the Chip Select Decode attributes are associated with block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that 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.

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.

WRITEMODE_A and WRITEMODE_B are used for Dual-Port RAM elements and refer to the A and B ports in True Dual-Port RAM.

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.

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 an H and the other tries to write an L. It is recommended that control logic be implemented to identify this situation if it occurs and do one of the following:

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

References

- LatticeECP3 Family Devices web page
- Boards, Demos, IP Cores, and Reference Designs for LatticeECP3 Family Devices web page
- Lattice Diamond Software web page
- Lattice Insights for Lattice Semiconductor Training Series and Learning Plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 2.0, June 2025

Section	Change Summary
All	• Replaced ispLever with the Lattice Diamond software globally. • Updated the document title from *LatticeECP3 Memory Usage Guide* to *LatticeECP3 Memory User Guide*.
Utilizing IPexpress	• Updated Figure 2.1. IPexpress Main Window, Figure 2.2. Example: Generating Pseudo Dual-Port RAM (RAM_DP) Using IPexpress, and Figure 2.3. Example: Generating Pseudo Dual-Port RAM (RAM_DP) Module Customization – General Options. • Updated the following description to match IPexpress GUI shown in Figure 2.2. Example: Generating Pseudo Dual-Port RAM (RAM_DP) Using IPexpress: – changed *in the right pane, options like Device Family, Macro Type, Category, and Module_Name are device- and selected-module-dependent* to *in the right pane, options including Macro Type, Version, Module Name, Device Family, Part Name, and Synthesis are device and selected module dependent*; – changed *the entity name for the module to be generated can be specified in the Module Name text box. Users must provide this entity name* to *the entity name for the module to be generated can be specified in the File Name text box. You must provide this entity name*; – removed *for Design Entry, whether Verilog or VHDL, is the same as the project type by default. If, the selected Design Entry option should be Schematic/VHDL if the project is a VHDL project, and Schematic/Verilog-HDL if the project type is Verilog- HDL*; – removed *the Device drop-down menu allows you to select different devices within the same family, LatticeECP3 in this example*; – added f*or Module Output, select either Verilog or VHDL*. • Removed the Load Parameters related description below to match the IPexpress GUI shown in Figure 2.3. Example: Generating Pseudo Dual-Port RAM (RAM_DP) Module Customization – General Options: *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.*
Dual-Clock First-In-First-Out (FIFO-DC) Memory	Added the following description in the FIFO_DC Flags section: *The FULL and ALMOST_FULL flags operating in the write-clock domain wait for three cycles before de-assertion because of a read in the read-clock domain. However, no latency is required for assertion as a write operation operates in the same write-clock domain. Additionally, this implementation is the same for the EMPTY and ALMOST_EMPTY flags which operate in the read-clock domain. There is a three-cycle delay before de-assertion, but no additional latency for the flag assertion as the read operates in the clock domain.*

Revision 1.9, March 2024

Section	Change Summary
All	• Changed document number from *TN1179* to *FPGA-TN-02188*. • Updated document template.
Disclaimers	Added this section.
Acronyms in this Document	Added this section.
References	Added this section.
Technical Support Assistance	Added the link to Lattice Answer Database.

Revision 1.8, April 2014

Section	Change Summary
Memory Modules	Removed Reset Mode attribute in the following tables:

Section	Change Summary
	• Table 4.3., Single-Port RAM Attributes of LatticeECP3 Devices • Table 4.6., True Dual-Port RAM Attributes for LatticeECP3 Devices • Table 4.9., Pseudo Dual-Port RAM Attributes for LatticeECP3 Devices Updated the following figures: • Figure 4.16., PSEUDO DUAL PORT RAM Timing Diagram – without Output Registers • Figure 4.17., PSEUDO DUAL PORT RAM Timing Diagram - with Output Registers
Distributed ROM (Distributed_ROM) – PFU-Based	Updated the following figures: • Figure 9.2., PFU Based Distributed ROM Timing Waveform – without Output Registers • Figure 9.3., PFU Based Distributed ROM Timing Waveform – with Output Registers
Technical Support Assistance	Updated Technical Support Assistance information.

Revision 1.7, February 2012

Section	Change Summary
All	Updated document with new corporate logo.

Revision 1.6, July 2011

Section	Change Summary
All	Added the setup and hold requirements for addresses to EBR-based memories.

Revision 1.5, January 2010

Section	Change Summary
All	Added Read Before Write mode for Single Port and True Dual Port RAM modules for LatticeECP3-xxEA devices.

Revision 1.4, November 2009

Section	Change Summary
Memory Modules	Updated Reset Mode based on device support.

Revision 1.3, June 2009

Section	Change Summary
All	• Text and screenshots were updated to match the latest ispLEVER software. • Added updated timing waveforms for EBR and Distributed RAMs. • Removed RAM-Based Shift Register.

Revision 1.2, June 2009

Section	Change Summary
All	Number of bits of RAM contained in each EBR block corrected to read 18,432.

Revision 1.1, May 2009

Section	Change Summary
All	References to 9K memories corrected to read 18K throughout the document.

Revision 1.0, February 2009

Section	Change Summary
All	Initial release.

www.latticesemi.com