LatticeXP2 Memory

Usage Guide

FPGA-UG-02080-2.3

March 2020

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 and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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
EBR	Embedded Block RAM
ECC	Error Correcting Code
PFF	Programmable Functional Unit (Without RAM)
PFU	Programmable Functional Unit
PMI	Parameterizable Module Instantiation
RAM	Random-Access Memory
ROM	Read-Only Memory
VHDL	VHSIC Hardware Description Language
EDIF	Electronic Design Interchange Format
FIFO	First In First Out
SPI	Serial Peripheral Interface

1. Introduction

This technical note discusses memory usage for the LatticeXP2™ device family. It is intended to be used by design engineers as a guide for integrating the User TAG, EBR- (Embedded Block RAM) and PFU-based memories in this device family using the ispLEVER® design tool.

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

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

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

2. Memories in LatticeXP2 Devices

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

The LatticeXP2 family of devices contains up to two rows of sysMEM EBR blocks. sysMEM EBRs are large, dedicated 18K fast memory blocks. Each sysMEM block can be configured in a variety of depths and widths of RAM or ROM. Each LatticeXP2 device also contains one dedicated row of User TAG memory with up to 451 bytes of space.

Table 2.1. LatticeXP2 LUT and Memories Densities

Parameter	XP2-5	XP2-8	XP2-17	XP2-30	XP2-40
EBR Rows	1	1	1	1	2
EBR Blocks	9	12	15	21	48
EBR Bits	165888	221184	276480	387072	884736
Distributed RAM Bits	10368	18432	34560	64512	82944
Total Memory Bits	176256	239616	311040	451584	967680

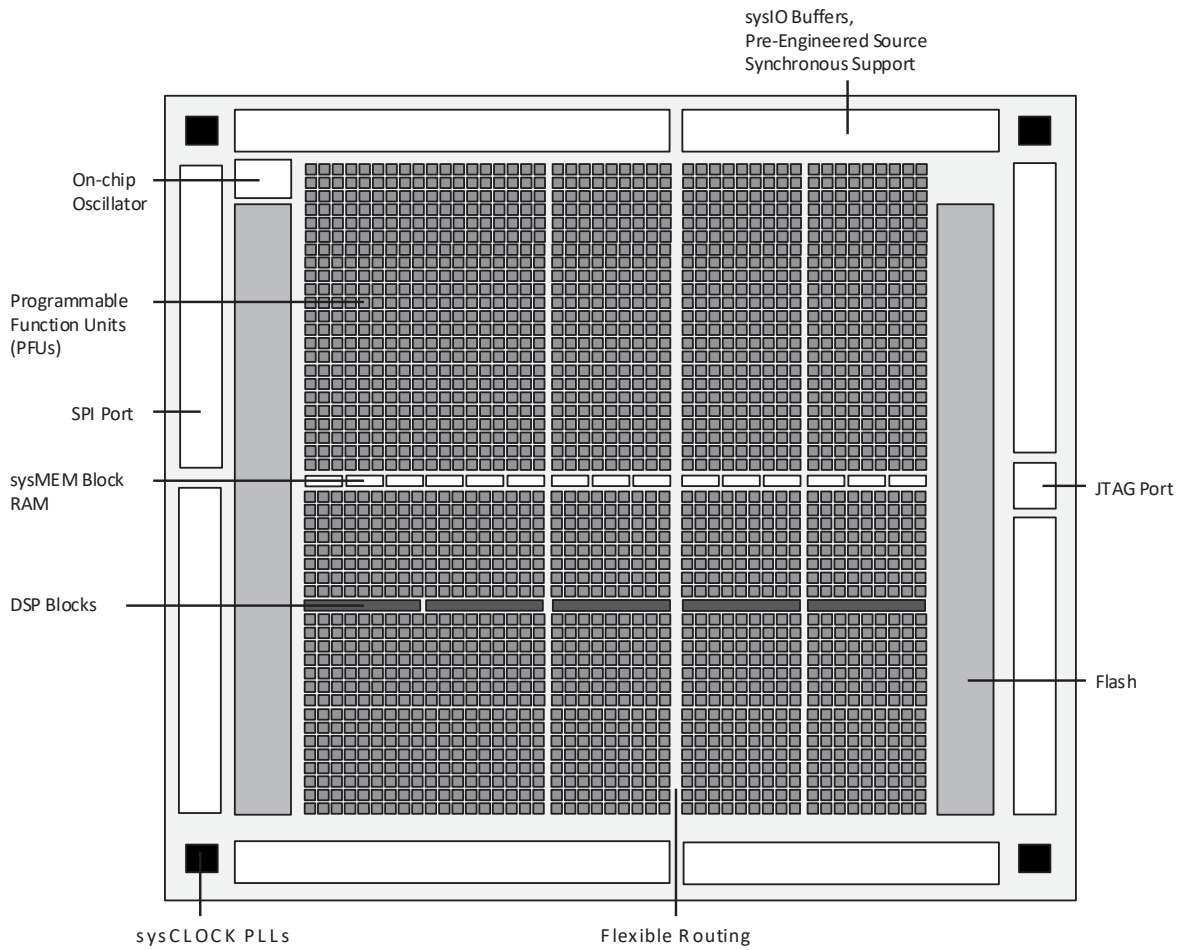

Figure 2.1. Simplified Block Diagram, LatticeXP2 Device (Top Level)

3. 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 LUTs, as required. The available primitives are:

- Single Port RAM (RAM_DQ) – EBR-based
- Dual PORT RAM (RAM_DP_TRUE) – EBR-based
- Pseudo Dual Port RAM (RAM_DP) – EBR-based
- Read Only Memory (ROM) – EBR-Based
- First In First Out Memory (Dual Clock) (FIFO_DC) – EBR-based
- Distributed Single Port RAM (Distributed_SPRAM) – PFU-based
- Distributed Dual Port RAM (Distributed_DPRAM) – PFU-based
- Distributed ROM (Distributed_ROM) – PFU/PFF-based
- User TAG memory (SSPIA) – TAG-based

3.1. IPexpress Flow

For generating any of these memories, create (or open) a project for the LatticeXP2 devices.

From the Project Navigator, select **Tools > IPexpress** or click on the button in the toolbar when LatticeXP2 devices are targeted in the project. This opens the IPexpress main window as shown in Figure 3.1.

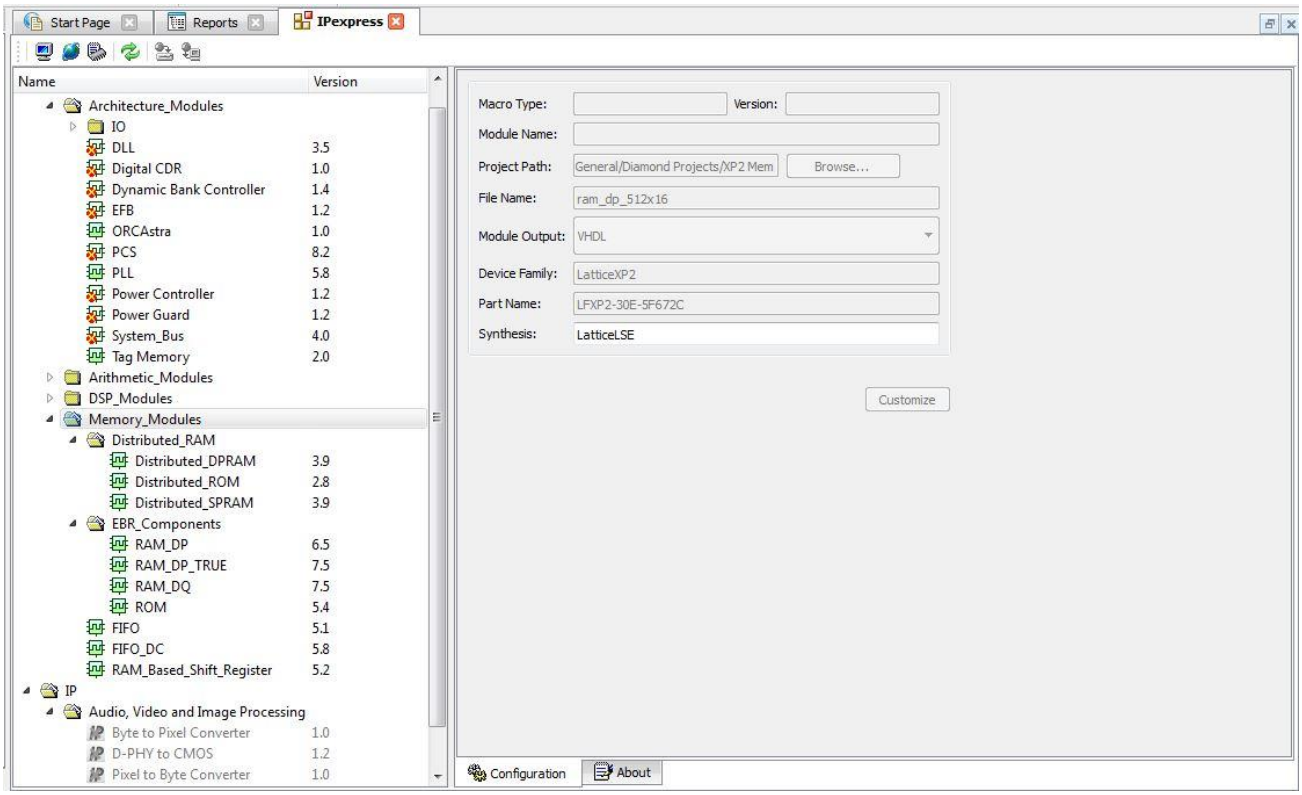

Figure 3.1. IPexpress – Main Window

The left pane of this window includes the Module Tree. The EBR-based Memory Modules are under the EBR_Components and the PFU-based Distributed Memory Modules are under Storage_Components, as shown in Figure 3.1.

As an example, let us consider generating an EBR-based Pseudo Dual Port RAM of size 512 × 16. Select RAM_DP under EBR_Components. The right pane changes as shown in Figure 3.2.

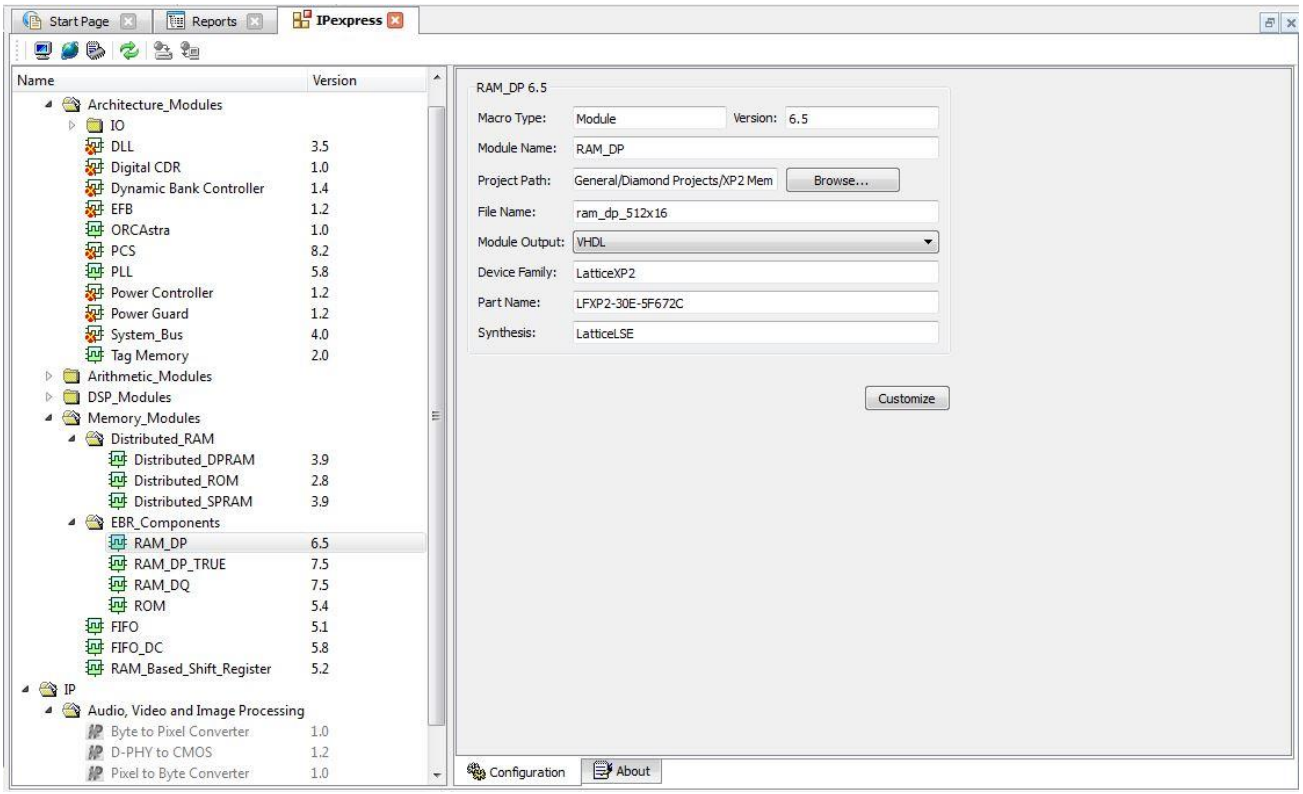

Figure 3.2. Example Generating Pseudo Dual Port RAM (RAM_DP) Using IPexpress

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

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

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

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

The **Device** pull-down menu allows you to select different devices within the same family, LatticeXP2 in this example. By clicking the **Customize** button, another window opens where you can customize the RAM (Figure 3.3).

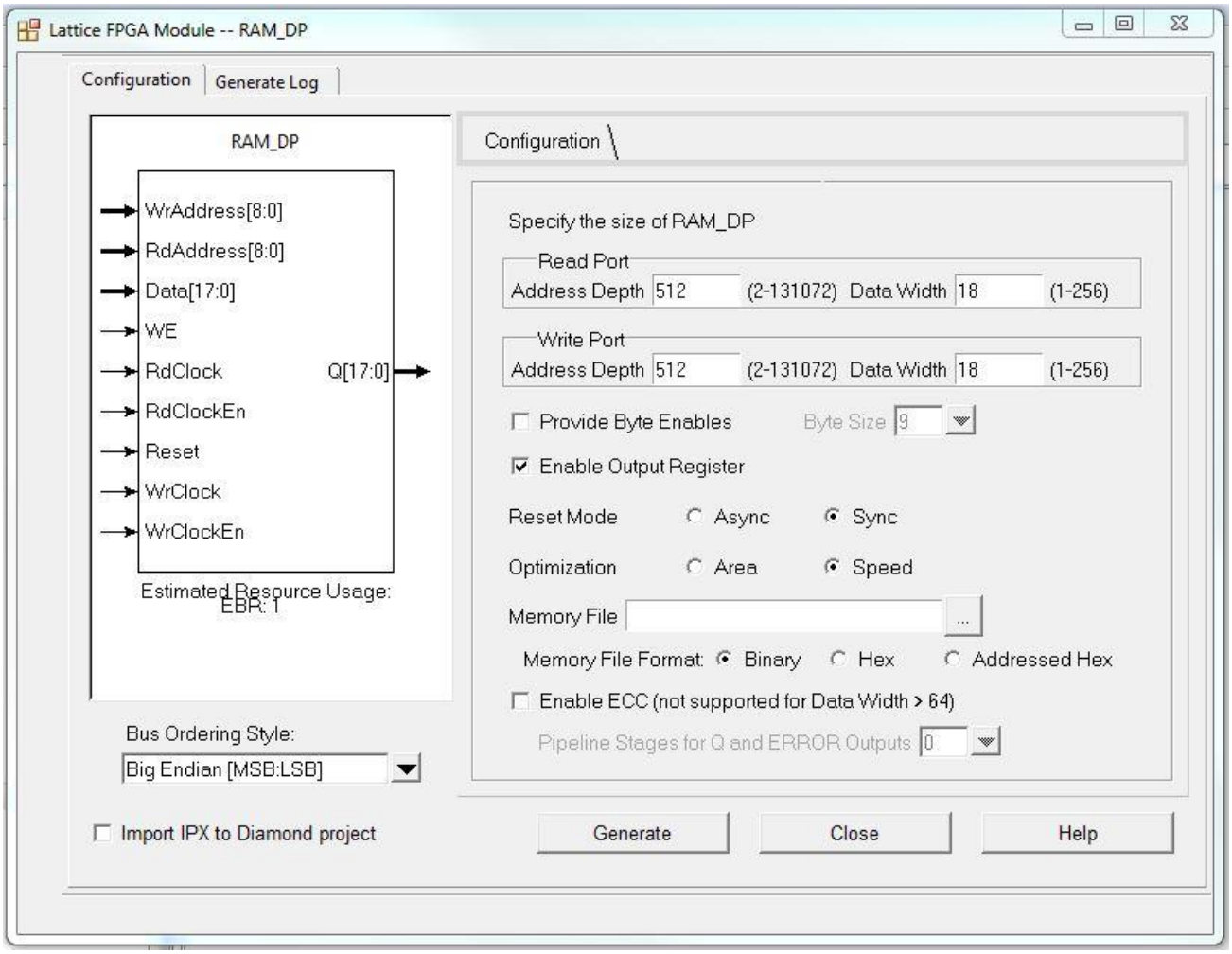

Figure 3.3. Example Generating Pseudo Dual Port RAM (RAM_DP) Module Customization

The left side of this window shows the block diagram of the module. The right side includes the **Configuration** tab where you can choose options to customize the RAM_DP (for example, specify 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, we are generating a Pseudo Dual Port RAM of size 512 × 16. 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 check box **Enable Output Registers** inserts the output registers in the Read Data Port. Output registers are optional for EBR-based RAMs.

You have the option to set the **Reset Mode** as Asynchronous Reset or Synchronous Reset. **Enable GSR** can be checked to enable the Global Set Reset.

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 section of this document.

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

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

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

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

3.2. Byte Order with Different Port Widths

When instantiating memories that have different port widths, the following examples show the byte order as it relates to endian of the memory input and output.

Example 1: 8-bit Write, 32-bit Read

Big Endian Write Order – Byte[31:24], Byte[23:16], Byte[15:8], Byte[7:0]

Big Endian Read Order – Word[31:0]

Little Endian Write Order – Byte[0:7], Byte[8:15], Byte[16:23], Byte[24:31]

Little Endian Read Order – Word[0:31]

Example 2: 32-bit Write, 8-bit Read

Big Endian Write Order – Word[31:0]

Big Endian Read Order – Byte[31:24], Byte[23:16], Byte[15:8], Byte[7:0]

Little Endian Write Order – Word[0:31]

Little Endian Read Order – Byte[0:7], Byte[8:15], Byte[16:23], Byte[24:31]

3.3. ECC in Memory Modules

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

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

3.4. Utilizing PMI

Parameterizable Module Instantiation (PMI) allows experienced users to skip the graphical interface and utilize the configurable memory modules on-the-fly from the ispLEVER Project Navigator.

The necessary parameters and control signals 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 ispLEVER 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.

PMI modules are instantiated the same way other modules are in your HDL. The process is similar to the process for IPexpress with the addition of setting parameters to customize the module. The ispLEVER software provides a template for the Verilog or VHDL instantiation command that specifies the customized module ports and parameters. Refer to the ispLEVER online help section Instantiating a PMI Module for further information.

3.5. Memory Module Inference

Finally, memories may be instantiated within Verilog or VHDL modules through inference. The HDL constructs for memory inferencing is synthesis vendor dependent. Refer to the documentation provided by the synthesis engine vendor for correct inference constructs and attribute settings.

4. Memory Modules

4.1. Single Port RAM (RAM_DQ) – EBR Based

The EBR blocks in LatticeXP2 devices can be configured as Single Port RAM or RAM_DQ. IPexpress allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirements.

IPexpress generates the memory module as shown in Figure 4.1.

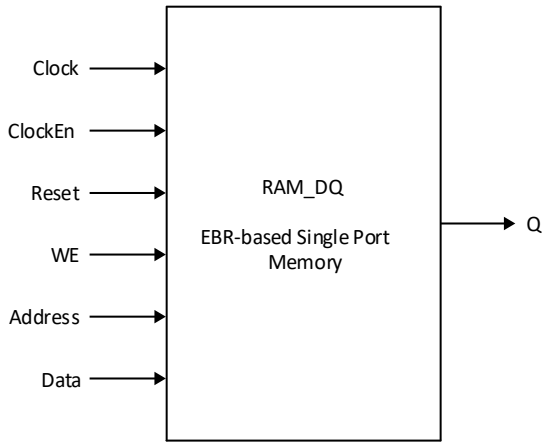

Figure 4.1. Single Port Memory Module Generated by IPexpress

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 specified by the user in the IPexpress GUI. For memory sizes smaller than an EBR block, the module is created in one EBR block. For memory sizes larger than one EBR block, multiple EBR blocks can be cascaded in depth or width (as required to create these sizes).

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

The various ports and their definitions for the Single Port Memory are listed in Table 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 Generated	Port Name in the EBR Block	Description	Active State
Clock	CLK	Clock	Rising Clock Edge
ClockEn	CE	Clock Enable	Active High
Address	AD[x:0]	Address Bus	—
Data	DI[y:0]	Data In	—
Q	DO[y:0]	Data Out	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

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

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

software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

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

Table 4.2. Single Port Memory Sizes for 16K Memories for LatticeXP2

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

Table 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. For detailed attribute definitions, refer to Appendix A.

Table 4.3. Single Port RAM Memories for LatticeXP2

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

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

Additionally, users can select to enable the output registers for RAM_DQ. Figure 4.2, Figure 4.3, Figure 4.4, and Figure 4.5 show the internal timing waveforms for the Single Port RAM (RAM_DQ) with these options.

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

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

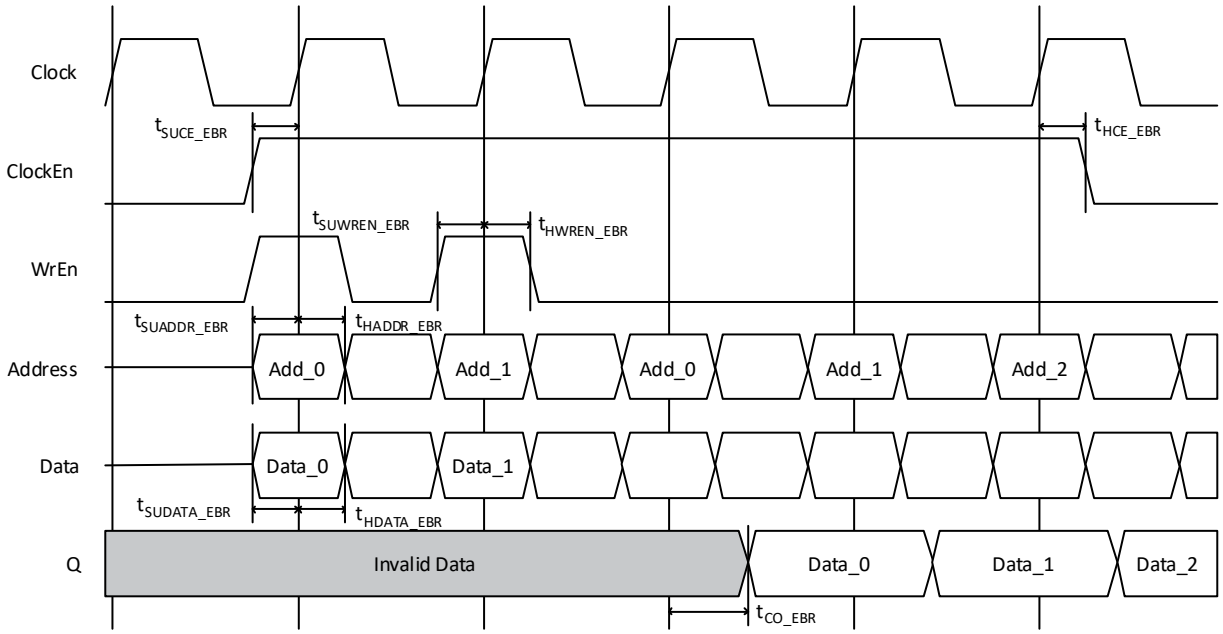

Figure 4.2. Single Port RAM Timing Waveform – NORMAL Mode, without Output Registers

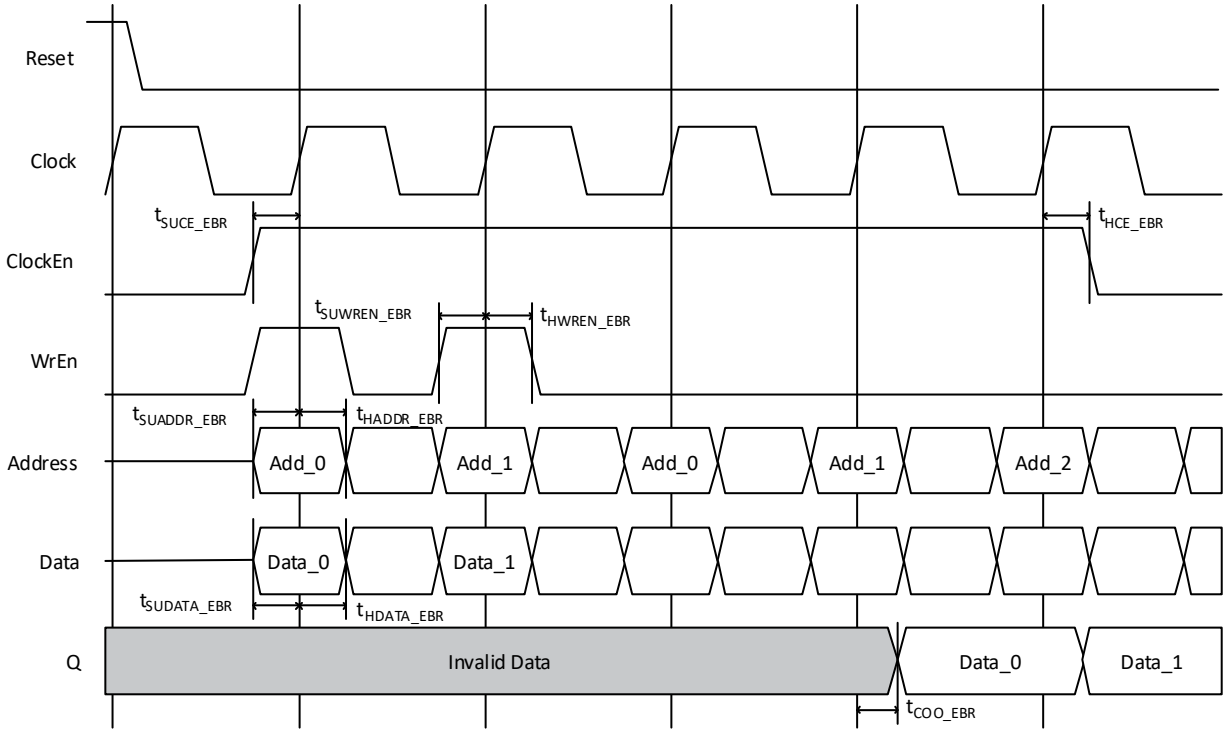

Figure 4.3. Single Port RAM Timing Waveform – NORMAL Mode, with Output Registers

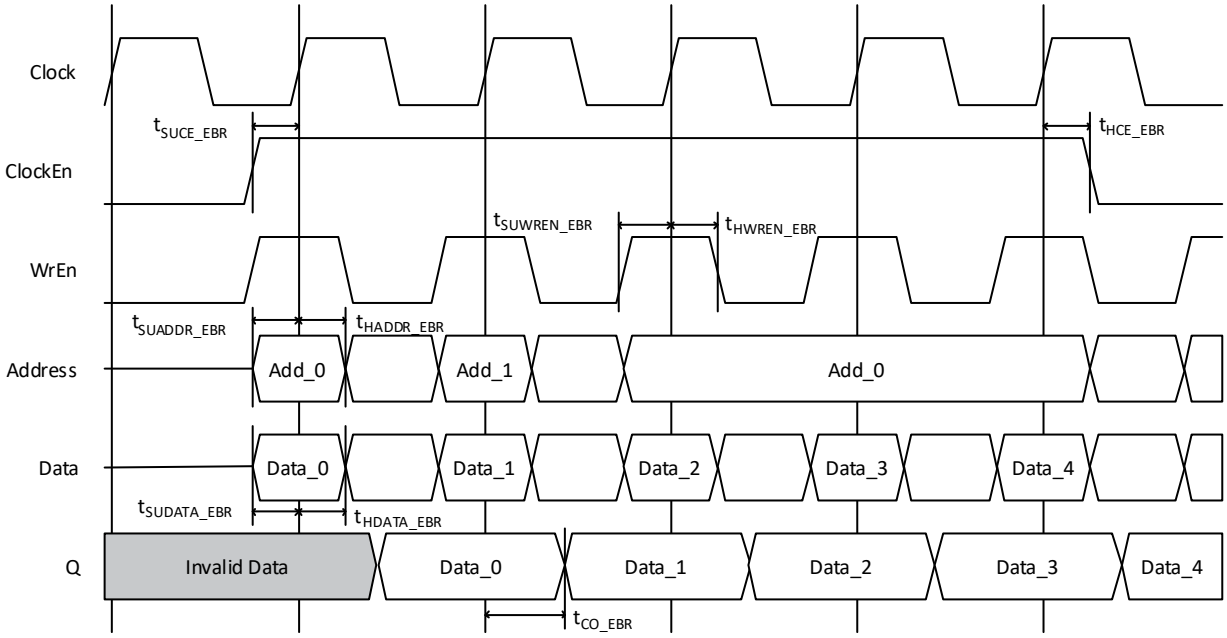

Figure 4.4. Single Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

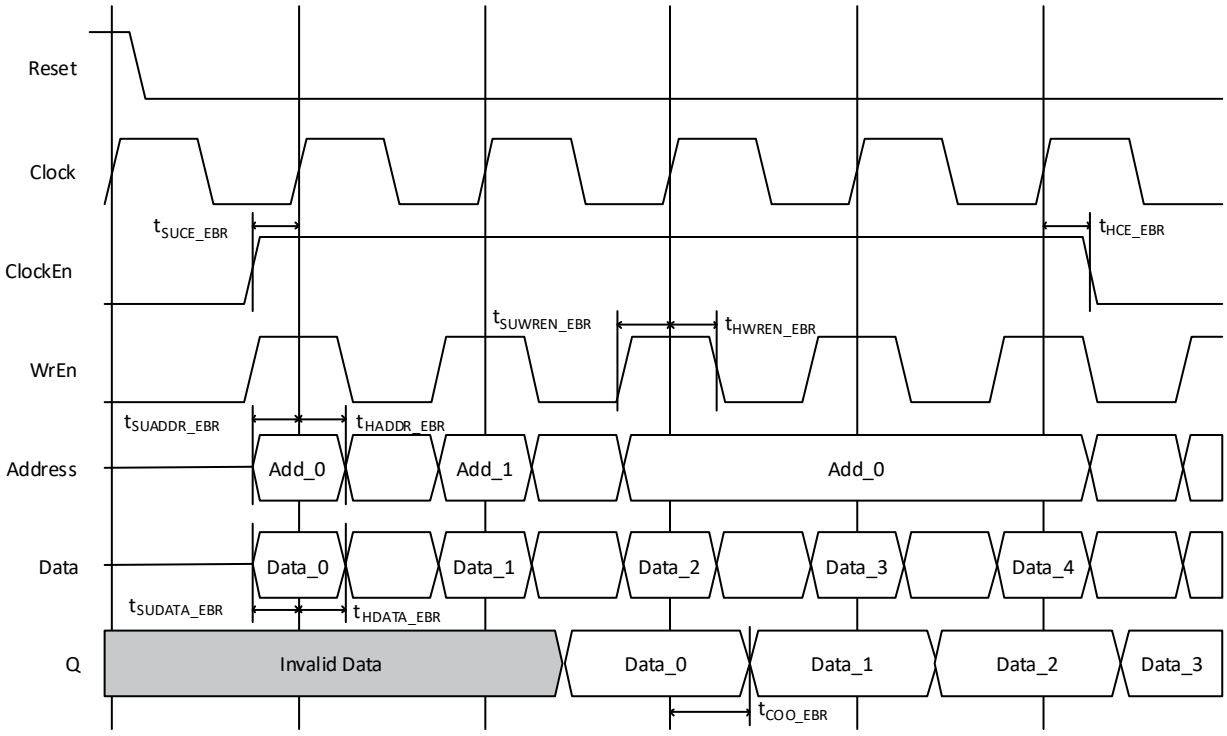

Figure 4.5. Single Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

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

The EBR blocks in the LatticeXP2 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 4.6.

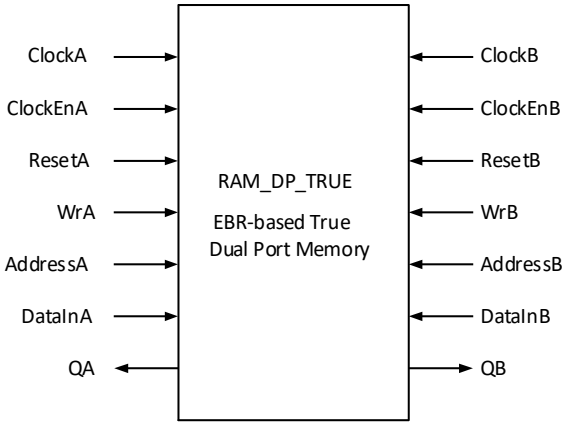

Figure 4.6. True Dual Port Memory Module Generated by IPexpress

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

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

The various ports and their definitions for Single Port Memory are listed in Table 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 Port Memory for 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[x1:0], ADB[x2:0]	Address Bus Port A and Port B	—
DataA, DataB	DIA[y1:0], DIB[y2:0]	Input Data Port A and Port B	—
QA, QB	DOA[y1:0], DOB[y2:0]	Output Data Port A and Port B	—
WrA, WrB	WEA, WEB	Write Enable Port A and Port B	Active High
ResetA, ResetB	RSTA, RSTB	Reset for Port A and Port B	Active High
—	CSA[2:0], CSB[2:0]	Chip Selects for each port	—

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

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

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices are listed in Table 4.5.

Table 4.5. True Dual Port Memory Sizes for 16K Memory for LatticeXP2

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

Table 4.6 shows the various attributes available for the Single Port Memory (RAM_DQ). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to the Appendix A.

Table 4.6. True Dual Port RAM Attributes for LatticeXP2

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Port A Address depth	Address Depth Port A	16K, 8K, 4K, 2K, 1K	—	YES
Port A Data Width	Data Word Width Port A	1, 2, 4, 9, 18	1	YES
Port B Address depth	Address Depth Port B	16K, 8K, 4K, 2K, 1K	—	YES
Port B Data Width	Data Word Width Port B	1, 2, 4, 9, 18	1	YES
Port A Enable Output Registers	Register Mode (Pipelining) for Port A	NOREG, OUTREG	NOREG	YES
Port B Enable Output Registers	Register Mode (Pipelining) for Port B	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format	—	BINARY, HEX, ADDRESSED HEX	—	YES
Port A Write Mode	Read/Write Mode for Port A	NORMAL, WRITE- THROUGH	NORMAL	YES
Port B Write Mode	Read / Write Mode for Port B	NORMAL, WRITE- THROUGH	NORMAL	YES
Chip Select Decode for Port A	Chip Select Decode for Port A	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Chip Select Decode for Port B	Chip Select Decode for Port B	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x00000000000000000000 00000000000000000000 00000000000000000000 0000000000000000.....0xF FFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFF	0x000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000	NO

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

Additionally, users can select to enable the output registers for RAM_DP_TRUE. Figure 4.7 through Figure 4.10 show the internal timing waveforms for the True Dual Port RAM (RAM_DP_TRUE) with these options.

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

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

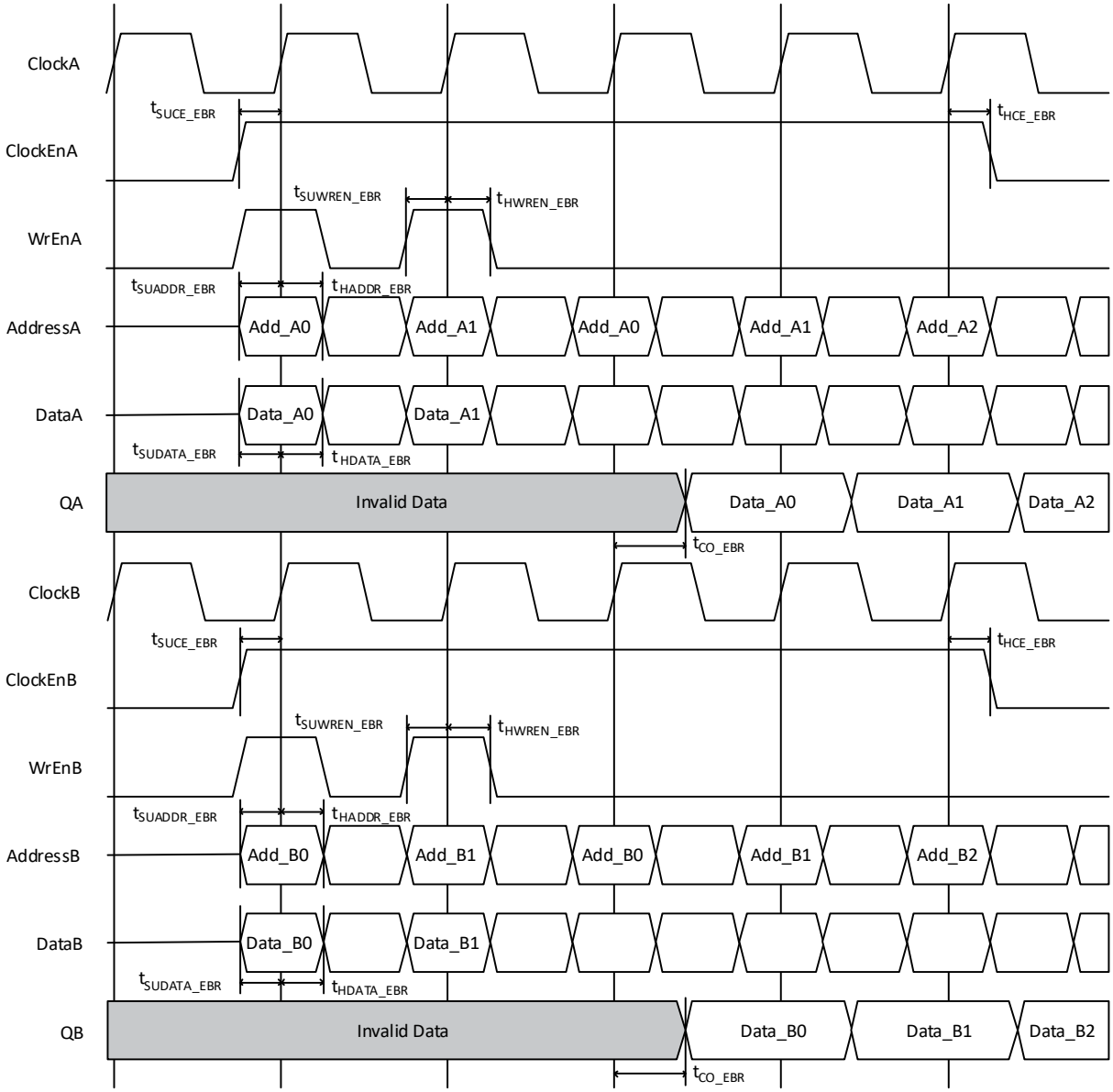

Figure 4.7. True Dual Port RAM Timing Waveform – NORMAL Mode, without Output Registers

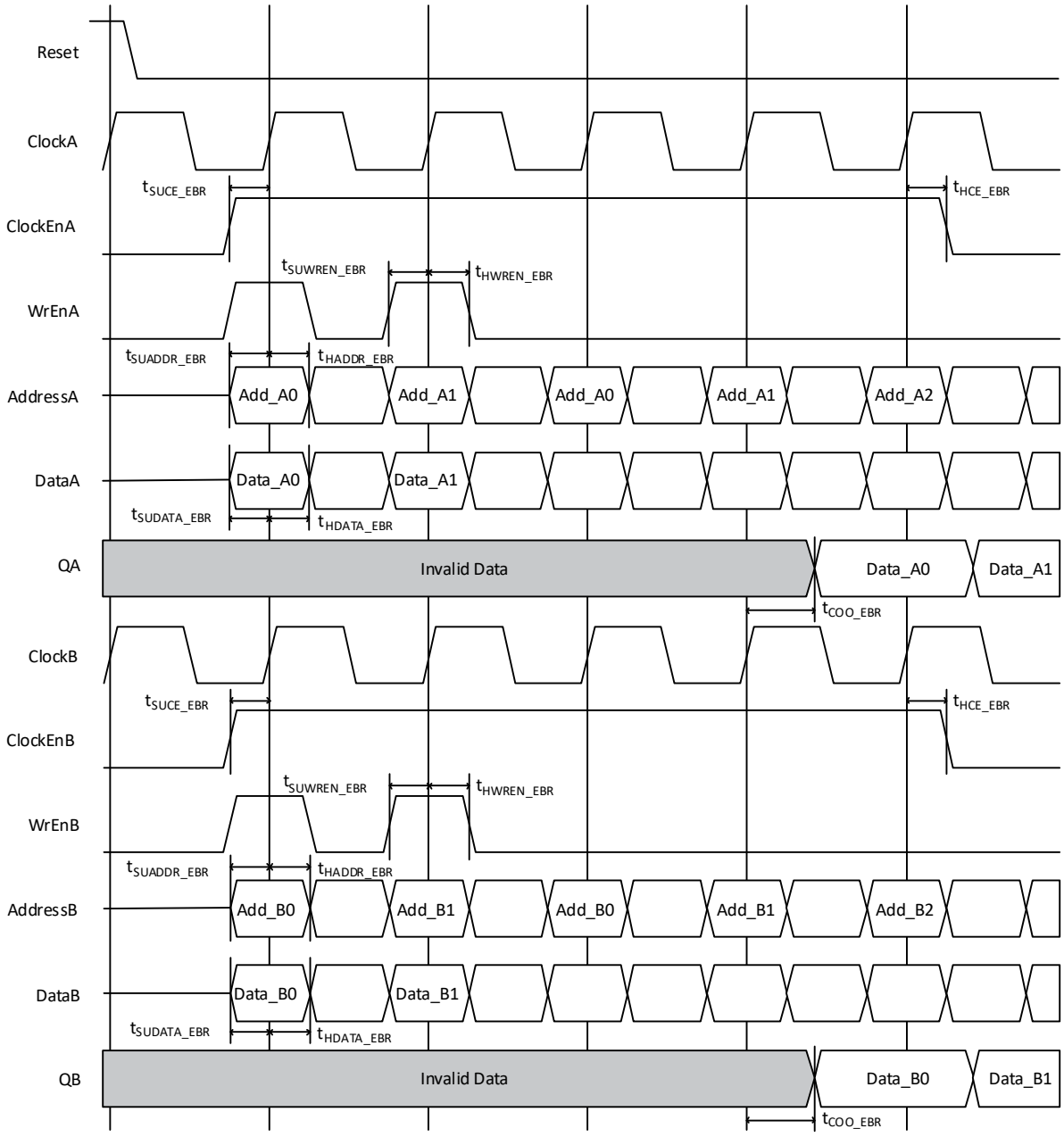

Figure 4.8. True Dual Port RAM Timing Waveform – NORMAL Mode, with Output Registers

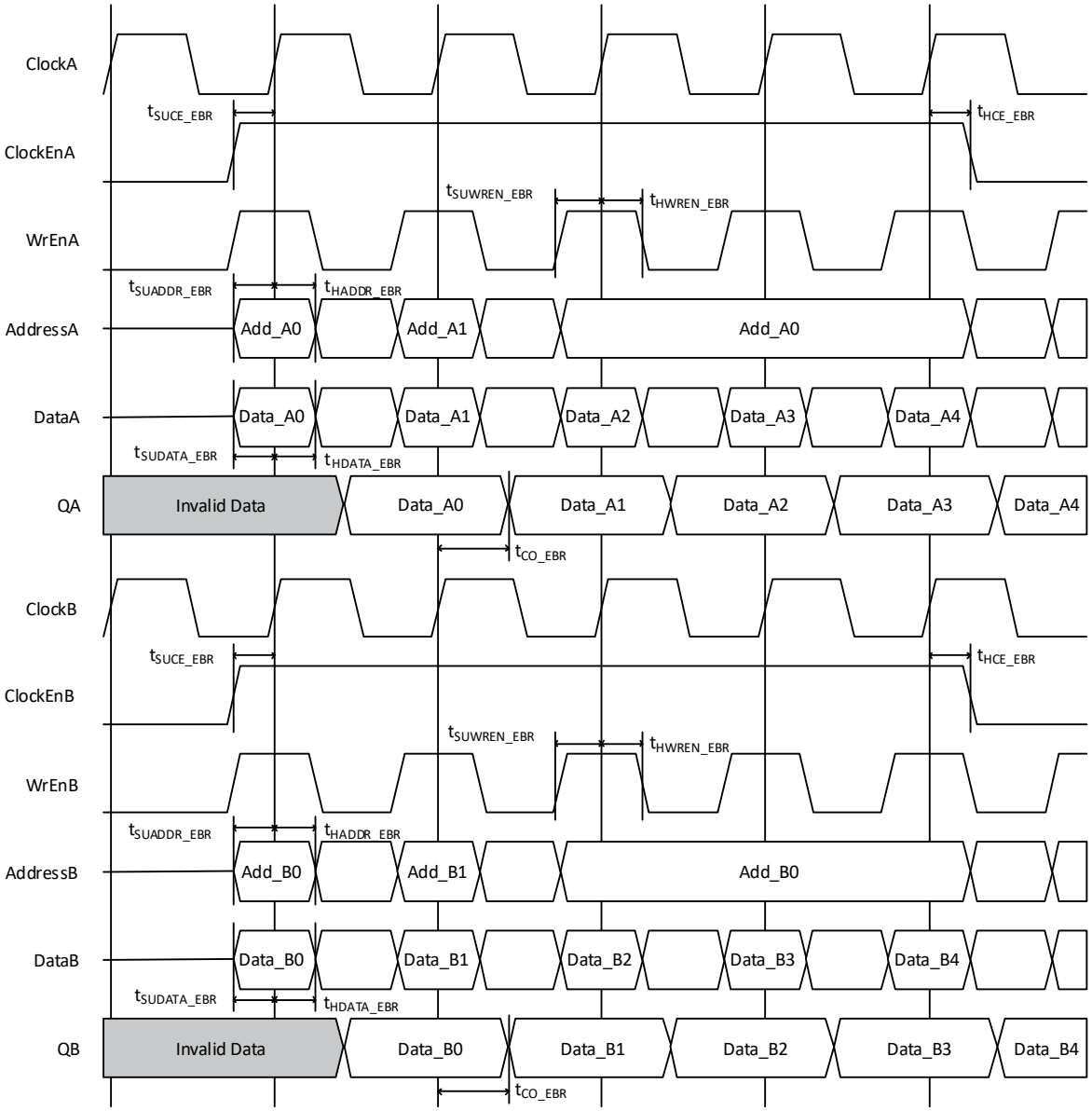

Figure 4.9. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

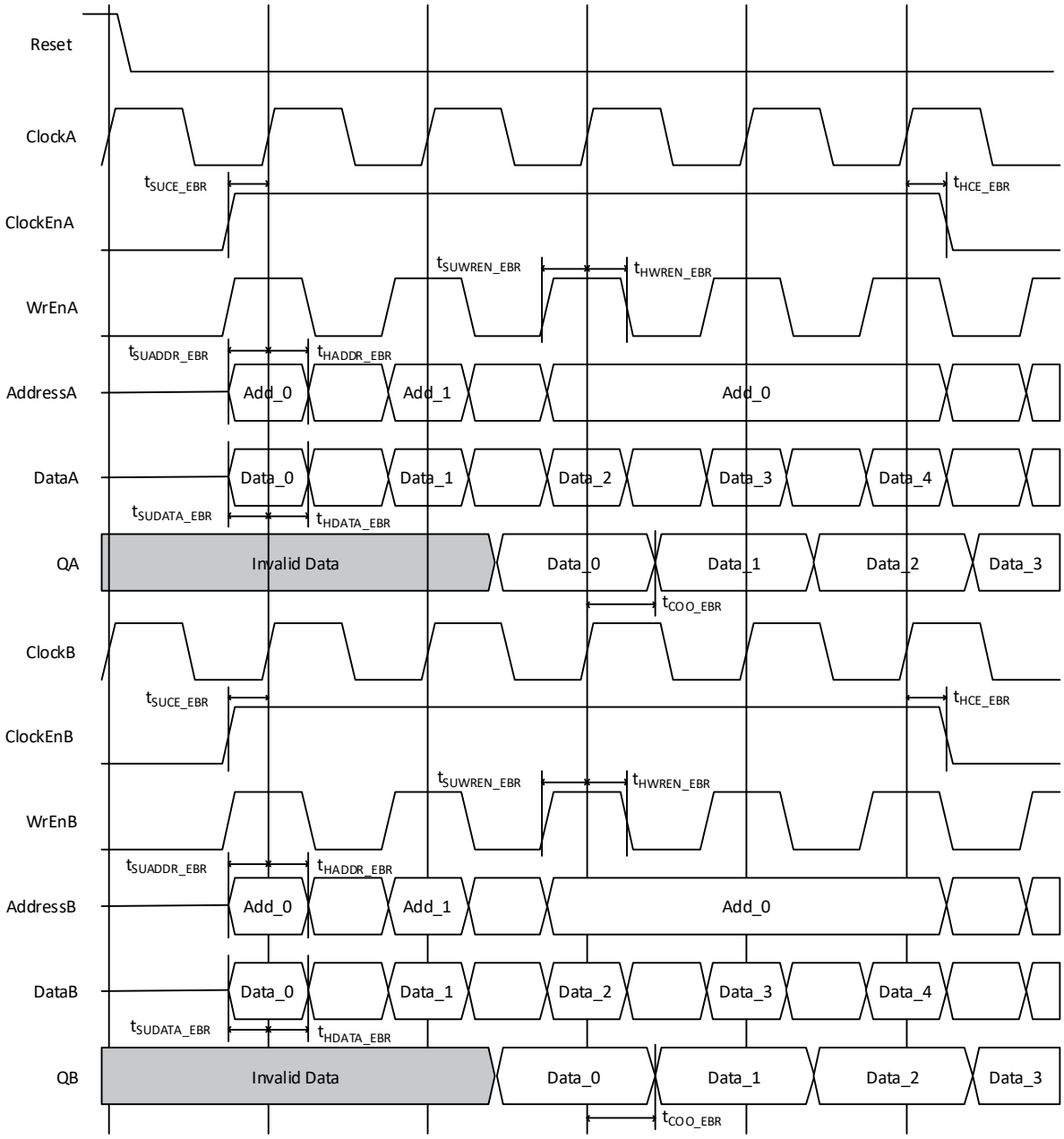

Figure 4.10. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

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

The EBR blocks in LatticeXP2 devices can be configured as Pseudo-Dual Port RAM or RAM_DP. IPexpress allows users to generate the Verilog-HDL or VHDL along with EDIF netlists for the memory size as per design requirements.

IPexpress generates the memory module as shown in Figure 4.11.

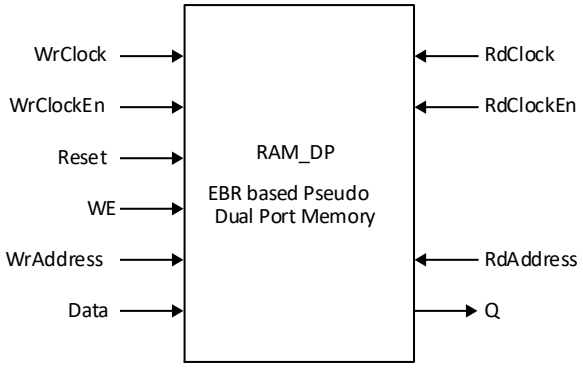

Figure 4.11. Pseudo Dual Port Memory Module Generated by IPexpress

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

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

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

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

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

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices are as in Table 4.8.

Table 4.8. Pseudo-Dual Port Memory Sizes for 16K Memory for LatticeXP2

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

Table 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. For detailed attribute definitions, refer to Appendix A.

Table 4.9. Pseudo-Dual Port RAM Attributes for LatticeXP2

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Read Port Address Depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512	—	YES
Read Port Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Write Port Address Depth	Address Depth Write Port	16K, 8K, 4K, 2K, 1K	—	YES
Write Port Data Width	Data Word Width Write Port	1, 2, 4, 9, 18, 36	1	YES
Write Port Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format	—	BINARY, HEX, ADDRESSED HEX	—	YES
Read Port Write Mode	Read / Write Mode for Read Port	NORMAL	NORMAL	YES
Write Port Write Mode	Read / Write Mode for Write Port	NORMAL	NORMAL	YES
Chip Select Decode for Read Port	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Chip Select Decode for Write Port	Chip Select Decode for Write Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x000000000000000000000000000000000000000000000000000000000000000000000000000000 0......0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0x000000000000000000000000000000000000000000000000000000000000000000000	NO

Users have the option to enable the output registers for Pseudo-Dual Port RAM (RAM_DP). Figure 4.12 and Figure 4.13 show the internal timing waveforms for Pseudo-Dual Port RAM (RAM_DP) with these options. It is important that no setup and hold time violations occur on the address registers (RdAddress and WrAddress). Failing to meet these requirements can result in corruption of memory contents. This applies to both read and write operations.

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

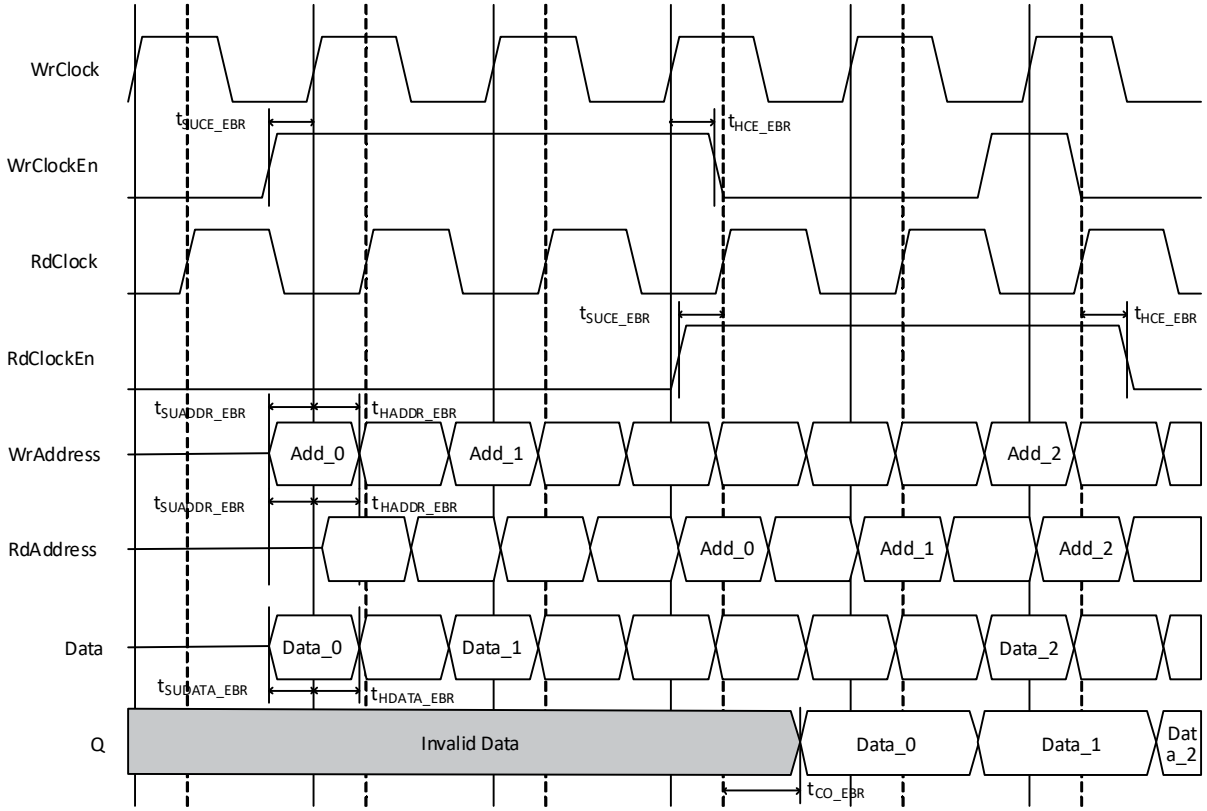

Figure 4.12. PSEUDO DUAL PORT RAM Timing Diagram – without Output Registers

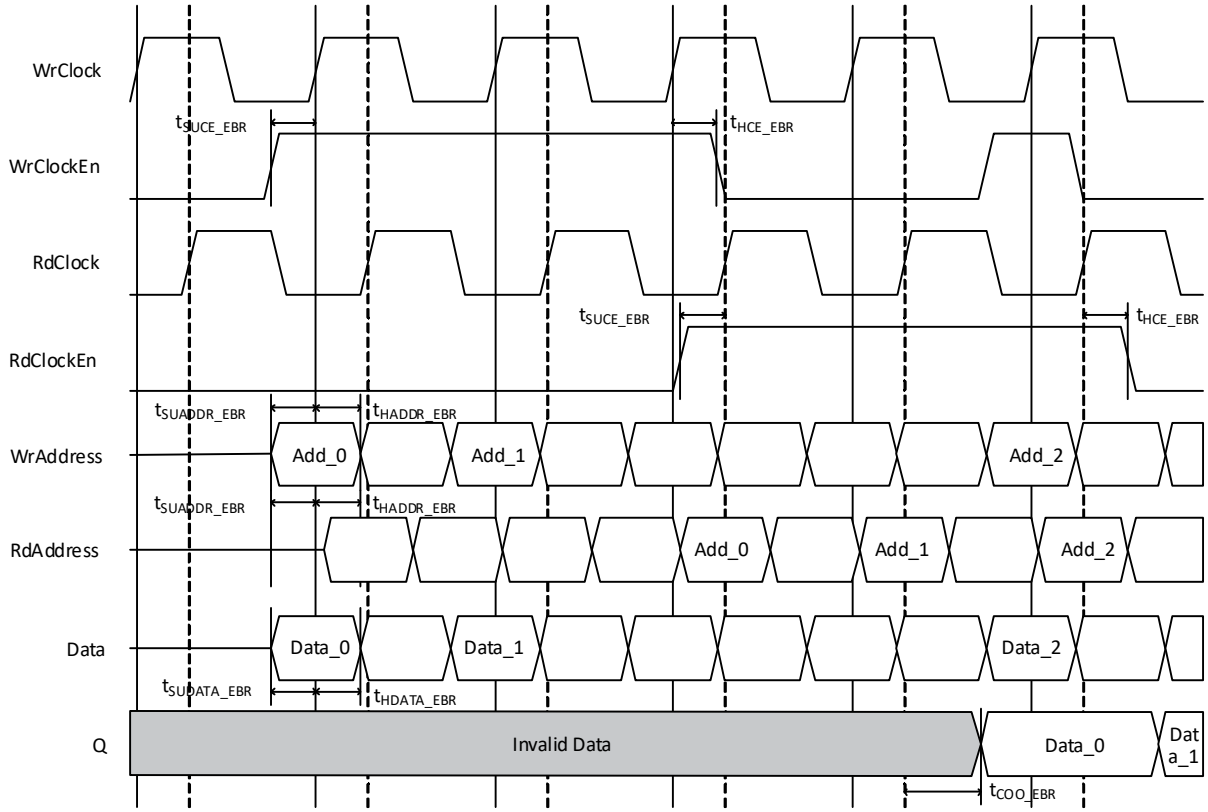

Figure 4.13. PSEUDO DUAL PORT RAM Timing Diagram – with Output Registers

4.4. Read Only Memory (ROM) – EBR Based

The EBR blocks in the LatticeXP2 devices can be configured as Read Only Memory or ROM. IPexpress allows users to generate the Verilog-HDL or VHDL and the EDIF netlist for the memory size, as per design requirements. Users are required to provide the ROM memory content in the form of an initialization file.

IPexpress generates the memory module as shown in Figure 4.14.

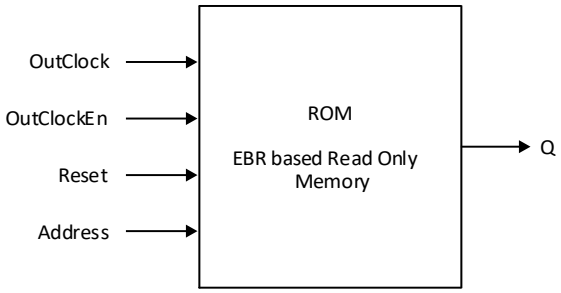

Figure 4.14. Read-Only Memory Module Generated by IPexpress

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

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

The various ports and their definitions for the ROM are listed in Table 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	Active State
Address	AD[x:0]	Read Address	—
OutClock	CLK	Clock	Rising Clock Edge
OutClockEn	CE	Clock Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

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

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

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

Users have the option of enabling the output registers for Read Only Memory (ROM). Figures 23 and 24 show the internal timing waveforms for the Read Only Memory (ROM) with these options.

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices are as per Table 4.11.

Table 4.11. ROM Memory Sizes for 16K Memory for LatticeXP2

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

Table 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.

Table 4.12. Pseudo-Dual Port RAM Attributes for LatticeXP2

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

Users have the option to enable the output registers for Read Only Memory (ROM). Figure 4.15 and Figure 4.16 show the internal timing waveforms for ROM with these options.

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

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

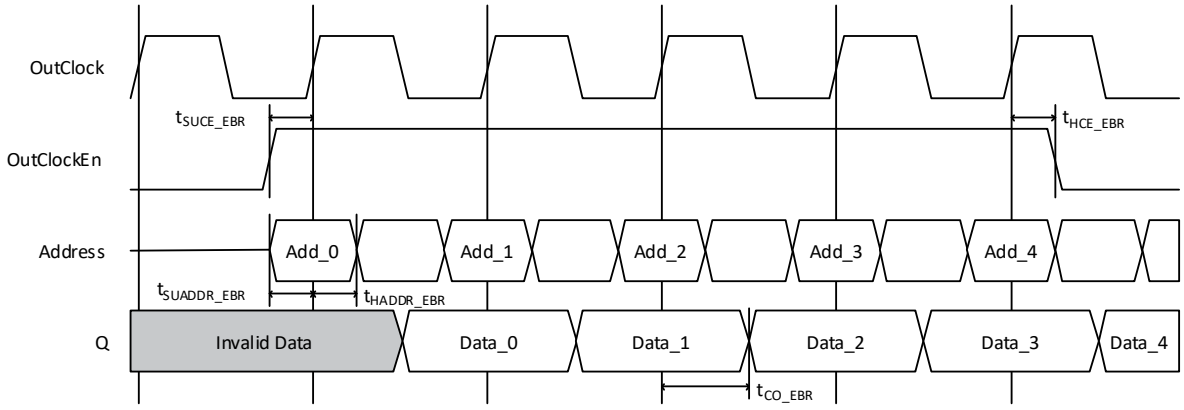

Figure 4.15. ROM Timing Waveform – Without Output Registers

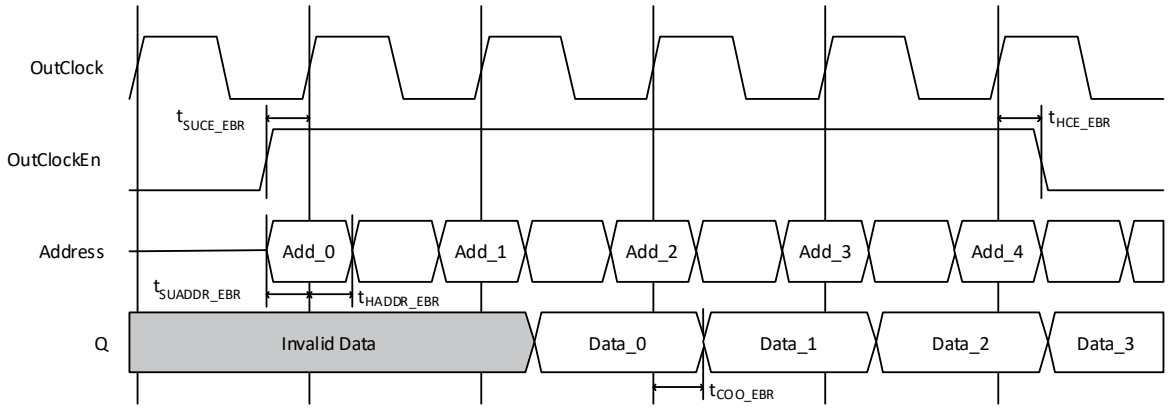

Figure 4.16. ROM Timing Waveform – With Output Registers

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

FIFOs are not supported in certain devices such as the LatticeECP/EC, LatticeECP2/M, LatticeXP and MachXO. The hardware has Embedded Block RAM (EBR) which can be configured in Single Port (RAM_DQ), Pseudo-Dual Port (RAM_DP) and True Dual Port (RAM_DP_TRUE) RAMs. The FIFOs in these devices can be emulated FIFOs that are built around these RAMs. The IPexpress point tool in the ispLEVER design software allows users to build a FIFO and FIFO_DC around Pseudo Dual Port RAM (or DP_RAM).

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

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

4.5.1. 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 4.17 shows the operation of the FIFO when it is empty and the data starts to get written into it.

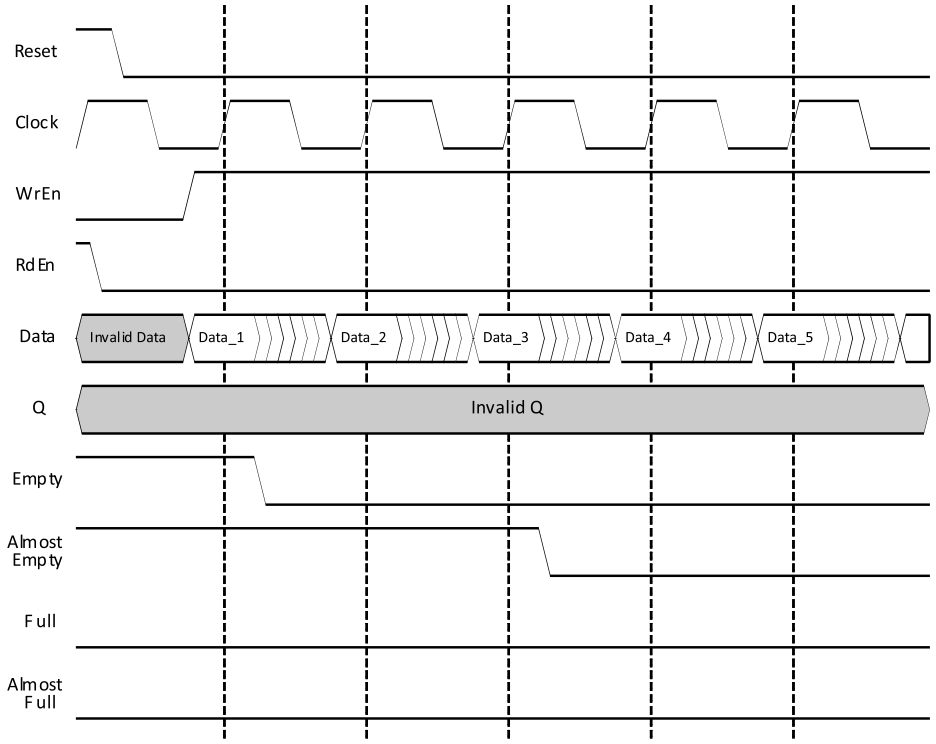

Figure 4.17. FIFO without Output Registers, Start of Data Write Cycle

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

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

Now let us assume that we continue to write into the FIFO to fill it. When the FIFO is filled, the Almost Full and Full Flags are asserted. Figure 4.18 shows the behavior of these flags. In this figure we assume that FIFO depth is 'N'.

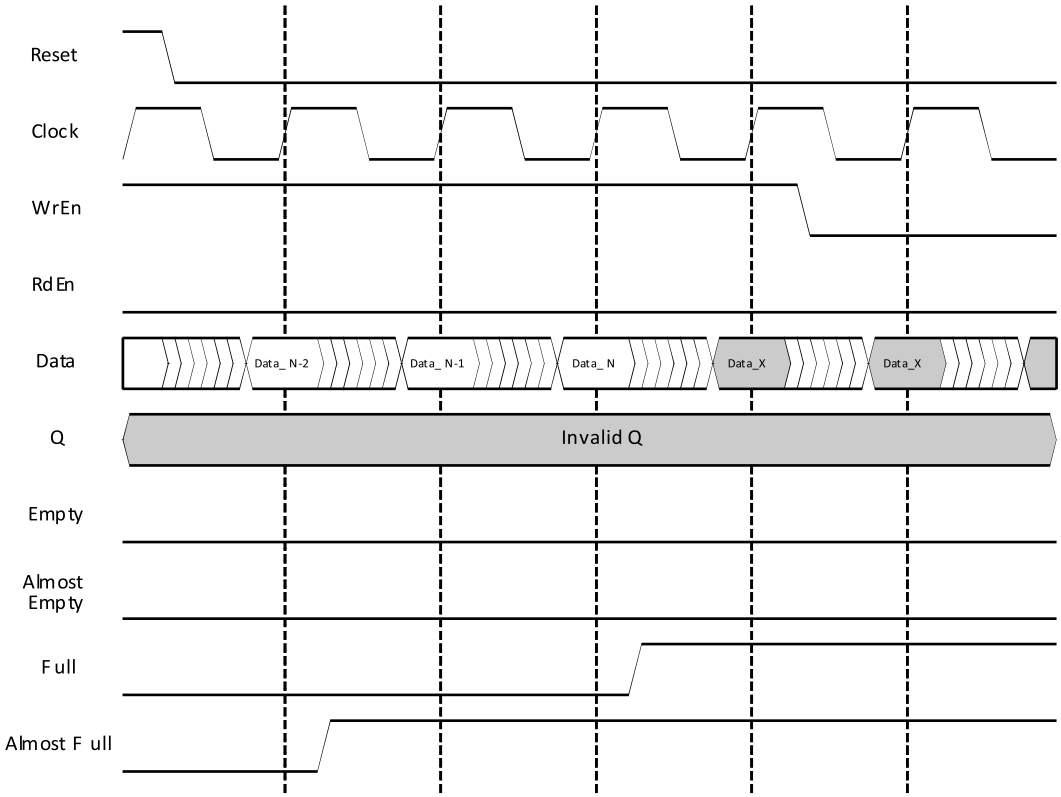

Figure 4.18. FIFO without Output Registers, End of Data Write Cycle

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

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

Now let us look at the waveforms when the contents of the FIFO are read out. Figure 4.19 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted, as shown.

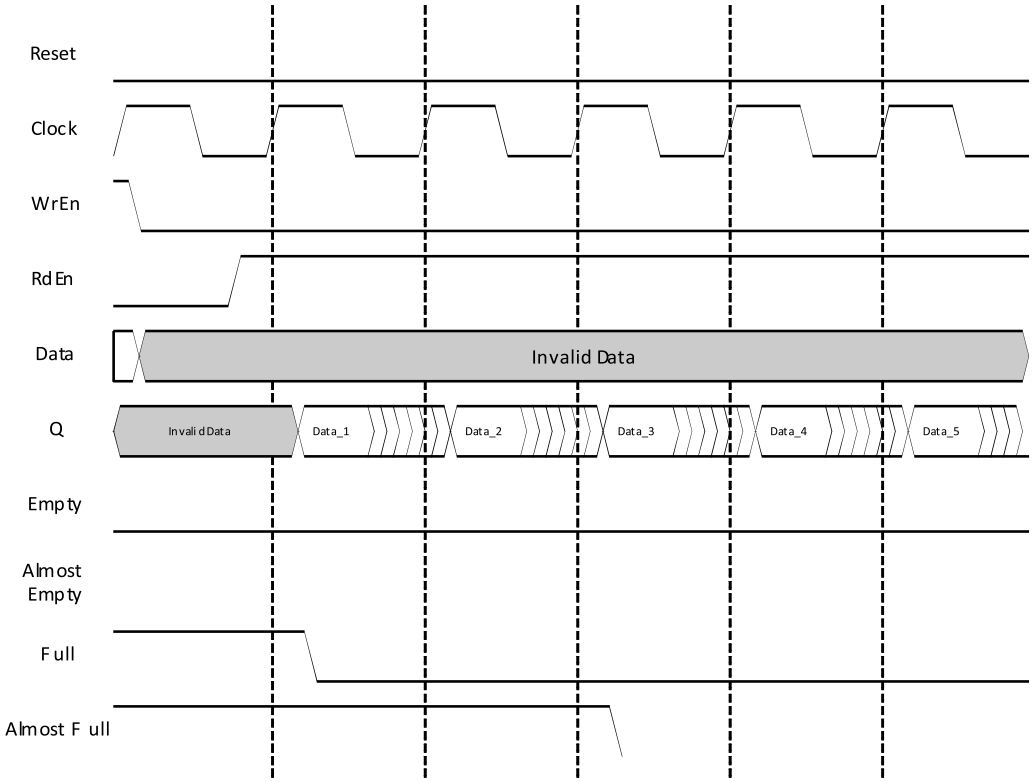

Figure 4.19. FIFO without Output Registers, Start of Data Read Cycle

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

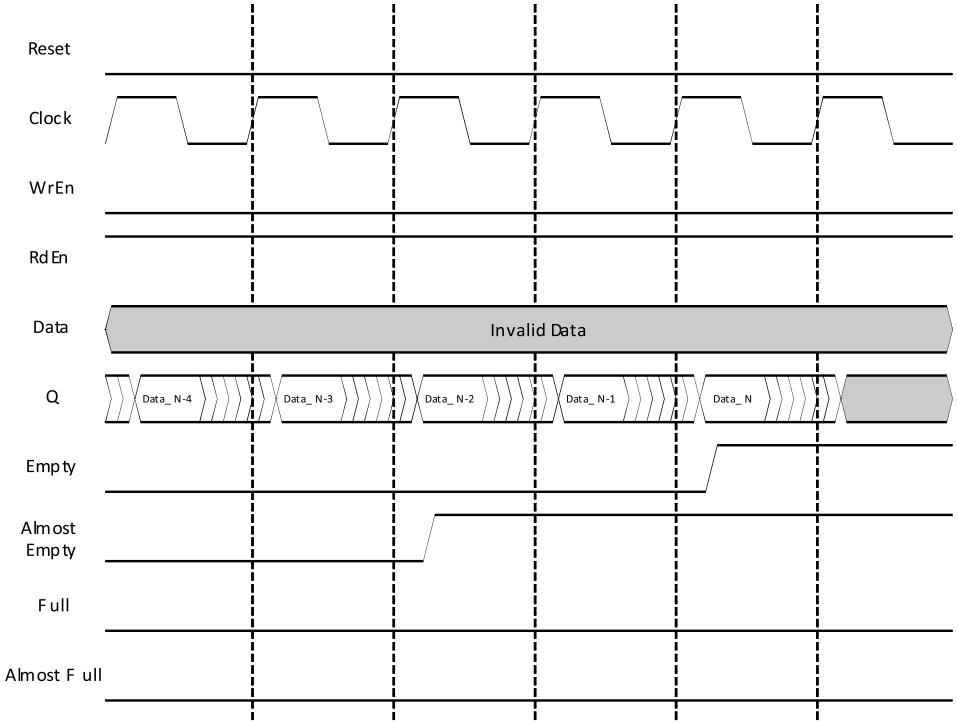

Figure 4.20. FIFO without Output Registers, End of Data Read Cycle

Figure 4.17, Figure 4.18, Figure 4.19, and Figure 4.20 show the behavior of non-pipelined FIFO or FIFO without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is also the extra option for Output registers to be enabled by the RdEn signal.

Figure 4.21, Figure 4.22, Figure 4.23, and Figure 4.24 show the similar waveforms for the FIFO with output register and with output register enable with RdEn. It should be noted that flags are asserted and de-asserted with similar timing to the FIFO without output registers. However, it is only the data out 'Q' that is delayed by one clock cycle.

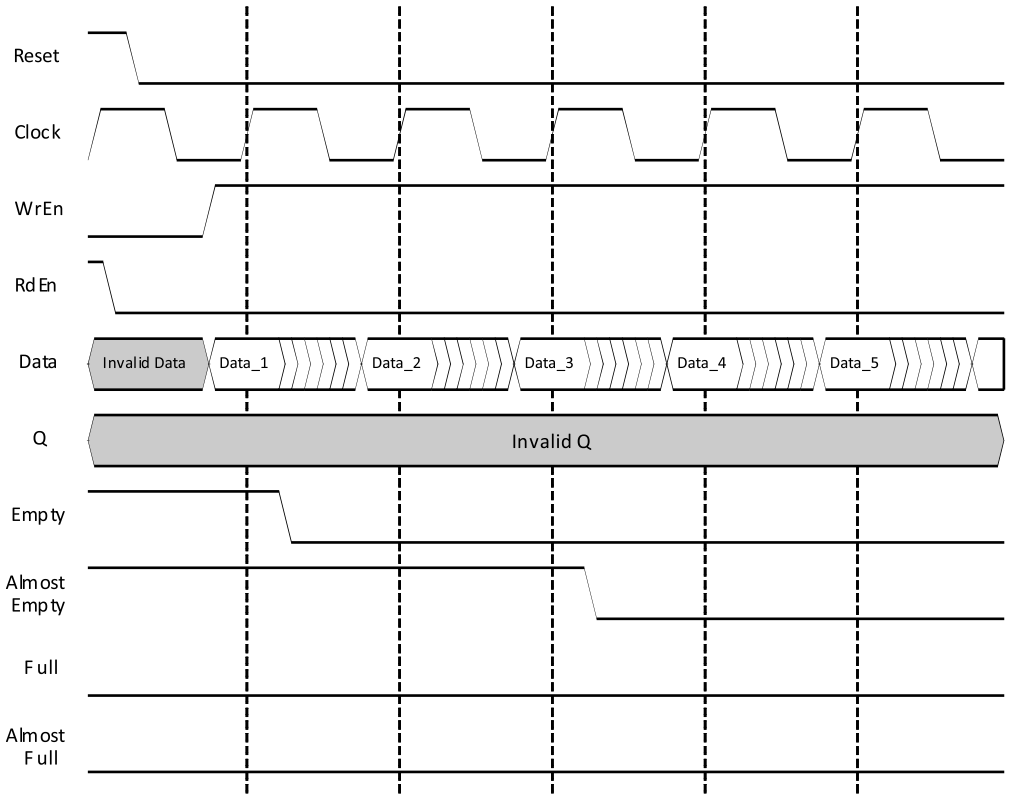

Figure 4.21. FIFO with Output Registers, Start of Data Write Cycle

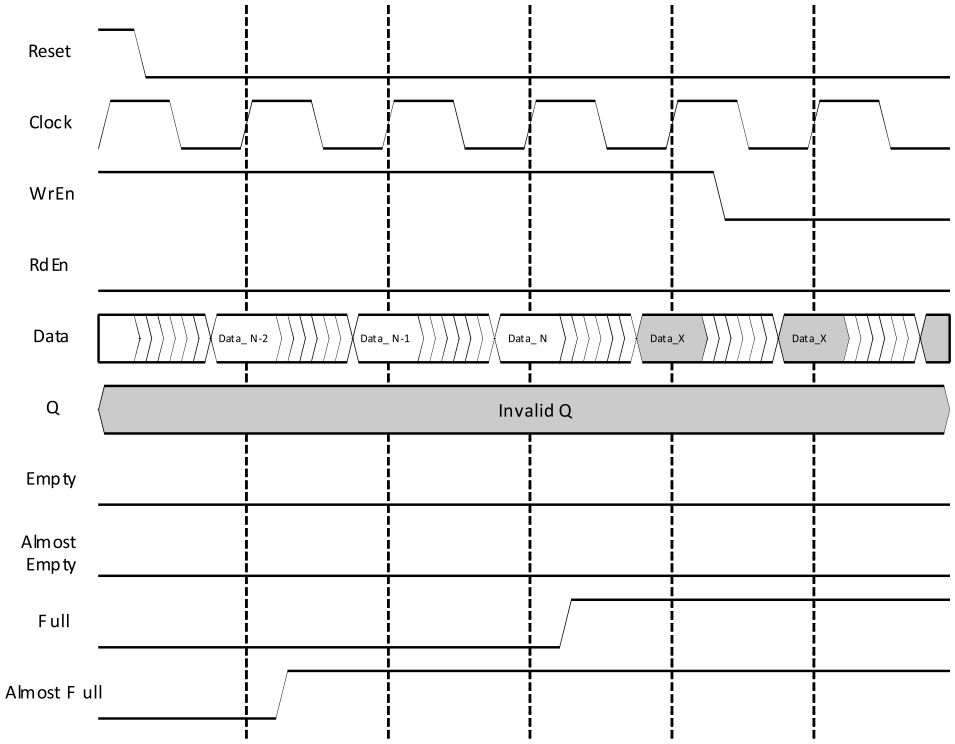

Figure 4.22. FIFO with Output Registers, End of Data Write Cycle

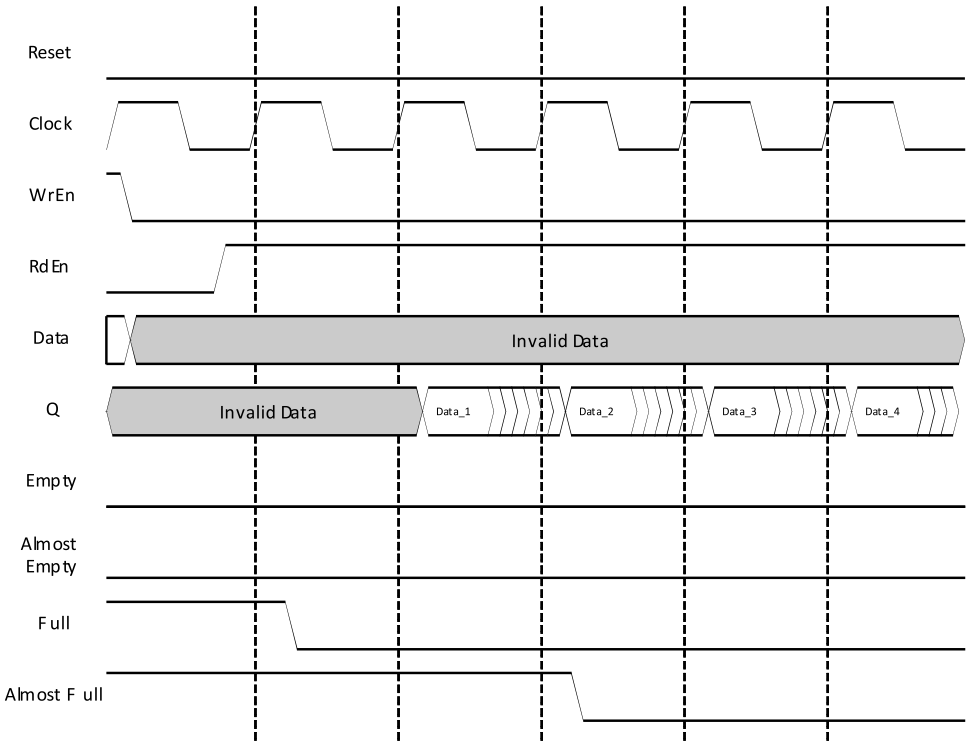

Figure 4.23. FIFO with Output Registers, Start of Data Read Cycle

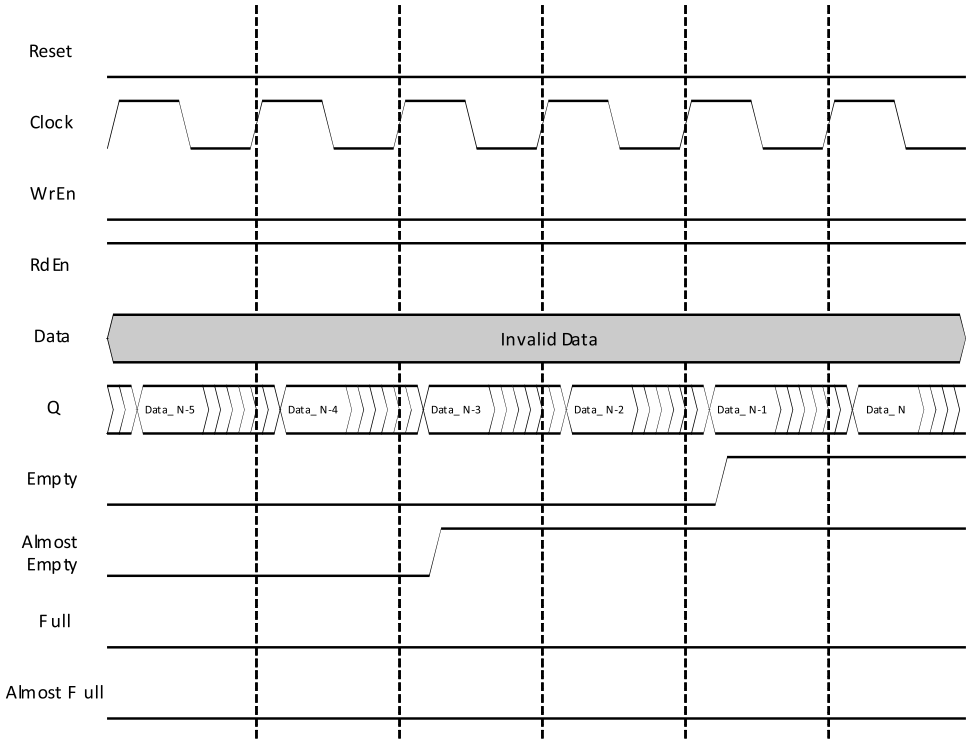

Figure 4.24. FIFO with Output Registers, End of Data Read Cycle

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

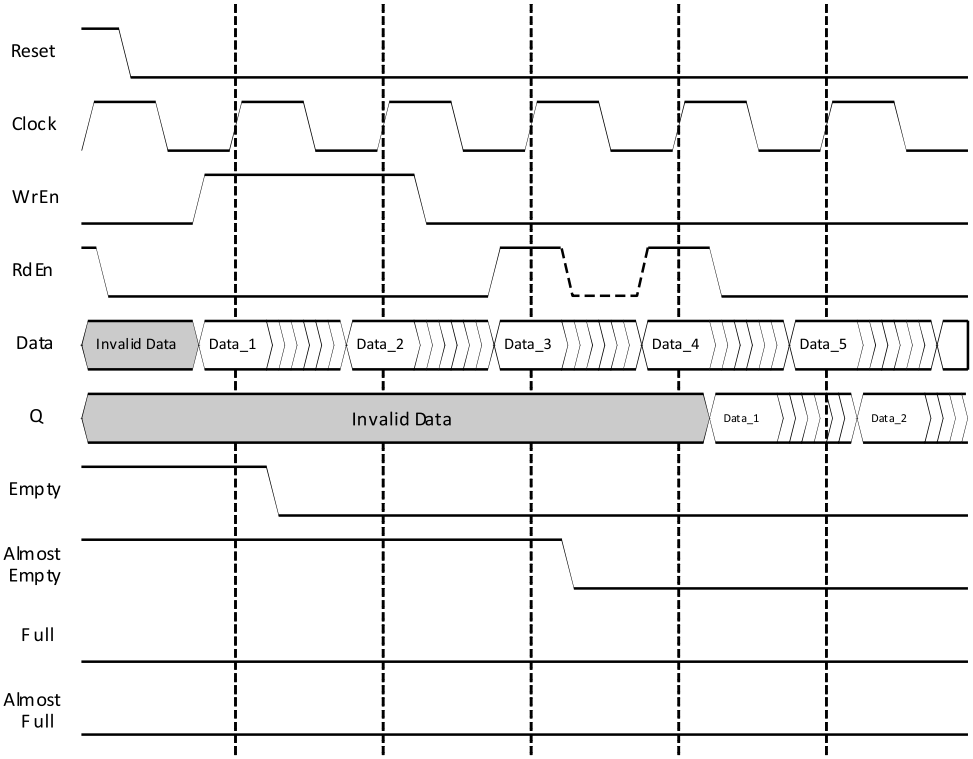

Figure 4.25. FIFO with Output Registers and RdEn on Output Registers

4.5.2. 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

4.5.3. FIFO_DC Flags

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

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

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

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

With this in mind, let us look at the FIFO_DC without output registers waveforms. Figure 4.26 shows the operation of the FIFO_DC when it is empty and the data starts to be written into it.

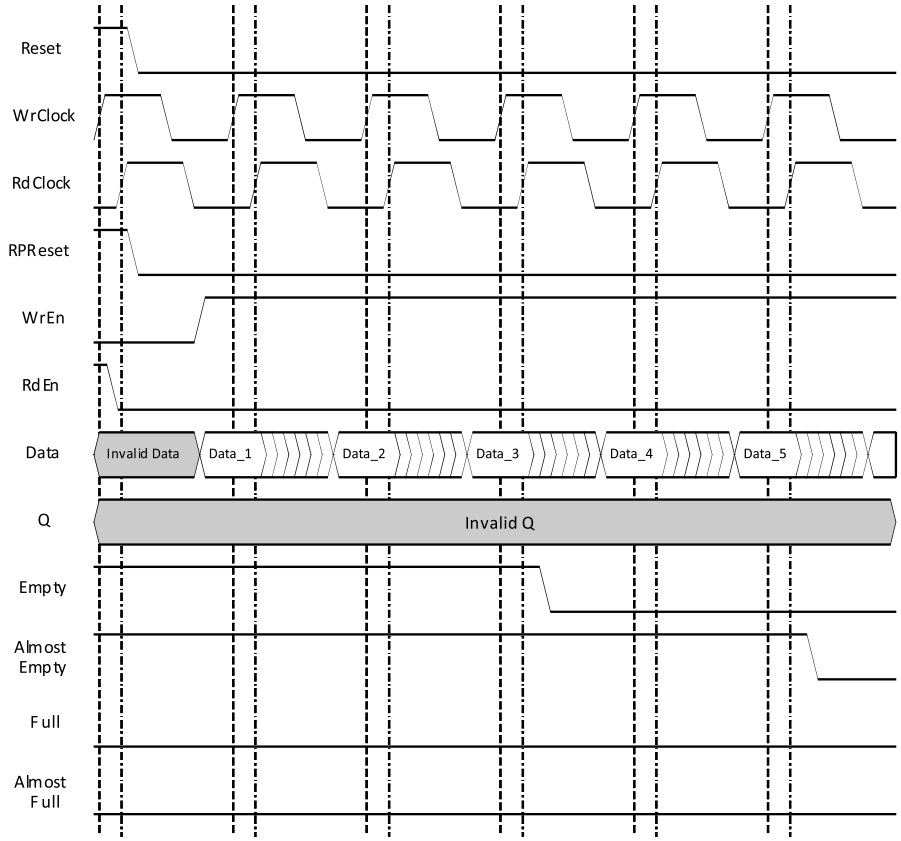

Figure 4.26. FIFO_DC without Output Registers, Start of Data Write Cycle

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

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

Now let us assume that we continue to write into the FIFO_DC to fill it. When the FIFO_DC is filled, the Almost Full and Full Flags are asserted. Figure 4.27 shows the behavior of these flags. In this figure the FIFO_DC depth is 'N'.

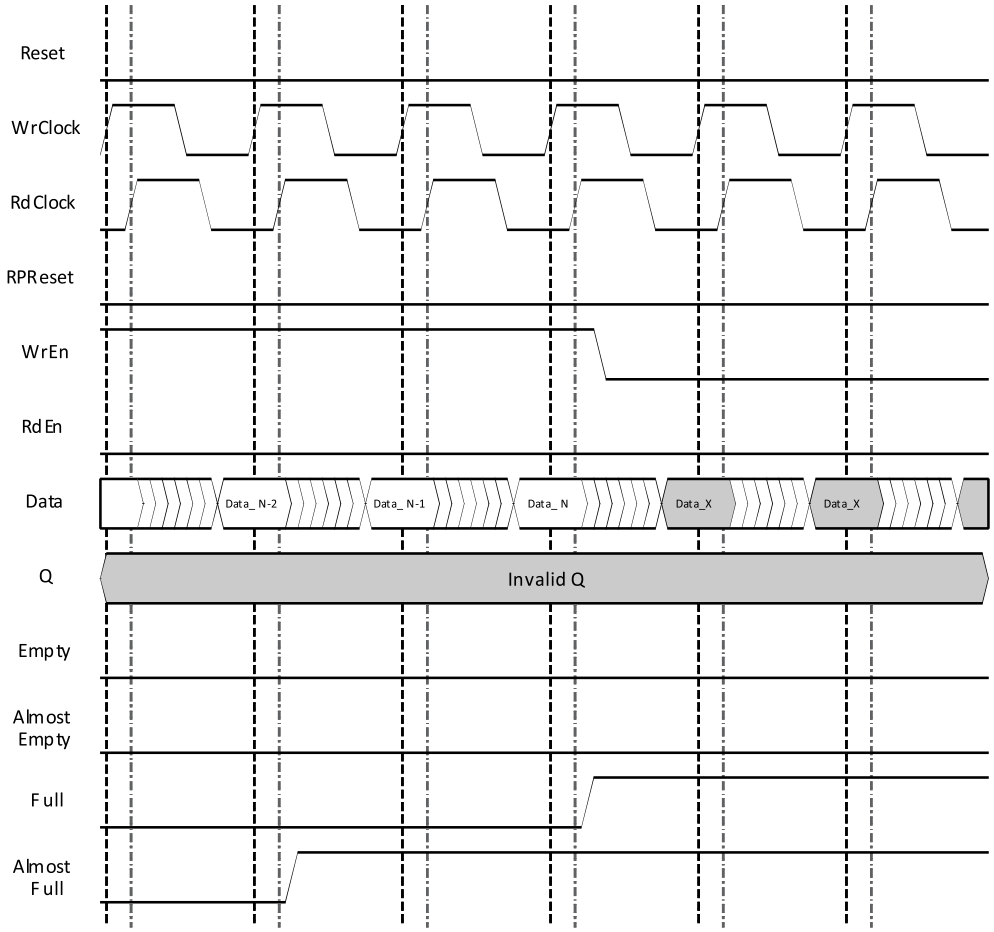

Figure 4.27. FIFO_DC without Output Registers, End of Data Write Cycle

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

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

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

Now let us look at the waveforms when the contents of the FIFO_DC are read out. Figure 4.28 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted, as shown. In this case, note that the de-assertion is delayed by two clock cycles.

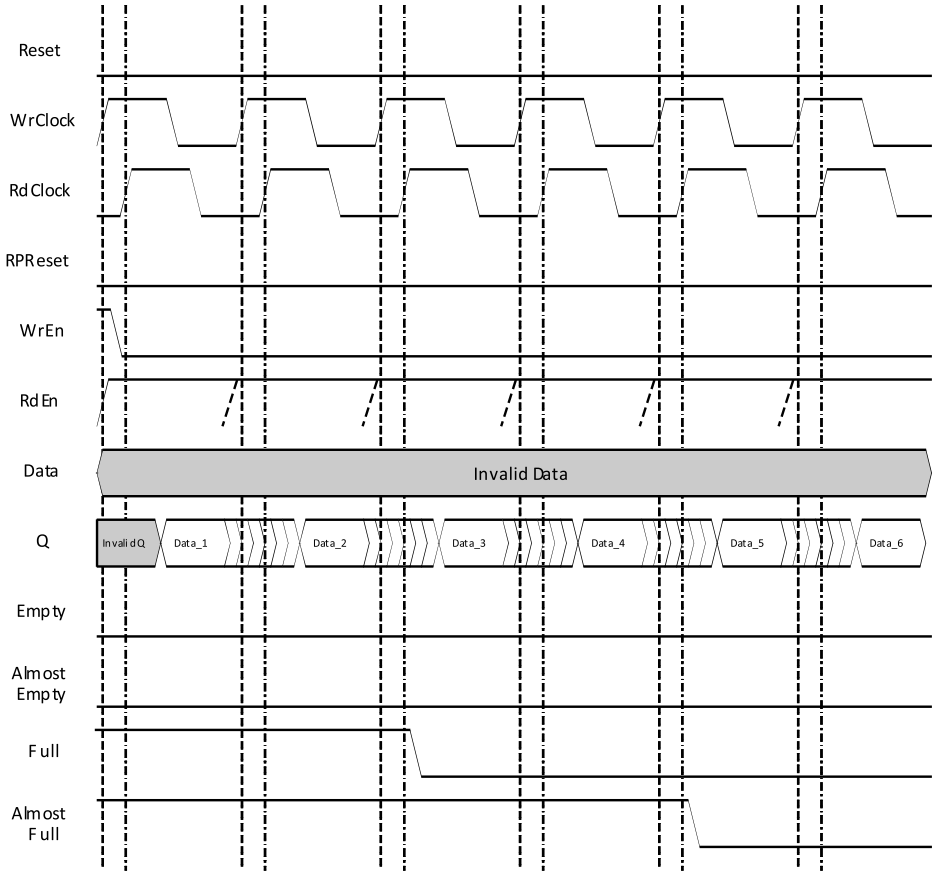

Figure 4.28. FIFO_DC without Output Registers, Start of Data Read Cycle

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

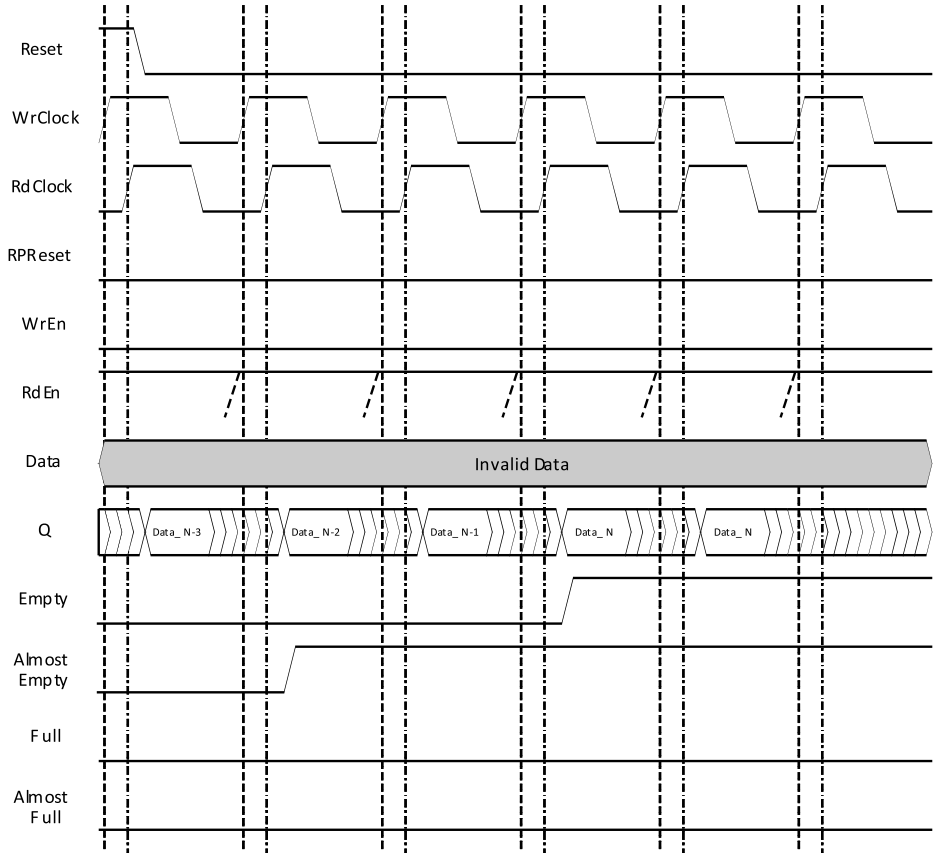

Figure 4.29. FIFO_DC without Output Registers, End of Data Read Cycle

Figure 4.29 show the behavior of non-pipelined FIFO_DC or FIFO_DC without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is an extra option for the output registers to be enabled by the RdEn signal.

Figure 4.30, Figure 4.31, Figure 4.32, and Figure 4.33 show the similar waveforms for the FIFO_DC with output register and without output register enable with RdEn. Note that flags are asserted and de-asserted with similar timing to the FIFO_DC without output registers. However, it is only the data out 'Q' that is delayed by one clock cycle.

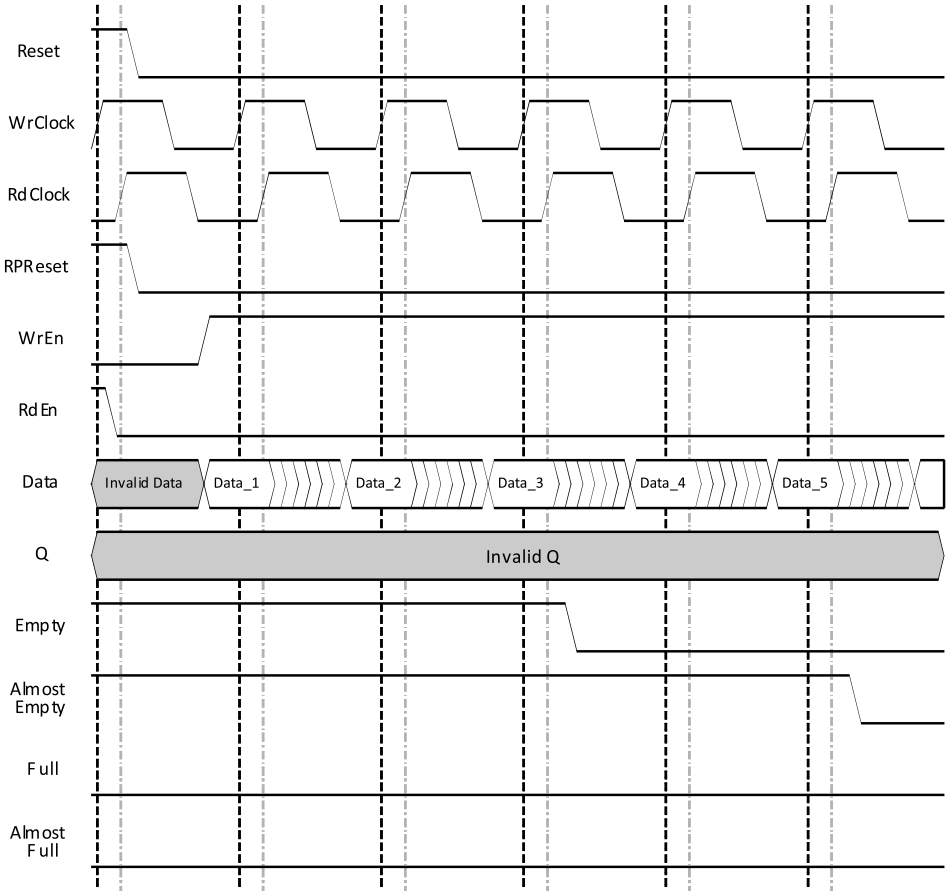

Figure 4.30. FIFO_DC with Output Registers, Start of Data Write Cycle

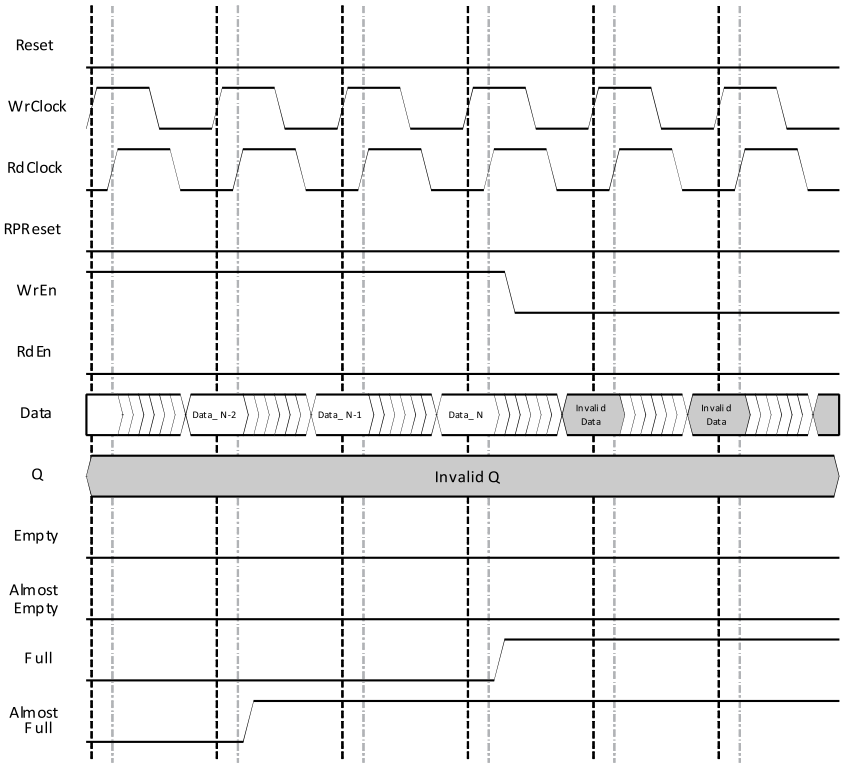

Figure 4.31. FIFO_DC with Output Registers, End of Data Write Cycle

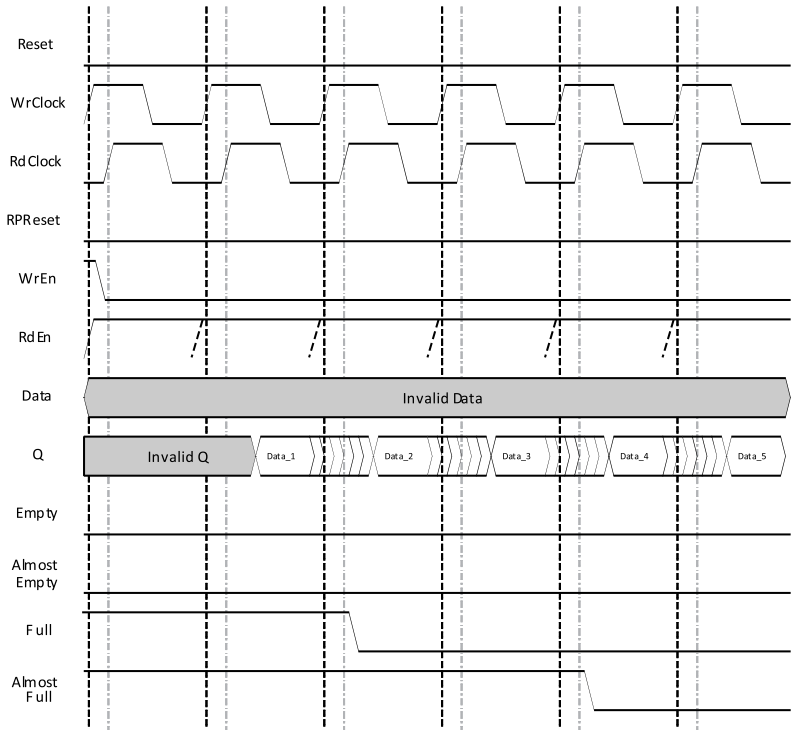

Figure 4.32. FIFO_DC with Output Registers, Start of Data Read Cycle

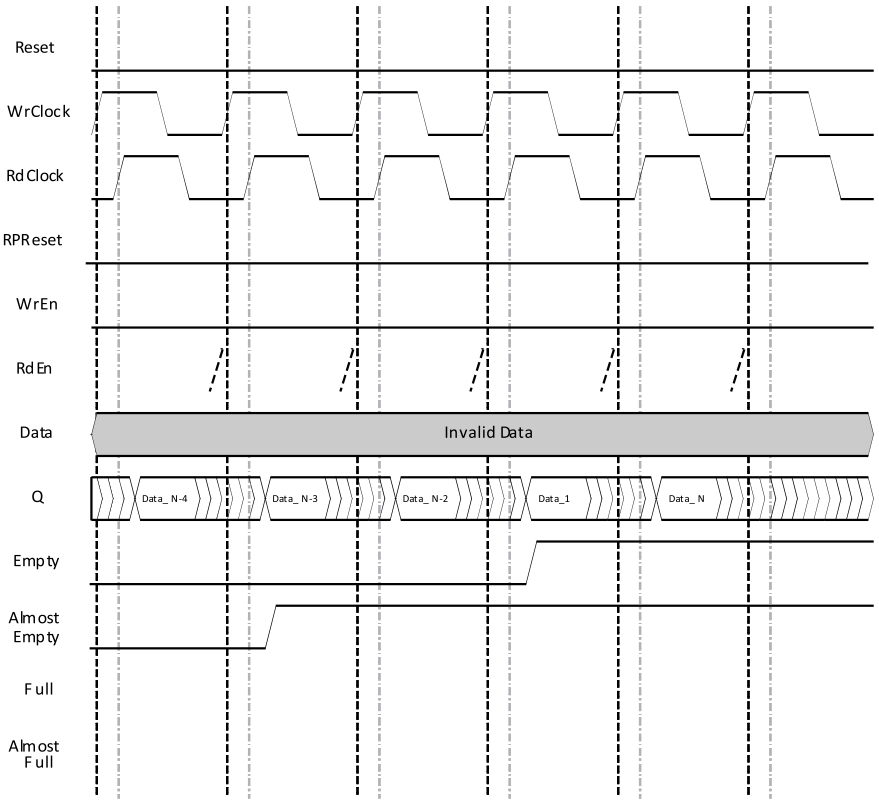

Figure 4.33. FIFO_DC with Output Registers, End of Data Read Cycle

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

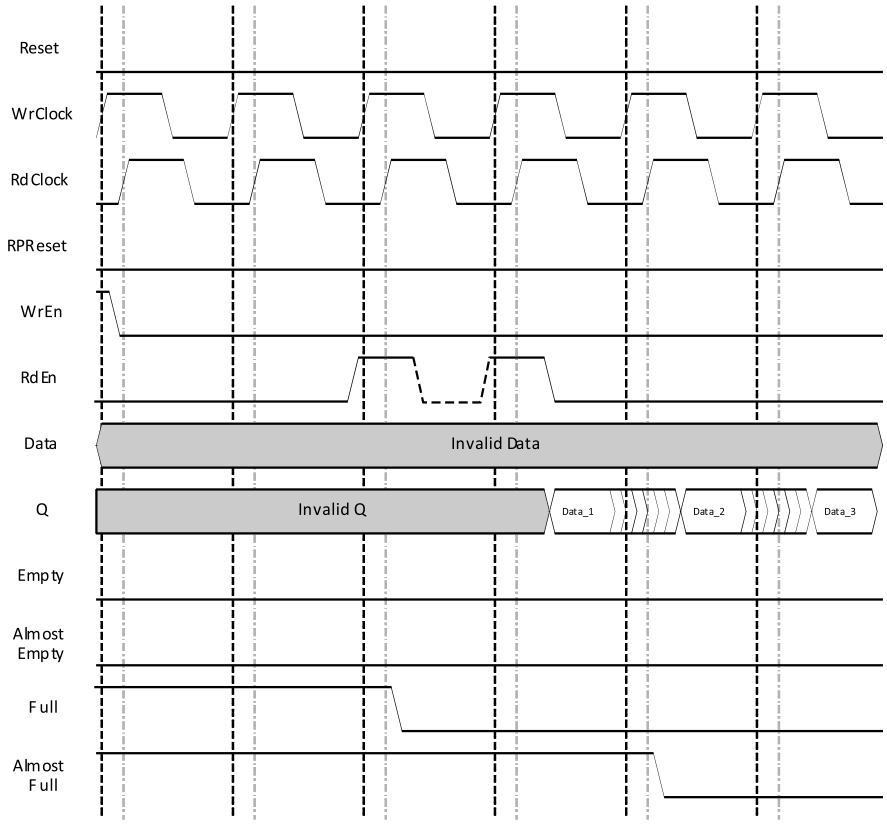

Figure 4.34. FIFO_DC with Output Registers and RdEn on Output Registers

4.6. Distributed Single Port RAM (Distributed_SPRAM) – PFU Based

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

Figure 4.35 shows the Distributed Single Port RAM module as generated by IPexpress.

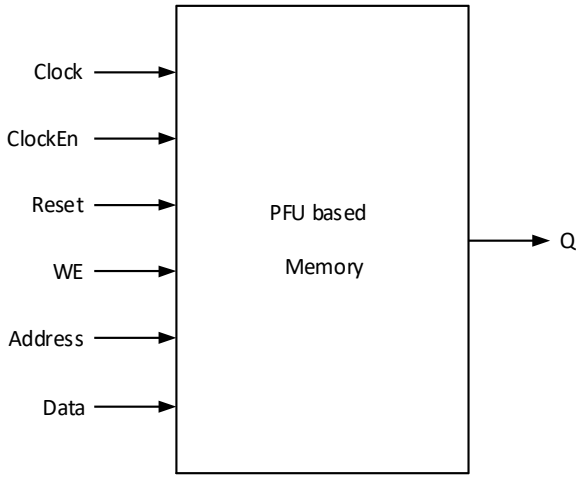

Figure 4.35. Distributed Single Port RAM Module Generated by IPexpress

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

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

The various ports and their definitions for the memory are as per Table 4.13. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 4.13. PFU-based Distributed Single Port RAM Port Definitions

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

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

Users have the option of enabling the output registers for Distributed Single Port RAM (Distributed_SPRAM). Figure 4.36 and Figure 4.37 show the internal timing waveforms for the Distributed Single Port RAM (Distributed_SPRAM) with these options.

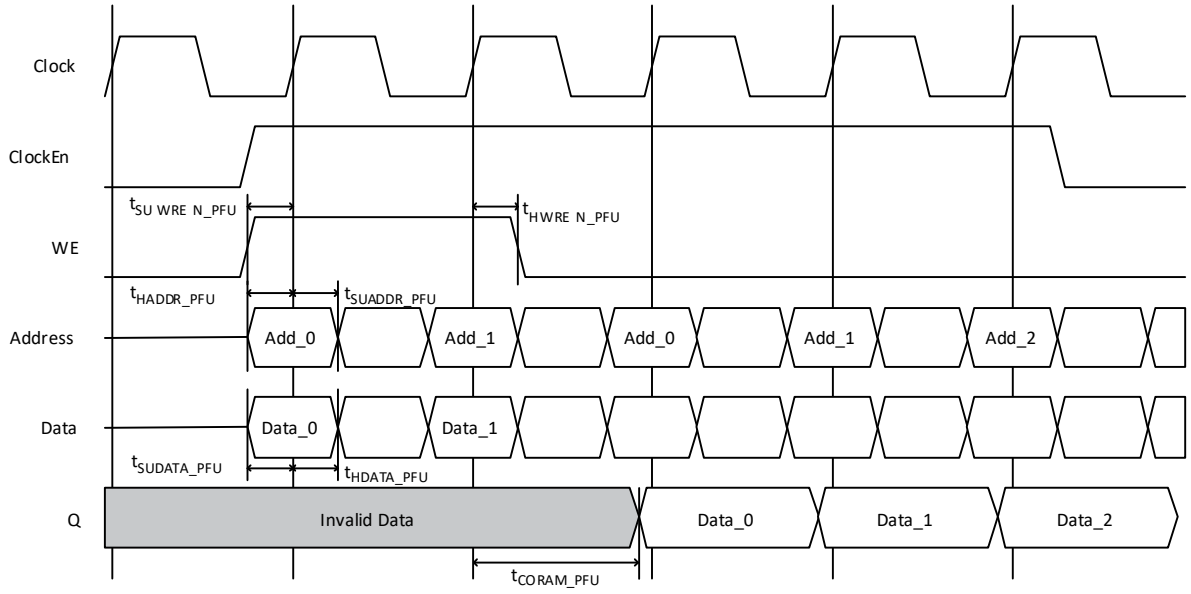

Figure 4.36. PFU Based Distributed Single Port RAM Timing Waveform – without Output Registers

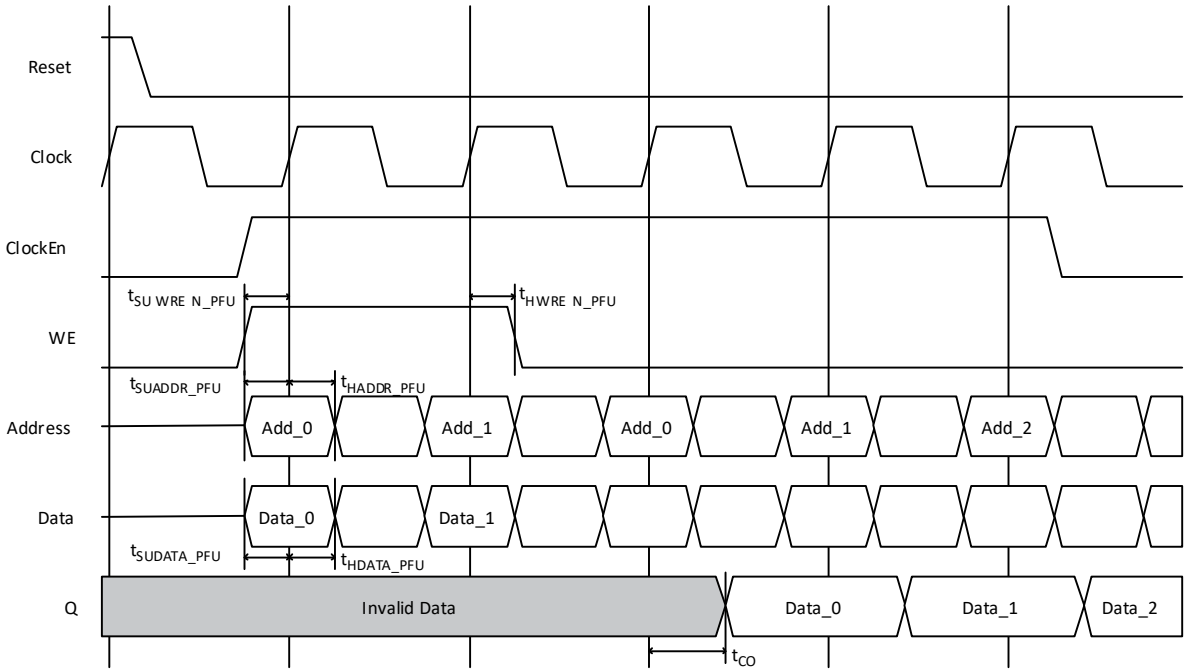

Figure 4.37. PFU Based Distributed Single Port RAM Timing Waveform – with Output Registers

4.7. Distributed Dual Port RAM (Distributed_DPRAM) – PFU Based

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

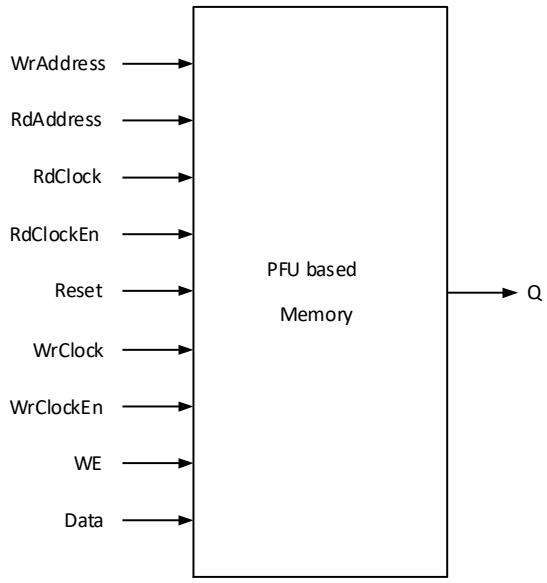

Figure 4.38. Distributed Dual Port RAM Module Generated by IPexpress

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

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

The various ports and their definitions for memory are as per Table 4.14. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 4.14. PFU-based Distributed Dual-Port RAM Port Definitions

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

Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when you want to enable the output registers in the IPexpress configuration. Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed_DPRAM). Figure 4.39 and Figure 4.40 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed_DPRAM) with these options.

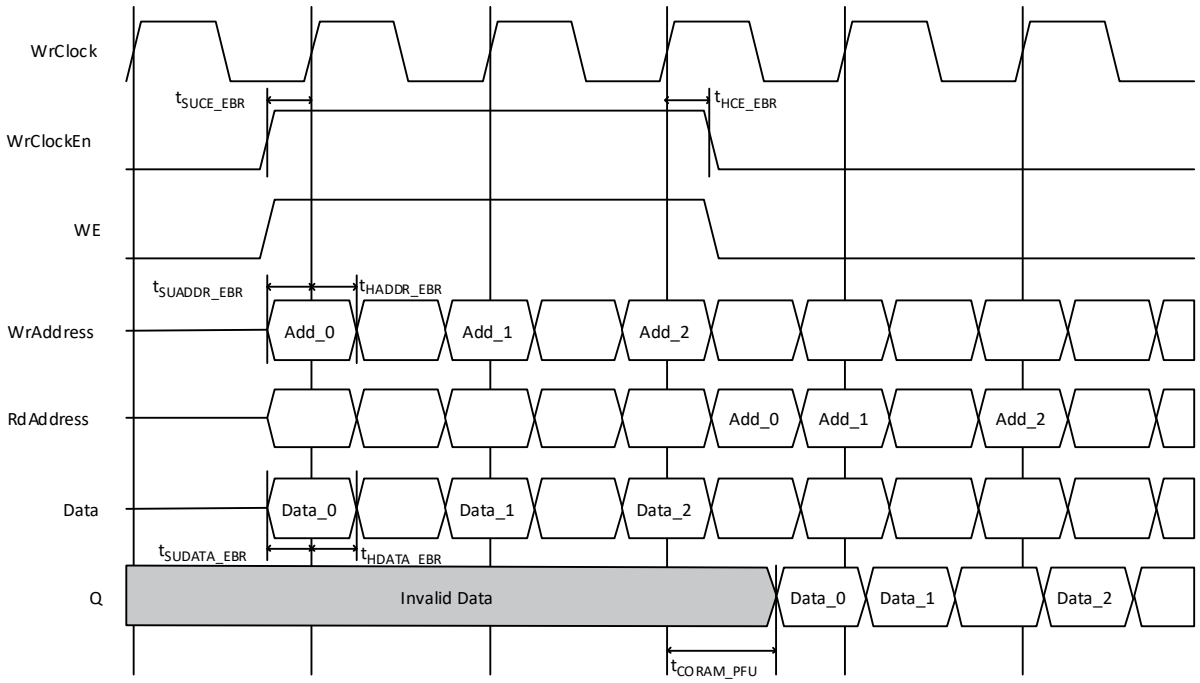

Figure 4.39. PFU Based Distributed Dual Port RAM Timing Waveform – without Output Registers

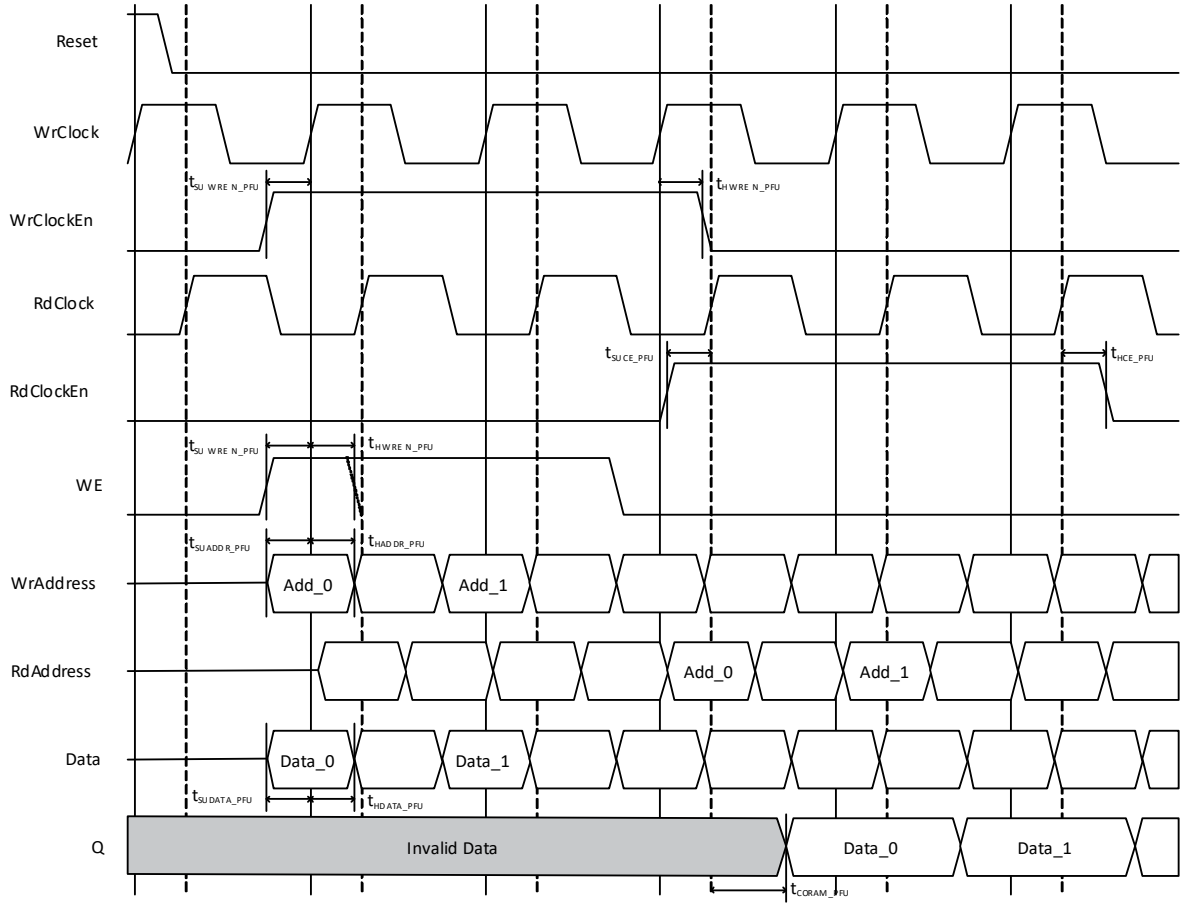

Figure 4.40. PFU Based Distributed Dual Port RAM Timing Waveform – with Output Registers

4.8. 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 4.41 shows the Distributed ROM module as generated by IPexpress.

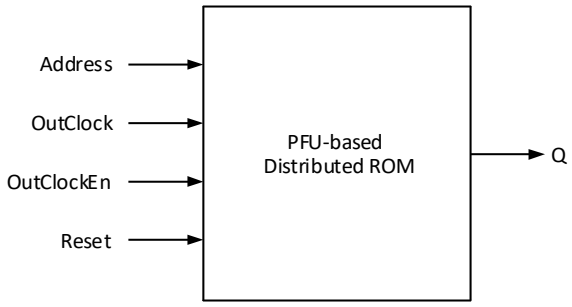

Figure 4.41. Distributed ROM Generated by IPexpress

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

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

The various ports and their definitions for memory are as per Table 4.15. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 4.15. PFU-based Distributed ROM Port Definitions

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

Users have the option to enable the output registers for Distributed ROM (Distributed_ROM). Figure 4.42 and Figure 4.43 show the internal timing waveforms for the Distributed ROM with these options.

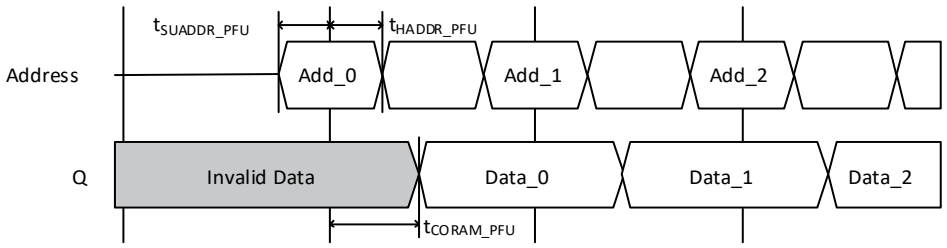

Figure 4.42. PFU Based ROM Timing Waveform – without Output Registers

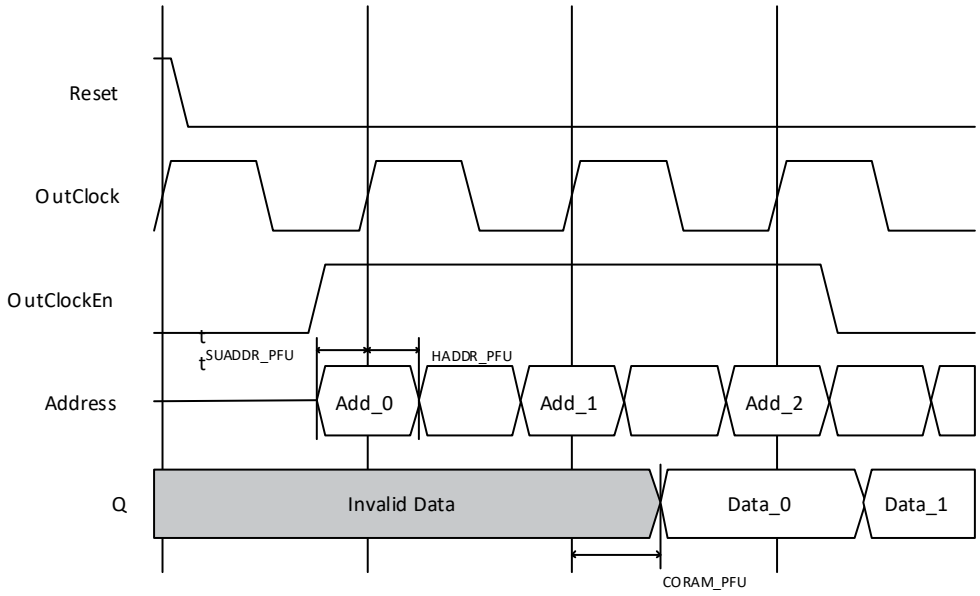

Figure 4.43. PFU Based ROM Timing Waveform – with Output Registers

4.9. User TAG Memory

The TAG memory is an area on the on-chip Flash which can be used for non-volatile storage. It is accessed in your design as if it were an external SPI Flash. Both SPI bus operation modes 0 (0,0) and 3 (1,1) are supported.

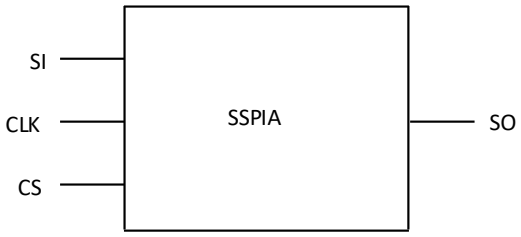

Figure 4.44. SSPIA Primitive

Table 4.16. User TAG Memory Signal Description

Primitive Port Name	Description
SI	Data input
SO	Data output
CLK	Clock
CS	Chip select

4.9.1. Basic Specifications for TAG Memory

There is one full page of TAG memory in each LatticeXP2 device. Page size ranges from 56 to 451 bytes.

Table 4.17. TAG Memory Density

Device	TAG Memory (Bits)	TAG Memory (Bytes)
XP2-5	632	79
XP2-8	768	96
XP2-17	2184	273
XP2-30	2640	330
XP2-40	3384	423

Table 4.18. Timing Specifications

Symbol	Parameter	Min	Max	Units
f_{MAXSPI}	Slave SPI CCLK Clock Frequency	—	25	MHz
t_{RF}	Clock and Data Input Rise and Fall Time	—	20	ns
t_{CSCCLK}	Slave SPI CCLK Clock High Time	20	—	ns
t_{SOCDO}	Slave SPI CCLK Clock Low Time	20	—	ns
t_{SCS}	CSSPIN High Time	25	—	ns
t_{SCSS}	CSSPIN Setup Time	25	—	ns
t_{SCSH}	CSSPIN Hold Time	25	—	ns
t_{STSU}	Slave SPI Data In Setup Time	5	—	ns
t_{STH}	Slave SPI Data In Hold Time	5	—	ns
t_{STVO}	Slave SPI Output Valid (after WRITE_EN)	—	20	ns
	Slave SPI Output Valid (without WRITE_EN)(1)	3	20	µs
t_{STCO}	Slave SPI Output Hold Time	0	—	ns
t_{SDIS}	Slave SPI Output Disable Time		100	ns

Note:
If the READ_TAG command is issued without first loading the WRITE_EN command, the device will need extra time, up to 20µs maximum, to transfer the data from TAG Flash to the data-shift register.

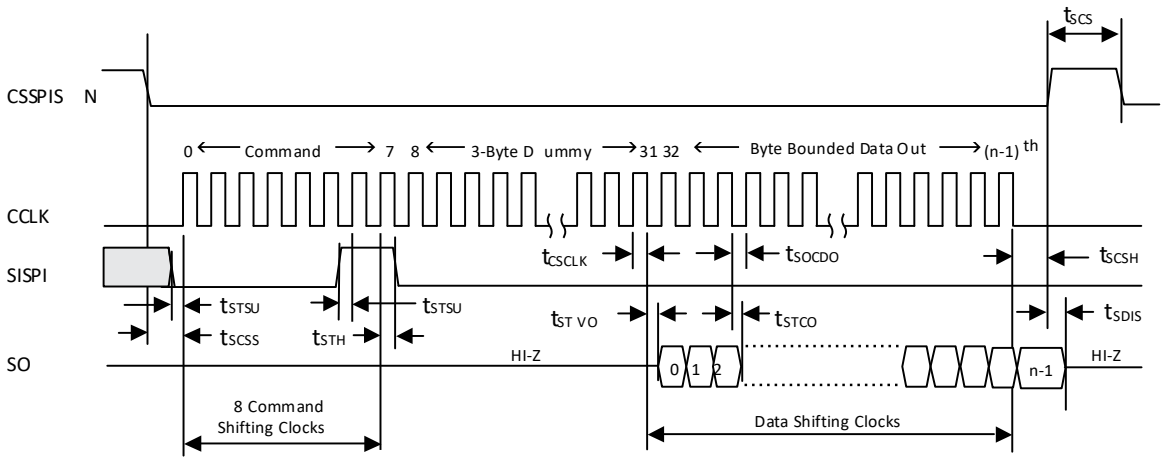

Figure 4.45. Generic Timing Diagram

4.10. Programming via the SPI Interface

Since the SSPIA module is an internal module, I/Os can be treated as I/Os of any other soft module. Therefore, they can be routed to other internal modules, or they can go out to I/O pads. The recommended routing is to the sysCONFIG port pins.

Table 4.19. Usage of Commands[3]

Command Name	OPCODE	Bytes 1-3[1, 2]	Data	Description
READ_TAG	0x4E	Dummy	Out	Read TAG memory
PROGRAM_TAG	0x8E	Dummy	In	Program TAG memory
ERASE_TAG	0x0E	Dummy	—	Erase TAG memory

Notes:

1. Data bytes are shifted with most significant bit first.
2. Byte 1-3 are dummy clocks to provide extra timing for the device to execute the command. The data presented at the SI pin during these dummy clocks can be any value and do not have to be 0x00 as shown.
3. Refer to the Flash Erase Time in the data sheet for delay times.

4.11. General Description

The LatticeXP2 family of devices is designed with instant-on, standalone TAG memory that is always available.

TAG memory is organized as a one-page Flash non-volatile memory accessible by the hardwired Serial Peripheral Interface port or the JTAG port.

The standalone TAG memory is ideal for use as scratch pad memory for critical data, board serialization, board revision logs and programmed pattern identification.

The integration of TAG memory into the LatticeXP2 device family saves chip count and board space. It also can be used to replace obsoleted low-density SPI EEPROM devices.

The hardwired SPI interface does not require the device to pre-program the configuration Flash first to enable the SPI interface. The interface is already enabled when the device is shipped from Lattice, saving board test time.

The hard-wired SPI interface allows the TAG memory to retain its independent identity or accessible always in spite of the TAG Memory Flash is embedded into the LatticeXP2 devices.

The hard-wired SPI interface is also important for field upgrades so that critical data can be maintained on the TAG memory and guaranteed to be accessible even if the device is field upgraded to a new pattern.

The instant-on capability is achieved by enabling the SPI interface when the devices are shipped from Lattice. Unlike the configuration Flash, the security setting of the device, standard or advanced, has no effect on the accessibility of the TAG memory. Therefore, the TAG memory is always accessible.

The TAG memory, same as other Flash fuses, can also be programmed using the IEEE 1532 compliant programming flow on the JTAG port for production programming support or for system debugging.

4.12. Pin Descriptions

The pins described below are not dedicated pins. If the TAG memory feature is not required, these pins can be regular user I/O pins. If the TAG memory feature is required, the TAG memory can be accessed by the internal SPI interface through the core. The internal SPI interface makes the TAG Memory capable of supporting advanced applications. For example:

1. Use an I^2C to SPI translator to convert the SPI TAG memory to be an I^2C TAG memory device.
2. Route the four SPI interfaces to the other four user I/Os.

Selections are made using the ispLEVER design tool. By default, the external SPI interface is enabled and TAG memory is selected.

The functional descriptions of the pins below are applicable to both the internal and external SPI interfaces.

4.12.1. Serial Data Input (SI)

The SPI Serial Data Input pin provides a means for commands and data to be serially written to (shifted into) the device. Data is latched on the rising edge of the serial clock (CLK) input pin.

4.12.2. Serial Data Output (SO)

The SPI Serial Data Output pin provides a means for status and data to be serially read out (shifted out of) the device. Data is shifted out on the falling edge of the serial clock (CLK) input pin.

4.12.3. Serial Clock (CLK)

The SPI Serial Clock Input pin provides the timing for serial input and output operations.

4.12.4. Chip Select (CS)

The SPI Chip Select pin enables and disables SPI interface operations. When the Chip Select is high the SPI interface is deselected and the Serial Data Output (SO) pin is at high impedance. When it is brought low, the SPI interface is selected and commands can be written into and data read from the device. After power up, CS must transit from high to low before a new command can be accepted.

4.13. SPI Operations

4.13.1. SPI Modes

The SPI interface is accessible through the SPI-compatible bus consisting of four signals: Serial Clock (CLK), Chip Select (CS), Serial Data Input (SI) and Serial Data Output (SO). Both SPI bus operation Modes 0 (0,0) and 3 (1,1) are supported. The primary difference between Mode 0 and Mode 3 concerns the normal state of the CLK pin when the SPI master is in standby and data is not being transferred to the device's SPI interface. For Mode 0 the CLK is normally low and for Mode 3 the CLK signal is normally high. In either case, data input on the SI pin is sampled only during the rising edge. Data output on the SO pin is clocked out only on the falling edge of CLK.

4.13.2. Status Register

The SPI interface can access the 1-bit status register required to support TAG Memory Flash programming.

The programming complete status register: This is the single bit status register for pooling. If the programming or erase operation is complete, then the status bit is set to 1, otherwise it is set to 0 for more programming or erase time.

4.13.3. Commands

Table 4.20. Commands

Command Name	Byte 1 (Opcode)	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	n-Byte
READ_ID	0x98	0x00	0x00	0x00	(D0-D7)	(D8-D15)	(D24-D31)
WRITE_EN	0xAC	0x00	0x00	0x00	—	—	—
WRITE_DIS	0x78	0x00	0x00	0x00	—	—	—
ERASE_TAG	0x0E	0x00	0x00	0x00	—	—	—
PROGRAM_TAG	0x8E	0x00	0x00	0x00	D7-D0	Next Byte	Last Byte
READ_TAG	0x4E	0x00	0x00	0x00	(D7-D0)	(Next Byte)	(Continue)
STATUS	0x4A	0x00	0x00	0x00	(b1xxxxxxx or b0xxxxxxx)	—	—

Notes:

1. Data bytes are shifted with least significant bit first. Byte field with data in parenthesis ( ) indicate data being read from the SO pin.
2. Byte 2-4 are dummy clocks to provide extra timing for the device to execute the command. The data presented at the SI pin during these dummy clocks can be any value and do not have to be 0x00 as shown.

3. The READ_ID command reads out the 32 bits JTAG IDCODE of the device. The first bit shifted out on SO pin is thus bit 0 of the JTAG IDCODE and the last bit is bit 31 of the IDCODE.
4. The PROGRAM_TAG command supports page programming only. The programming data shift into the TAG Memory must be exactly the same size as the one page of the TAG Memory. Under shifting or over shifting will cause erroneous data programmed into the TAG Memory. The Last Byte shown on the n-Byte column indicates the last byte of the data must be shifted into the device before driving the Chip Select to high to start the programming action.
5. The STATUS command read from the single bit status register. When read from the register, only the first bit is valid, the other bits are dummies and should be ignored.

4.13.3.1. READ_ID (98h)

The READ_ID command captures the IEEE 1149.1 JTAG IDCODE out of the device on the SO pin. This command is commonly used to verify whether communication is established with the SPI bus. After the 8-bit READ_ID command is received, the device ignores the data presented at the Serial Data Input (SI) pin. The Serial Output (SO) pin is enabled on the falling edge of clock 31 to drive out the first bit of the IDCODE. After 32 bits of the IDCODE are shifted out, additional clocking will cause dummy data to be shifted out on SO.

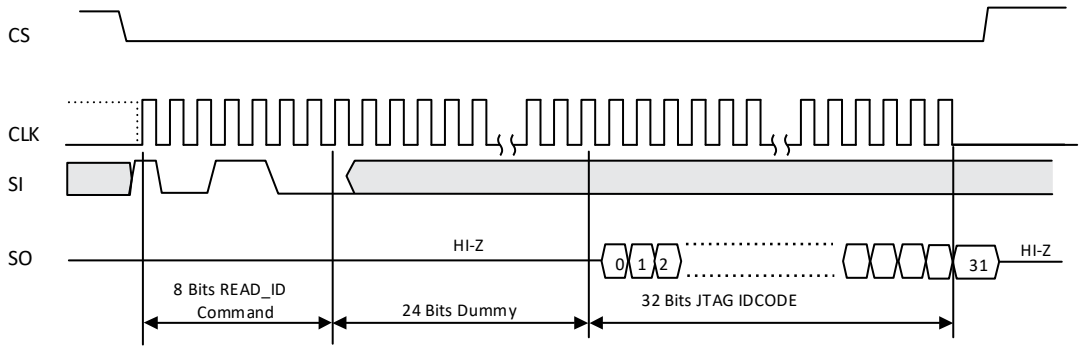

Figure 4.46. READ_ID Waveform

4.13.3.2. WRITE_EN (ACh)

The WRITE_EN command enables the TAG memory for programming. If the WRITE_EN command has not been shifted into the device first, the PROGRAM_TAG, ERASE_TAG and STATUS commands do not take effect. This is to prevent the TAG memory from erroneous erase or program.

The command is executed when the Chip Select pin is driven from low to high after the 24th dummy clock. Any extra dummy clocks, if presented before driving the Chip Select pin to high, are ignored. After the Chip Select pin is driven from low to high, a minimum of three clocks are required to complete the execution of the command.

The effect of this command is terminated by the WRITE_DIS command.

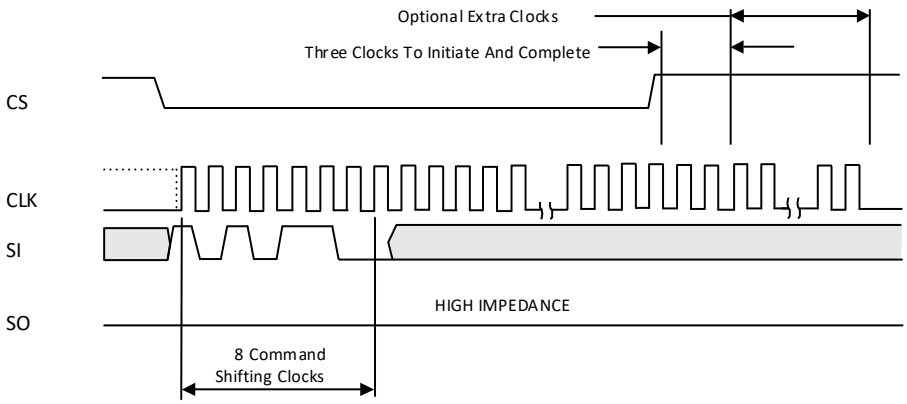

Figure 4.47. WRITE_EN Waveform

4.13.3.3. WRITE_DIS (78h)

The WRITE_DIS command disables the TAG memory for programming. It does not nullify the READ_TAG and READ_ID commands.

The command is executed when the Chip Select pin is driven from low to high after the 24th dummy clock. Any extra dummy clocks, if presented before driving the Chip Select pin to high, are ignored. After the Chip Select pin is driven from low to high, a minimum of three clocks are required to complete the execution of the command.

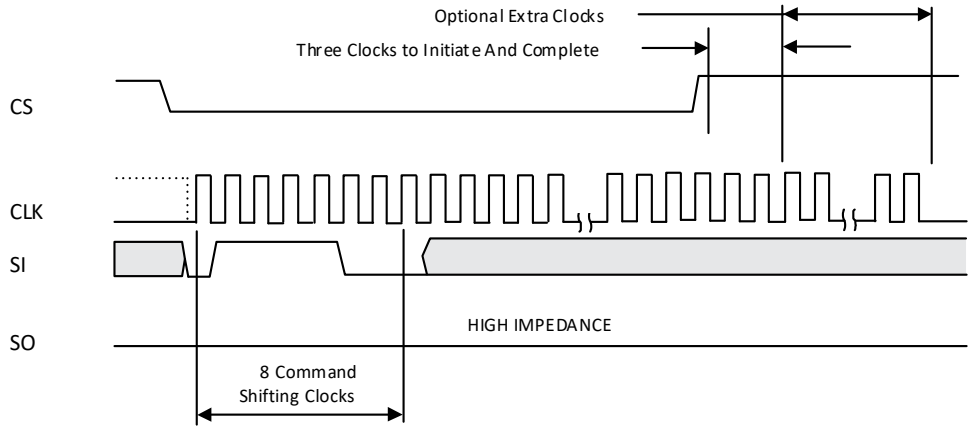

Figure 4.48. WRITE_DIS Waveform

4.13.3.4. ERASE_TAG (0Eh)

The ERASE_TAG command is enabled after the command WRITE_EN has been shifted into the device and executed. The ERASE_TAG command erases all the TAG Memory Flash cells.

The command is executed when the Chip Select pin is driven from low to high after the 24th dummy clock. Any extra dummy clocks, if presented before driving the Chip Select pin to high, are ignored. After the Chip Select pin is driven from low to high, a minimum of three clocks is required to initiate the erase action. After the three clocks, extra clocks are optional. Once the erase action is initiated, it is carried out until it is done. There is no mechanism to terminate the erase action.

This command sets the STATUS bit to 0 when the erase action begins. The programming engine sets the status bit to 1 when the erase is done successfully.

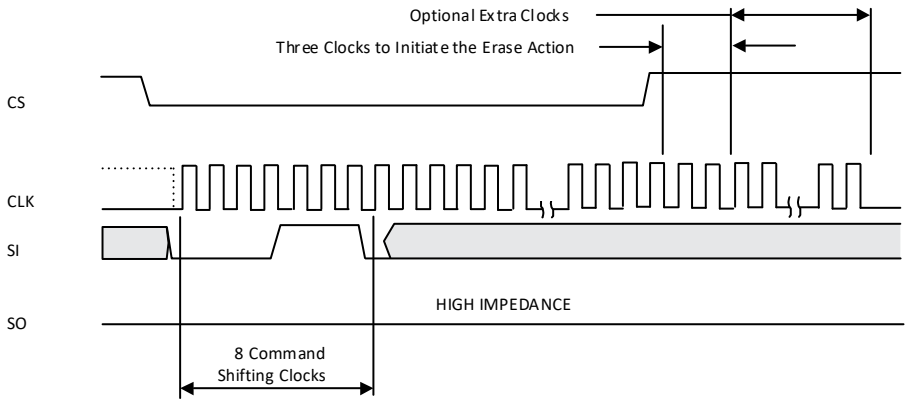

Figure 4.49. ERASE_TAG Waveform

4.13.3.5. PROGRAM_TAG (8Eh)

The PROGRAM_TAG is enabled after the command WRITE_EN has been shifted into the device and executed. The PROGRAM_TAG command programs the entire TAG memory page at once.

After the command is shifted into the device on the SI pin, and followed by 24 dummy clocks, the TAG memory column decoder serves as the data buffer for the programming data to be shifted into serially. The shifting direction is from left to right, as shown. The first bit to be shifted out closest to the SO pin appears on the right-most side of the shift register. The data buffer functions like a FIFO (First In First Out) serial data shift register. In order to make sure that bit 0 is read out first, data bit 0 must be shifted into the right-most location of the data shift register. To achieve this, the data buffer must be filled up completely. Consequently, over-filling the data buffer will cause overflow of the data buffer, resulting in loss of data.

The SO pin stays in the HIGHZ state throughout this command.

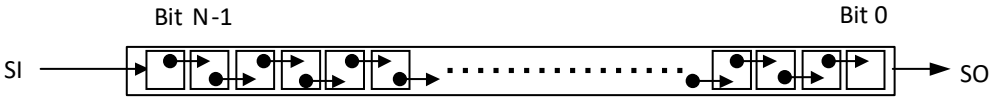

Figure 4.50. Bit Shifting Order

When the data buffer has filled up to one page of data, driving the Chip Select pin to high terminates the data shifting. After the Chip Select pin is driven from low to high, a minimum of three clocks are required to initiate the programming action. In the programming action, the data buffer content is copied in parallel from the data buffer into the TAG memory Flash cells. The status bit is set to 0 when the programming actions begin. The status bit is set to 1 when the programming action is done successfully.

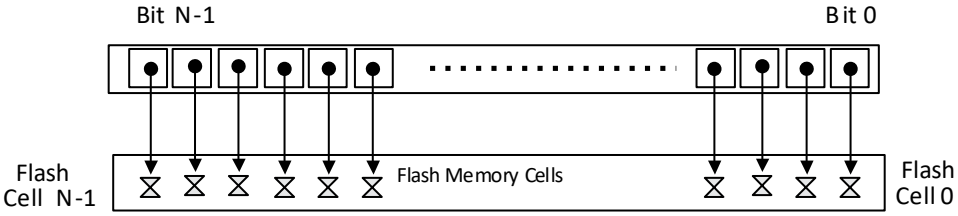

Figure 4.51. Data Buffer to Flash Cell Mapping

When the programming action complete, the STATUS bit is set to 1. The exact same image is written into the Flash cells of the TAG Memory block when the programming action complete.

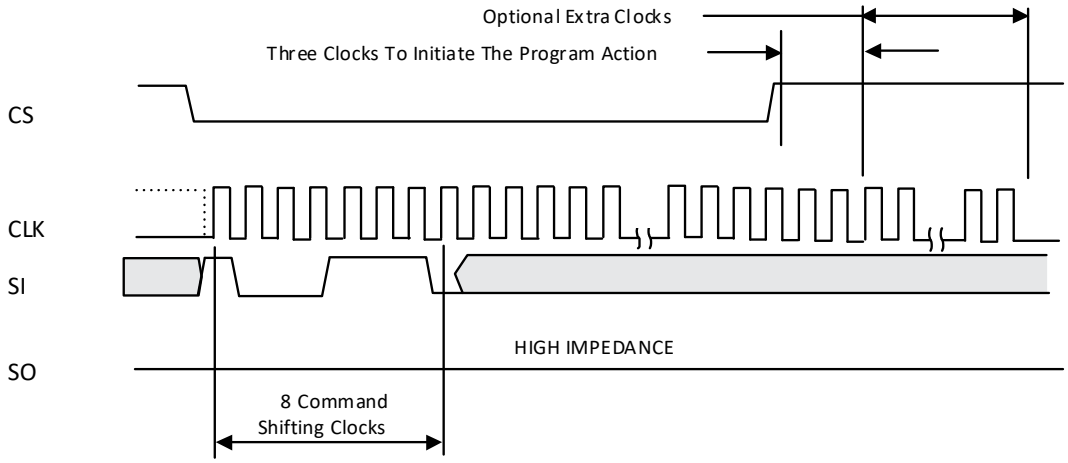

Figure 4.52. PROGRAM_TAG Waveform

4.13.3.6. READ_TAG (4Eh)

The READ_TAG command enables the Flash programming engine to transfer data programmed in the Flash cells to the data buffer. The transfer action starts on the third dummy clock after the 8-bit opcode. The delay time derived from the 21 dummy clocks is the time required to transfer the Flash cells data into the data buffer. The transfer action, once initiated, does not need the clock to continue to run. The clock count is only required to enable the SO pin. If the Flash circuitry is not yet enabled, the device needs extra delay time to enable the Flash circuitry before transfer can take place. This extra delay must be provided after the third dummy clock and before the 24th dummy clock.

If the READ_TAG command is preceded by the WRITE _EN command, then it is fast read. The device does not require extra delay to enable the Flash circuitry.

The 20 dummy clocks after the transfer is initiated to before enabling the SO pin are considered delay clocks.

Delay time = 20 x 1/frequency.

The transfer delay time, including the extra delay time to enable the Flash circuitry, is 5uS minimum. The clock frequency can then be set to 2.5 MHz if continuous clocking is desired.

When all the data captured into the data buffer are shifted out, additional clocks will shift out dummy data. The SI pin is not connected to the input of the data buffer when the READ_TAG command is shifted into the device. While the data in the data buffer is shifted out to the direction of SO, dummy data is shifted into the data buffer. Consequently, when over-shifting occurs, dummy data of unknown value is shifted out.

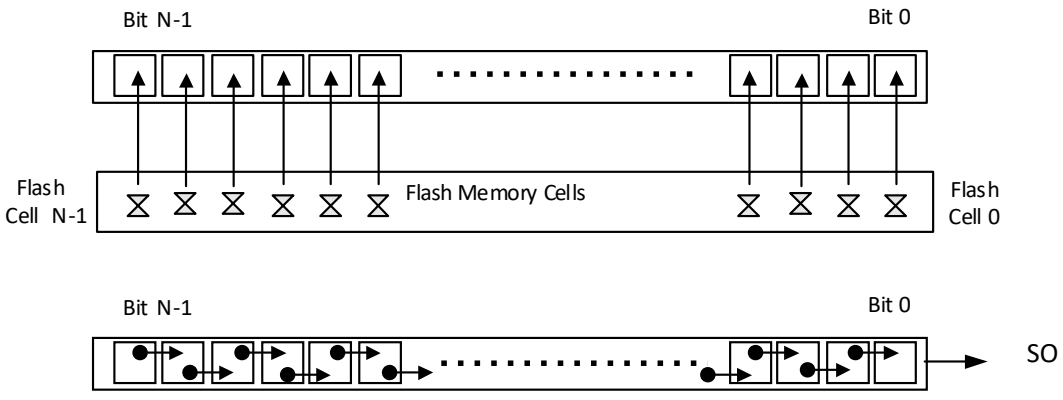

Figure 4.53. Readout Order

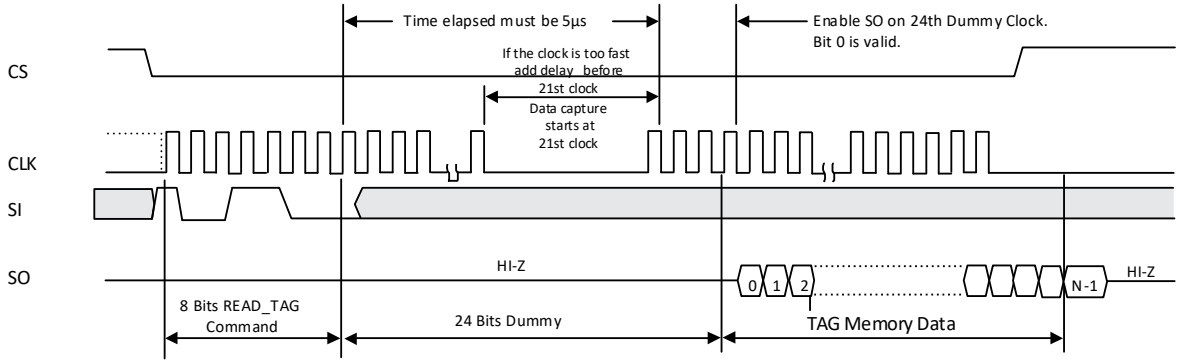

Figure 4.54. READ_TAG Waveform

4.13.3.7. STATUS (4Ah)

The STATUS command allows the single-bit status register to be read. This command can be loaded at any time after the WRITE_EN command has already been shifted into the device first. This command does not terminate the programming or erase action. It is used to report the progress of the programming or erase action.

The status register actual size is only 1 bit. Dummy data is shifted out on the SO pin if extra data shifting clocks are applied. The command can be shifted into the device again to capture the status bit and then read out.

During the interval of shifting the command, the additional programming or erase time is provided by driving the Chip Select pin to high and holding the CLK pin low. Clocking while holding the Chip Select pin high is optional.

If the maximum programming time or erasure has expired and the status bit still is not set to 1, then erase or programming has failed.

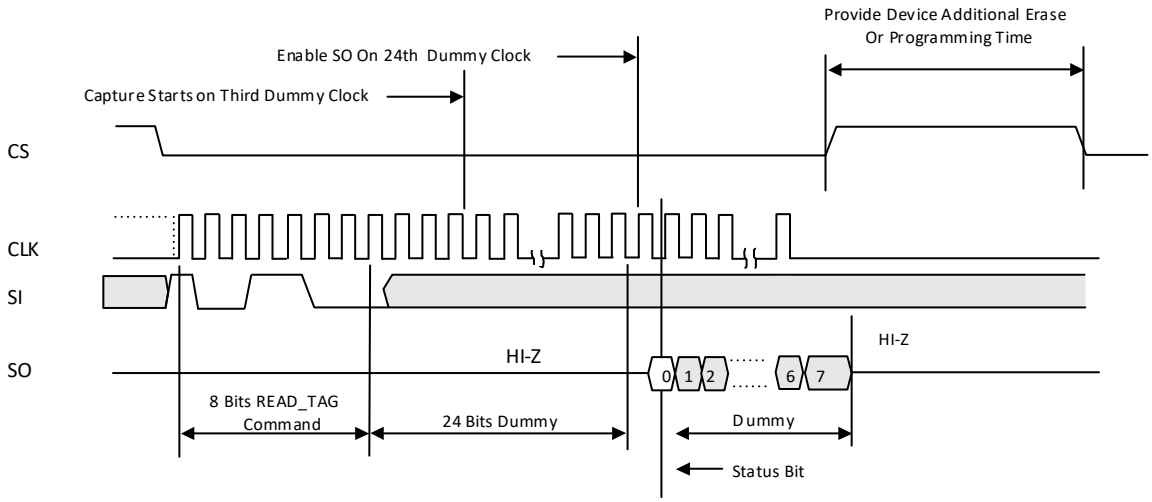

Figure 4.55. STATUS Waveform

4.14. Specifications and Timing Diagrams

4.14.1. Powering Up

TAG memory is available when the boot-up either from the internal embedded Flash or from the external SPI Flash boot PROM is complete. If the embedded Flash is blank, the boot up will not work. It is recommended to wait for the same amount of delay as if the embedded configuration has been programmed before accessing the TAG memory. If the boot-up is from external SPI Flash, longer delay time should be given or check the DONE pin for a high first before accessing the TAG memory.

The SPI interface needs the low-to-high transition on the Chip Select pin to reset. During power-up, the low-to-high transition is assured by requiring the CLK pin tracking the VCC. The other method is to drive the Chip Select pin to high then low then high to reset the SPI interface before shifting the first command into the device.

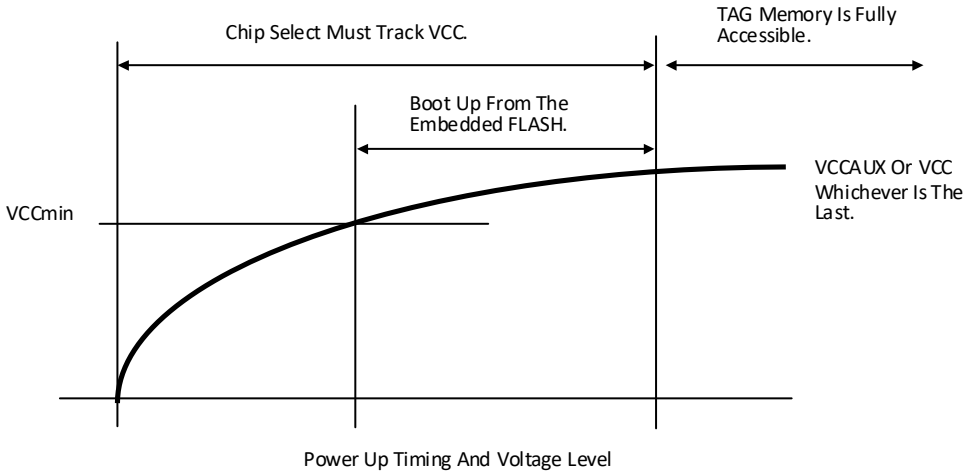

Figure 4.56. Device Power-up Waveform

4.14.2. Availability of TAG Memory

TAG memory is available most of time on the Slave SPI interface with the following exceptions:

- When the SRAM fuses are being accessed by the JTAG port, Slave SPI interface or refreshing
- When the other Flash cells are being accessed through the JTAG port or Slave SPI interface
- While JTAG BSCAN testing is taking place
- The Slave SPI interface is disabled with the persistent fuse programmed (set to off)

4.14.3. AC Timing

- 25 MHz maximum CLK
- 5 uS minimum read command delay
- 2 mS minimum delay from VCCmin to shifting in the first command

4.14.4. Programming Timing

- 1 sec. maximum erase time
- 5 mS maximum programming time

4.15. Programming via the JTAG Interface

.VME files can be generated for the ispVM System software which only programs the TAG memory. These .VME files are handled according to the standard ispVME flow.

5. 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.

5.1. Initialization File Format

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

- Binary file
- Hex File
- Addressed Hex

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

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

The TAG memory uses hex or binary non-addressed files. Since it is a SPI, it cannot use the addressed hex file.

5.2. 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
```

5.3. 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
```

5.4. Addressed Hex

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

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

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

There is no limitation on the values of address and data. The value range is automatically checked based on the values of addr_width and data_width. If there is an error in an address or data value, an error message is printed.

Users need not specify data at all address locations. If data is not specified at certain address, the data at that location is initialized to 0. IPexpress makes memory initialization possible both through the synthesis and simulation flows.

5.5. FlashBak™ Capability

The LatticeXP2 FPGA family offers FlashBak capability, which is a way to store the data in the EBRs to the Flash memory upon user command. This protects the user's data from being lost when the system is powered off. The FlashBak module (STFA primitive) has a single-command-two-operation process (see Figure 65). When the FlashBak operation is initiated, an erase-UFM-Flash signal is enabled to erase the Flash, followed by the transfer-to- flash operation. Once the transfer is done, the Flash controller sends a transfer-done signal back to the user logic. During the FlashBak operation, the EBRs are not accessible. There is no difference between the regular EBR RAM configuration and the shadow Flash (UFM) EBR RAM configuration in the ispLEVER GUI. The presence of the STFA (FlashBak) primitive in a design determines the EBR RAM configuration. FlashBak cannot be used if the soft-error detect (SED) is operating in an Always mode. Since there is no addressing but just a 'dump' of all EBR to Flash, only one STFA module is necessary. Multiple modules are not necessary or allowed.

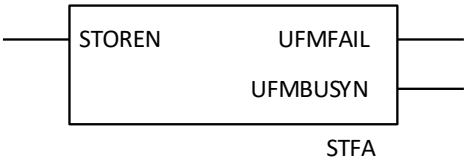

Figure 5.1. FlashBak Primitive

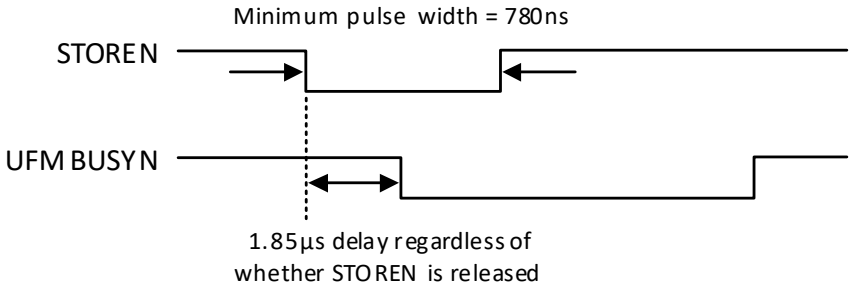

Figure 5.2. FlashBak Waveform

Table 5.1. STFA Port Descriptions

Port Name	Corresponding Hardware Port Name	I/O	Description
STOREN	storecmdn	I	Initiates to store the EBR content to Flash
UFMFAIL	ufm_fail	O	Store to Flash operation failed
UFMBUSYN	fl_busyn	O	Tells the user whether the Flash is in the busy state or not

Appendix A. Attribute Definitions

A.1. 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 takes the values defined in the RAM size tables in each memory module.

A.2. 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.

A.3. 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.

A.4. CSDECODE

CSDECODE, or the Chip Select Decode attributes, are associated to block RAM elements. 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. 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.

A.5. 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 and x18 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 an 'H' and the other tries to write an 'L'.

It is recommended that users implement control logic to identify this situation if it occurs and then either:

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

A.6. GSR

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

A.7. ASYNC_RESET_RELEASE

When RESETMODE is set to ASYNC, the ASYNC_RESET_RELEASE attribute allows users to select how the reset is de-asserted/released: When set to SYNC, the reset is de-asserted synchronously to the clock. When set to ASYNC, the memory reset is released asynchronously (without relation to the clock).

A.8. INIT_DATA (MachXO3LF Only)

The INIT_DATA attribute allows the user to specify how EBR initialization values are stored and accessed. When set to STATIC, the EBR initialization values are compressed by the software and stored in a variable location in UFM (User Flash Memory). When set to DYNAMIC, the initialization values are not compressed, and stored in a user-accessible, fixed location in UFM.

Technical Support Assistance

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

Revision History

Revision 2.3, March 2021

Section	Change Summary
Utilizing IPexpress	• Newly added the Byte Order with Different Port Widths, ECC in Memory Modules, Utilizing PMI, and Memory Module Inference sections.
Memory Modules	• Removed the original ECC support description.

Revision 2.2, October 2018

Section	Change Summary
All	Changed document number from TN1137 to FPGA-UG-02080.
All	Changed to the latest template.
Acronyms in This Document	Newly added.
Table 4.20	Changed “most” to “least” in the notes.

Revision 2.1, June 2015

Section	Change Summary
Programming via the SPI Interface	• Removed Delay Time column in Table 20, Usage of Commands. • Added footnote 3.
Technical Support Assistance	Updated.

Revision 02.0, August 2013

Section	Change Summary

All	• Updated corporate logo.
	• Updated the following waveform figures:
	  • Generic Timing Diagram
	  • READ_ID Waveform
	  • WRITE_EN Waveform
	  • WRITE_DIS Waveform
	  • ERASE_TAG Waveform
	  • PROGRAM_TAG Waveform
	  • READ_TAG Waveform
	  • STATUS Waveform

Serial Data Input (SI)	Updated.
Technical Support Assistance	Updated.

Revision 01.9, July 2011

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

Revision 01.8, November 2008

Section	Change Summary
All	Updated the following waveform figures: • Generic Timing Diagram • READ_ID Waveform • WRITE_EN Waveform • WRITE_DIS Waveform • ERASE_TAT Waveform

Section	Change Summary
	• PROGRAM_TAW Waveform • READ_TAG Waveform • STATUS Waveform.
Serial Data Input (SI)	Updated.
Memory Modules	Updated.

Revision 01.7, June 2008

Section	Change Summary
All	Added TAG memory timing waveforms and instructions.

Revision 01.6, June 2008

Section	Change Summary
All	Removed Read-Before-Write sysMEM EBR mode.
First In First Out (FIFO, FIFO_DC) – EBR Based	Updated

Revision 01.5, July 2008

Section	Change Summary
All	Added FlashBAK waveform diagram.

Revision 01.4, February 2008

Section	Change Summary
All	Updated FIFO_DC without Output Registers (Non-Pipelined) diagram.

Revision 01.3, January 2008

Section	Change Summary
All	• Updated Read_Tag Commands Waveform diagram. • Changed minimum delay between the 3rd and 24th dummy clock from 3µs to 5µs.

Revision 01.2, November 2007

Section	Change Summary
All	TAG memory added.

Revision 01.1, July 2007

Section	Change Summary
All	Added FlashBak Capability section.

Revision 01.0, February 2007

Section	Change Summary
All	Initial release

www.latticesemi.com