MachXO4 Memory User Guide

Technical Note

FPGA-TN-02402-1.0

December 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 REDUNDANCIES, 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.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
CS	Chip Select
EBR	Embedded Block RAM
ECC	Error Correction Code
EF	Empty Flag
FIFO	First In First Out
GSR	Global Set or Reset
HDL	Hardware Description Language
I2C	Inter-Integrated Circuit
IP	Intellectual Property
JTAG	Joint Test Action Group
LDC	Lattice Design Constraints
LUT	Look-Up Table
LSB	Least Significant Bit
MSB	Most Significant Bit
PFU	Programmable Functional Unit
PIC	Programmable I/O Cells
PLD	Programmable Logic Device
PMI	Parameterizable Module Instantiation
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
ROM	Read-Only Memory
RP	Read Pointer
SPI	Serial Peripheral Interface
UFM	User Flash Memory
VHDL	VHSIC Hardware Description Language

1. Introduction

This technical note discusses the memory usage for the Lattice MachXO4™ FPGA family. It is intended to be used by design engineers as a guide in integrating the EBR and PFU based memories for these devices in the Lattice Radiant™ software.

The architecture of these devices provides resources for memory intensive applications. The sysMEM™ Embedded Block RAM (EBR) complements the distributed Programmable Functional Unit (PFU)-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM, FIFO, and ROM memories can be constructed using the EBR. LUTs and PFU can implement Distributed Single-Port RAM, Dual-Port RAM, and ROM.

The capabilities of the EBR Block 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:

- Through IP Catalog – The IP Catalog interface allows you to specify the memory type and size that is required. IP Catalog takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.
- Through the Parameterizable Module Instantiation (PMI) – PMI allows experienced users to skip the graphical interface and utilize the configurable memory modules on the fly from the Radiant Source Template. 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.

In addition to familiar Block RAM and PFU RAM primitives, MachXO4 devices provide a new User Flash Memory (UFM) block, which can be used for a variety of applications including storing a portion of the configuration image, storing and initializing EBR data, storing PROM data or as a general purpose non-volatile User Flash Memory. The UFM block connects to the device core through the embedded function block WISHBONE interface. You can also access the UFM block through the JTAG, I2C and SPI interfaces of the device. The UFM block offers the following features:

- Non-volatile storage up to 448 kB
- Byte addressable for read access. Write access is performed in 128-byte pages.
- Program, erase, and busy signals
- Auto-increment addressing
- WISHBONE interface
- External access is provided through JTAG, I2C, and SPI interfaces

For more information on the UFM, refer to MachXO4 Hardened Control Functions User Guide (FPGA-TN-02403).

The remainder of this document discusses these approaches, utilizing IP Catalog, PMI inference, memory modules, and memory primitives.

2. Memories in MachXO4 Devices

All MachXO4 devices contain an array of logic blocks called PFUs surrounded by Programmable I/O Cells (PICs). This is shown in Figure 2.1 and Figure 2.2.

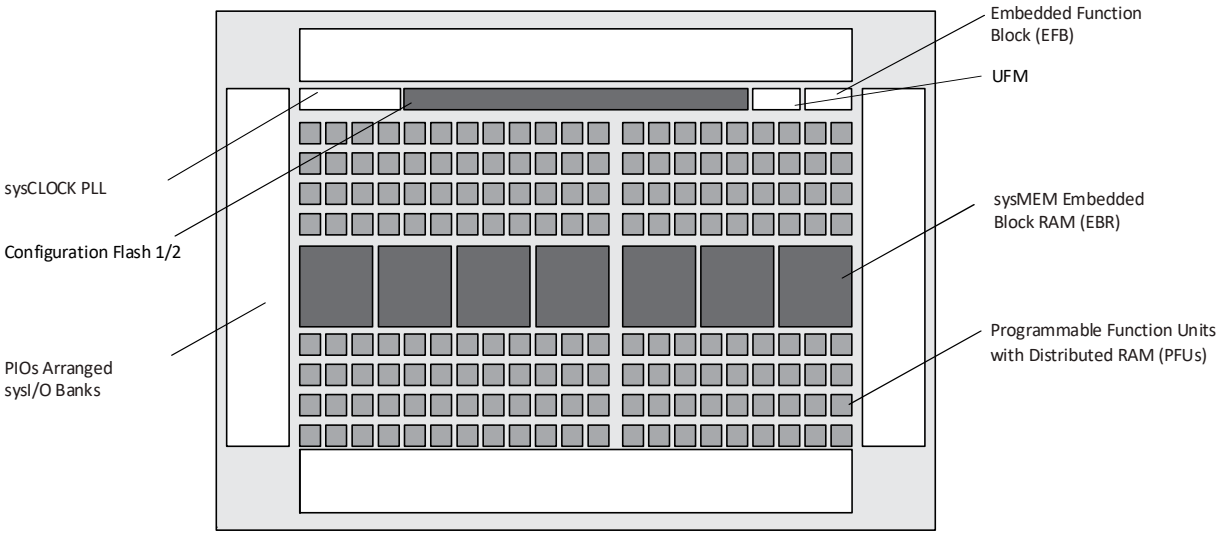

Figure 2.1. Top View of the MachXO4 LFMXO4-010 Device

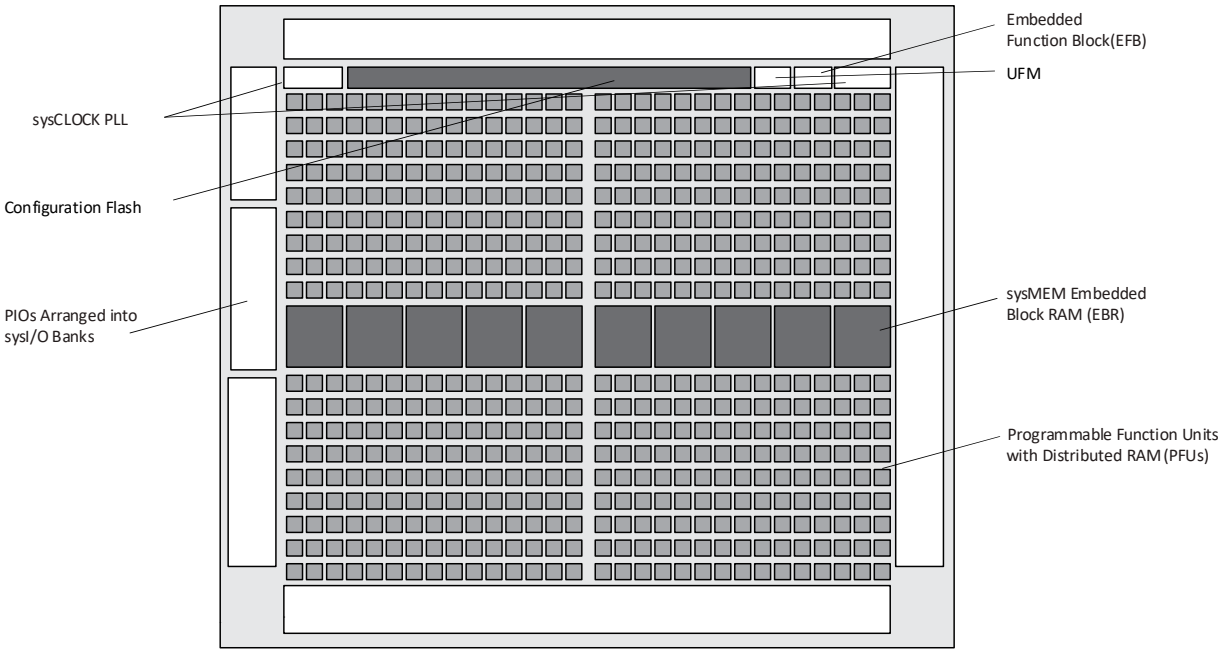

Figure 2.2. Top View of the MachXO4 LFMXO4-050 Device

The PFU contains the building blocks for logic and Distributed RAM and ROM. Some PFUs provide the logic building blocks without the distributed RAM. This document describes the memory usage and implementation for both Embedded Memory Blocks (EBRs) and Distributed RAM of the PFU. Refer to MachXO4 Family Data Sheet (FPGA-DS-02125) for details on the hardware implementation of the EBR and Distributed RAM.

3. Utilizing IP Catalog

You can utilize IP Catalog 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 (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-based
- RAM-based Shift Register – PFU-based

3.1. IP Catalog Flow

For generating any of these memories, create or open a project for the MachXO4 devices.

In the Lattice Radiant software, select **Views** > **Show Views** > **IP Catalog**. Alternatively, you can also click on the IP Catalog icon in the toolbar when MachXO4 devices are targeted in the project.

This opens the **IP Catalog** window, as shown in Figure 3.1.

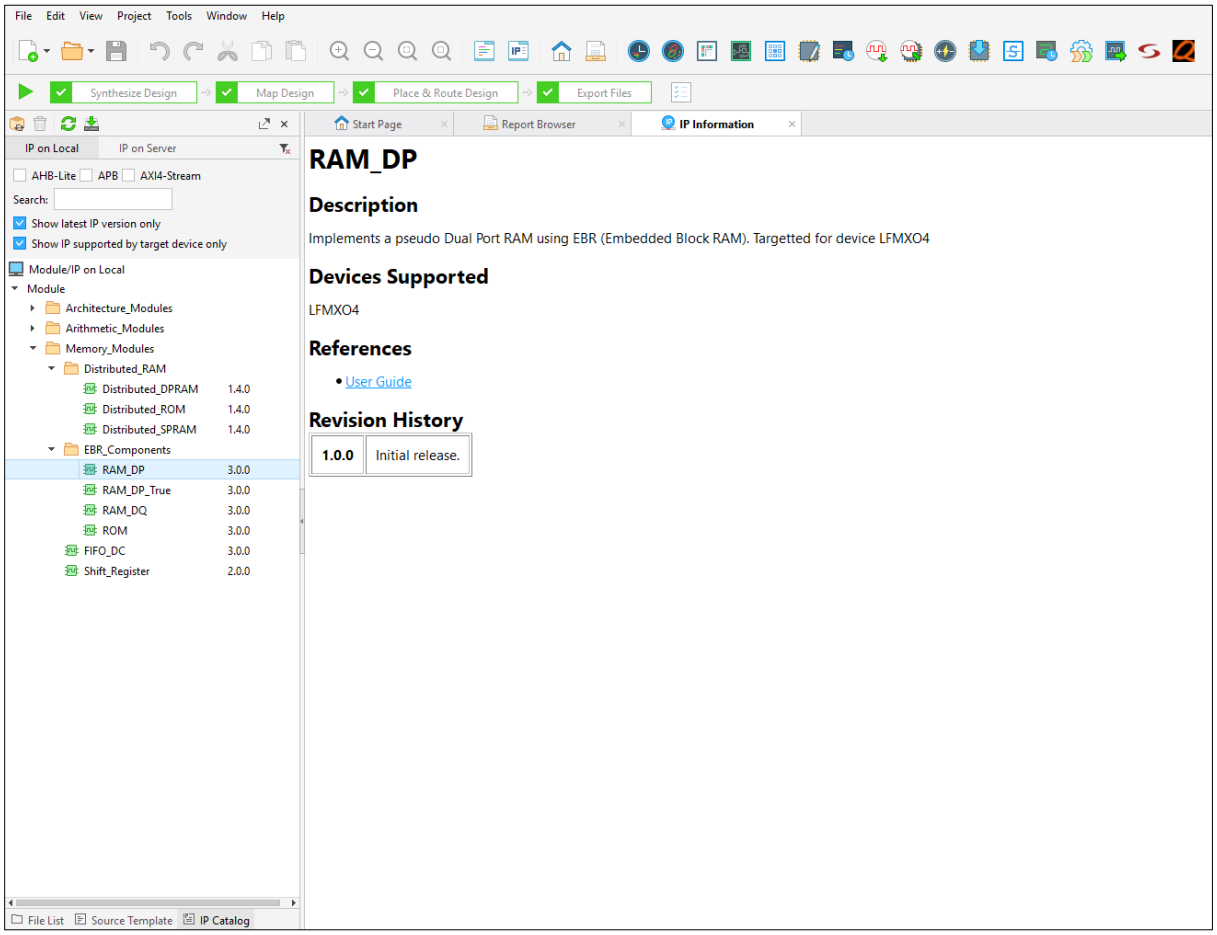

Figure 3.1. IP Catalog – Main Window

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 Distributed_RAM, as shown in Figure 3.1.

As an example, consider generating an EBR-based Pseudo Dual-Port RAM of size 512x16. Select RAM_DP under the EBR_Components. Double click the EBR_Components type generates the pop-up window shown in Figure 3.2.

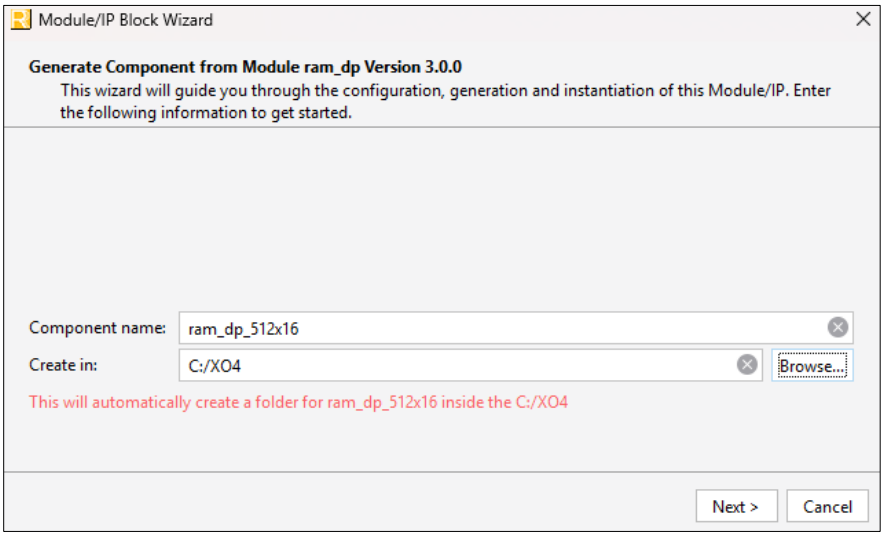

Figure 3.2. Generating a Pseudo Dual-Port RAM (RAM_DP) Using IP Catalog

You can change the directory where the generated modules are placed by clicking the **Browse** button.

The Component name text box allows you to specify an entity name for the module to be generated. You must provide this entity name.

After selecting the directory and putting in the entity name, click **Next**.

This opens another window where you can customize the RAM (Figure 3.3).

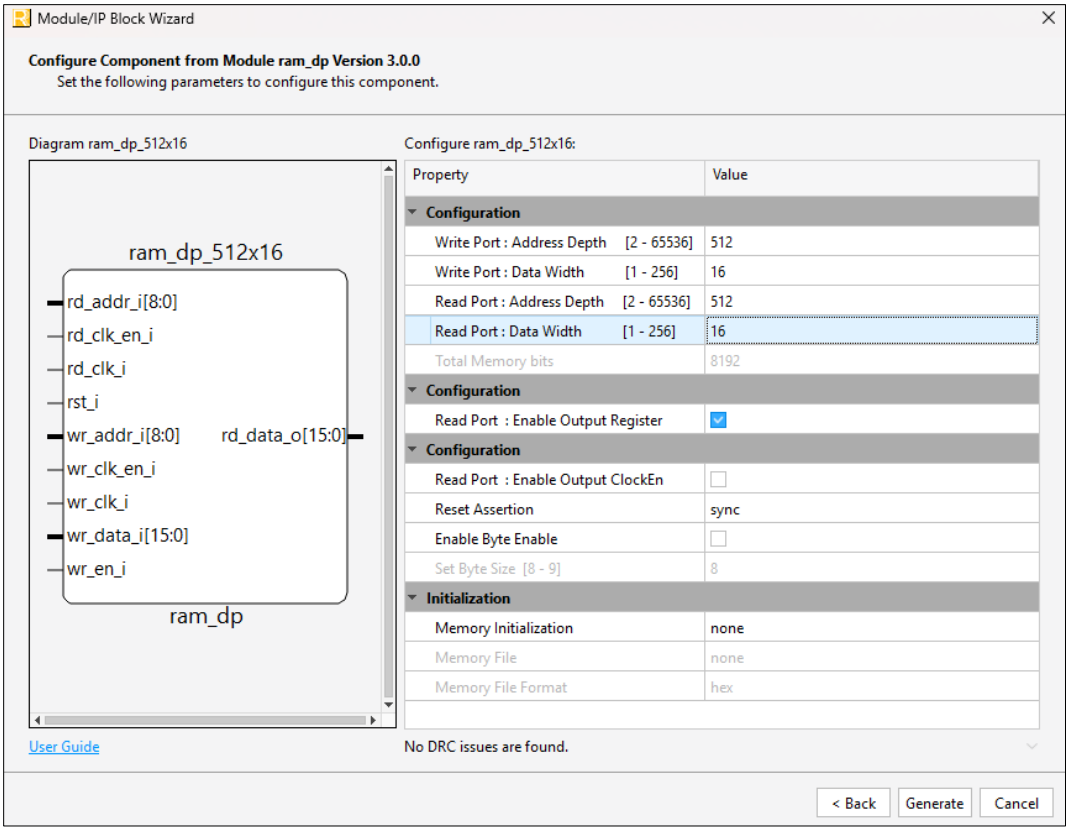

Figure 3.3. Pseudo Dual-Port RAM (RAM_DP) Module Customization

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, such as 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 16 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 check box **Enable Output Register** inserts the output registers in the Read Data Port, as the output registers are optional for EBR-based RAMs.

The Clock Enable control is always provided for Input Data and Address signals. When Output Registers are enabled, separate Output Clock Enables can be selected.

You can specify the use of Byte Enables. Byte Enables can be used to mask the input data so that only specific bytes of memory are overwritten. The unwritten bytes retain the previously written data.

You can specify the Reset Mode of the memory for assertion and release. For the synchronous reset, the clock should be there and the reset signal should satisfy setup or hold time requirements for both asserting and deasserting edges. The write function is automatically disabled during a synchronous reset because the CS registers are reset, but the write is not disabled during an asynchronous reset operation.

The asynchronous reset can be programmed to be released or deasserted synchronously. As shown in Figure 3.4, when deasserted synchronously, the first clock edge after reset releases the internal reset to all registers that have asynchronous reset, such as the data output registers, the FIFO counters, and the FIFO flag registers.

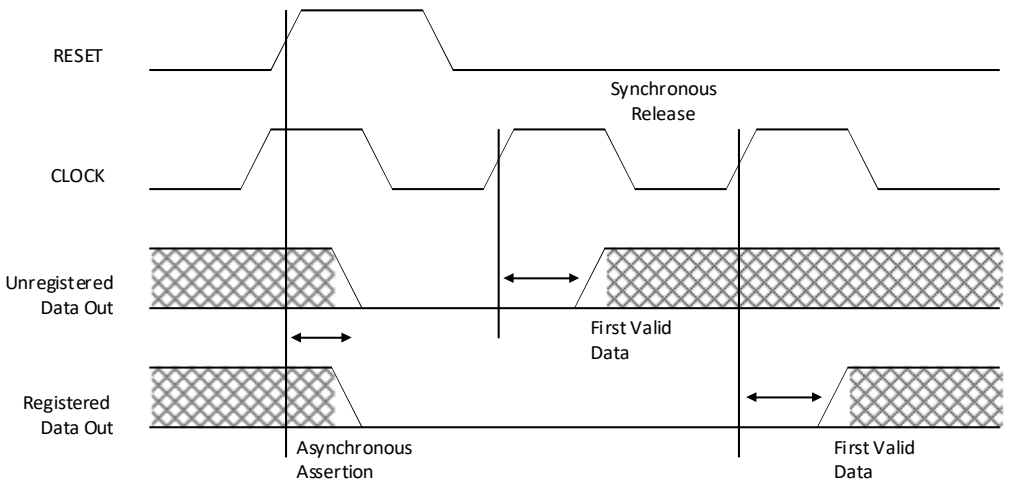

Figure 3.4. Asynchronous Reset with Synchronous Release

A memory may be initialized at configuration to all 1s or all 0s. To maximize the number of UFM/Flash bits, initialize the EBRs to an all 0's pattern. Initializing to an all 0's pattern does not use up UFM/Flash bits. You can also initialize your memory with the contents specified in the Memory File. It is optional to provide this file for RAM. However, for ROM, the Memory File is required. These files can be of the Binary or Hex format. The details of these formats are discussed in the Initializing Memory section of this document.

At this point, you can click the **Generate** button to generate the module you have customized. A module (*.ipx) is then generated and placed in the specified location. You can insert this module in your design. The module contains full Verilog and black box Verilog file (*.v and *_bb.v), a constraints file (*.ldc), supporting parameter and instantiation files (dut_inst.v, constraint.pdc, dut_params.v), Verilog and VHDL templates (*_tmpl.vhd and *_tmpl.v), and various testbench support files. Once the module is generated, you can either instantiate the *.ipx or the Verilog file in top-level module of the design.

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 the 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

IP Catalog allows you to implement Error Correction 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[1:0] = 01 – Indicates there is a 1-bit error which is fixed.
- Error[1:0] = 10 – Indicates there is a 2-bit error which cannot be corrected.
- Error[1:0] = 11 – Not used.

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 Lattice Radiant 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 the Lattice Radiant 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.

PMI modules are instantiated the same way other modules are in your HDL. The process is similar to the process for IP Catalog with the addition of setting parameters to customize the module. The Lattice Radiant software provides a template for the Verilog or VHDL instantiation command that specifies the customized module's ports and parameters.

Refer to the Lattice Radiant online help section, Instantiating a PMI Component 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. IP Catalog Memory Modules

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

The EBR blocks in the MachXO4 devices can be configured as Single-Port RAM, RAM_DQ. IP Catalog allows you to generate the Verilog netlist for the memory size, as per design requirements.

IP Catalog generates the RAM_DQ memory module, as shown in Figure 4.1.

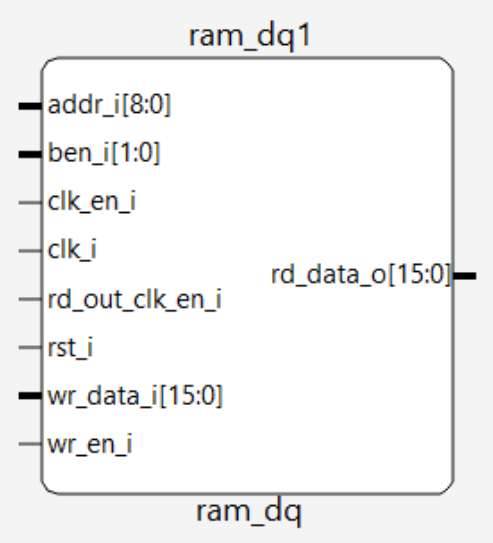

Figure 4.1. Single-Port Memory Module Generated by IP Catalog

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 you specify in the IP Catalog interface. 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.

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

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
clk_i	Clock	Clock	Rising Clock Edge
clk_en_i[2]	Clock Enable	ClockEn	Active High
rd_out_clk_en_i[1, 3]	Output Clock Enable	—	Active High
rst_i[4]	Reset	—	Active High
ben_i[1, 5]	Byte Enable	—	Active High
wr_en_i	Write Enable	WE	Active High
addr_i	Address Bus	Address	—
wr_data_i	Data In	Data	—
rd_data_o	Data Out	Q	—
ERROR[1]	Error Check Code	—	Active High

Notes:

1. Denotes optional ports.
2. clk_en_i is used as clock enable for all the input registers.

3. rd_out_clk_en_i can be used as clock enable for the optional output registers. This allows the full pipeline of data to be output, including the last word.
4. rst_i resets only the optional output registers of the RAM. It does not reset the input registers or the contents of memory.
5. ben_i can be used to mask the input data so that only specific bytes of memory are overwritten.

The Single-Port RAM, RAM_DQ, can be configured in NORMAL, READ-BEFORE-WRITE, or WRITE-THROUGH modes. Each of these modes affects what data comes out of the port rd_data_o of the memory during the write operation followed by the read operation at the same memory location.

IP Catalog implements the MachXO4 Single-Port RAM RAM_DQ using an appropriately configured DP8KC primitive.

Figure 4.2 to Figure 4.7 show the internal timing waveforms for the Single-Port RAM, RAM_DQ.

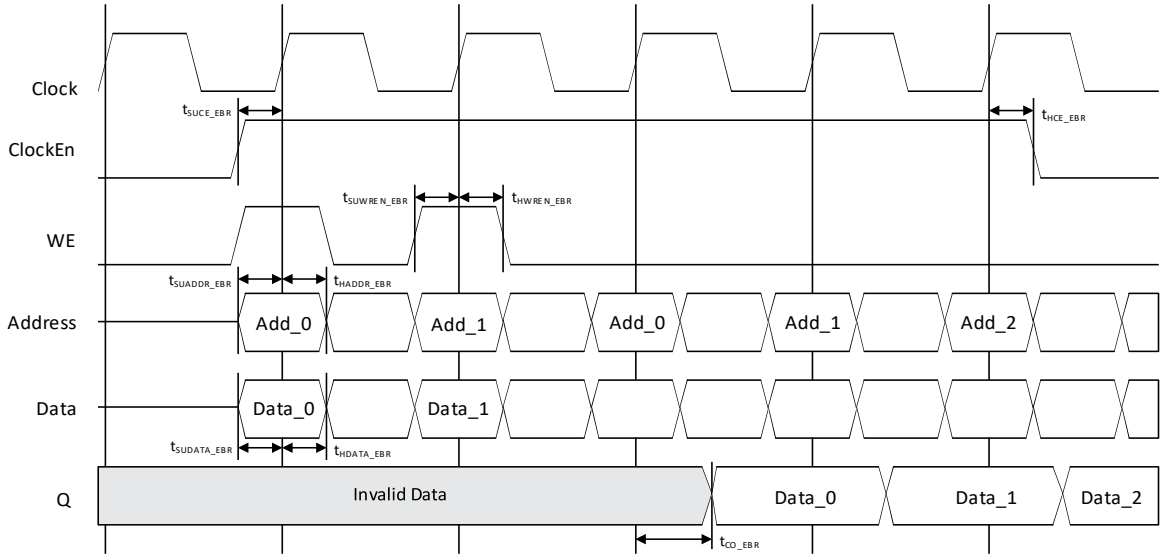

Figure 4.2. Single-Port RAM Timing Waveform – NORMAL Mode, without Output Registers

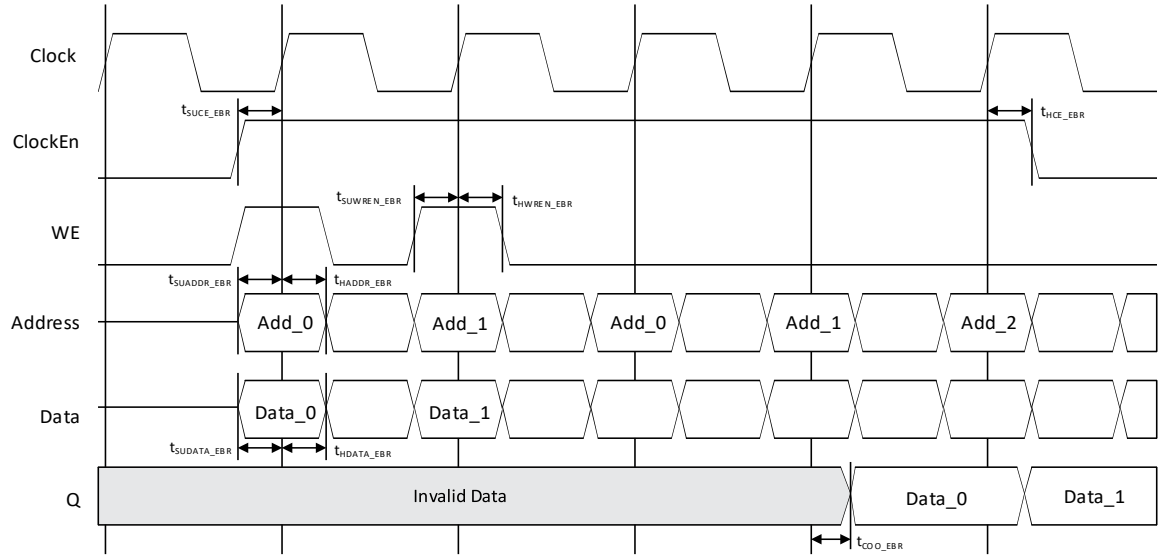

Figure 4.3. Single-Port RAM Timing Waveform – NORMAL Mode, with Output Registers

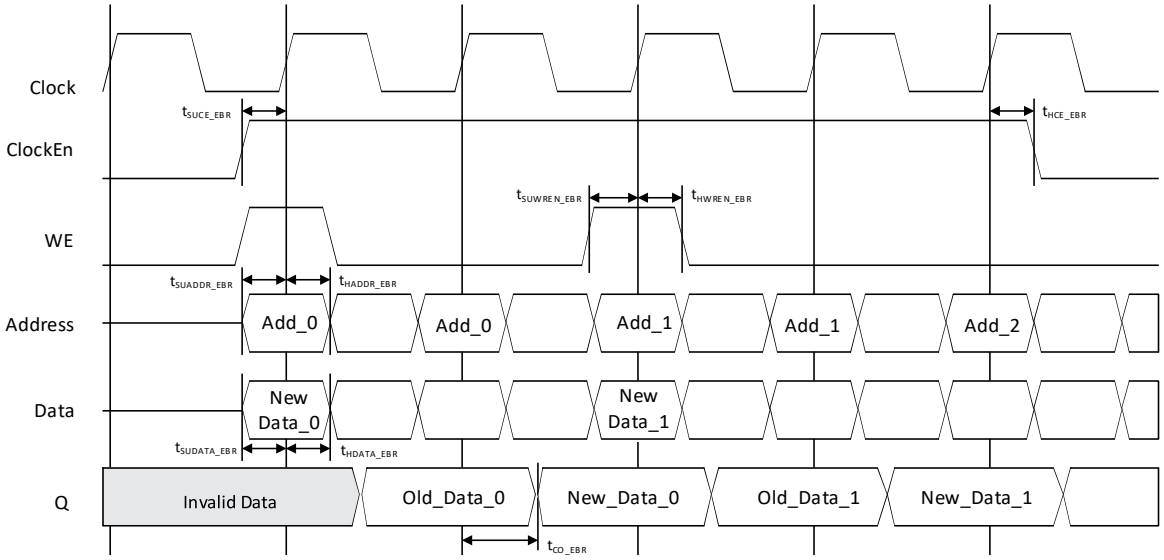

Figure 4.4. Single-Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers

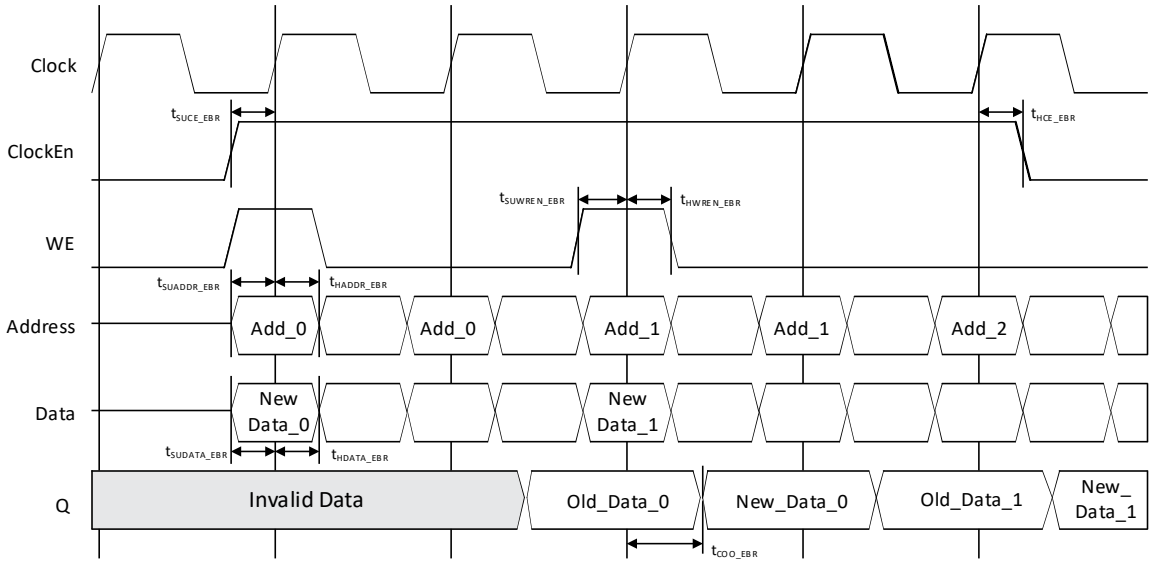

Figure 4.5. Single-Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers

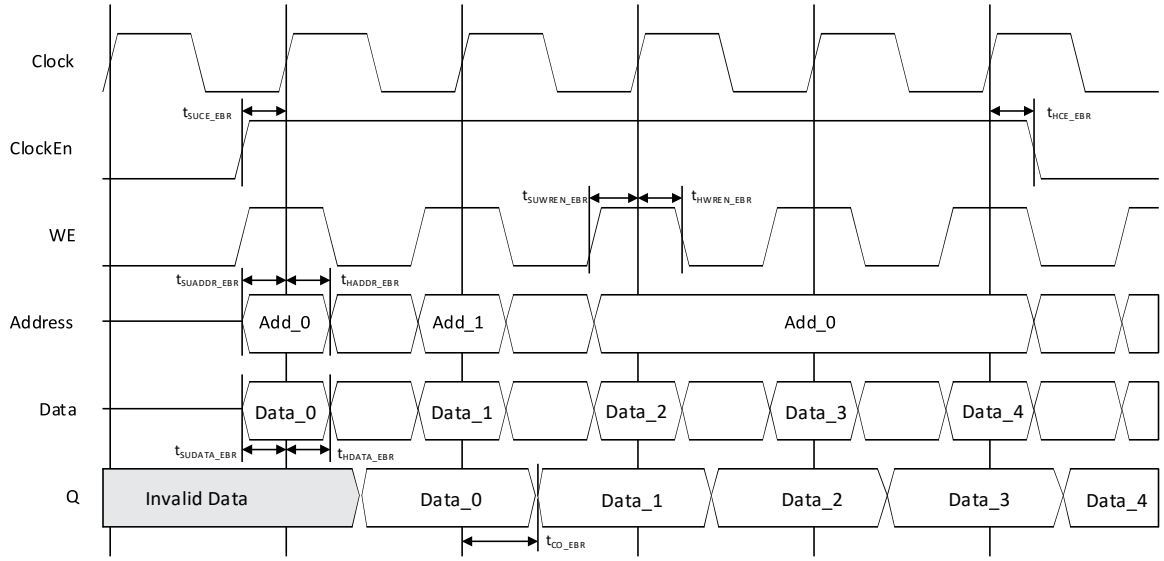

Figure 4.6. Single-Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

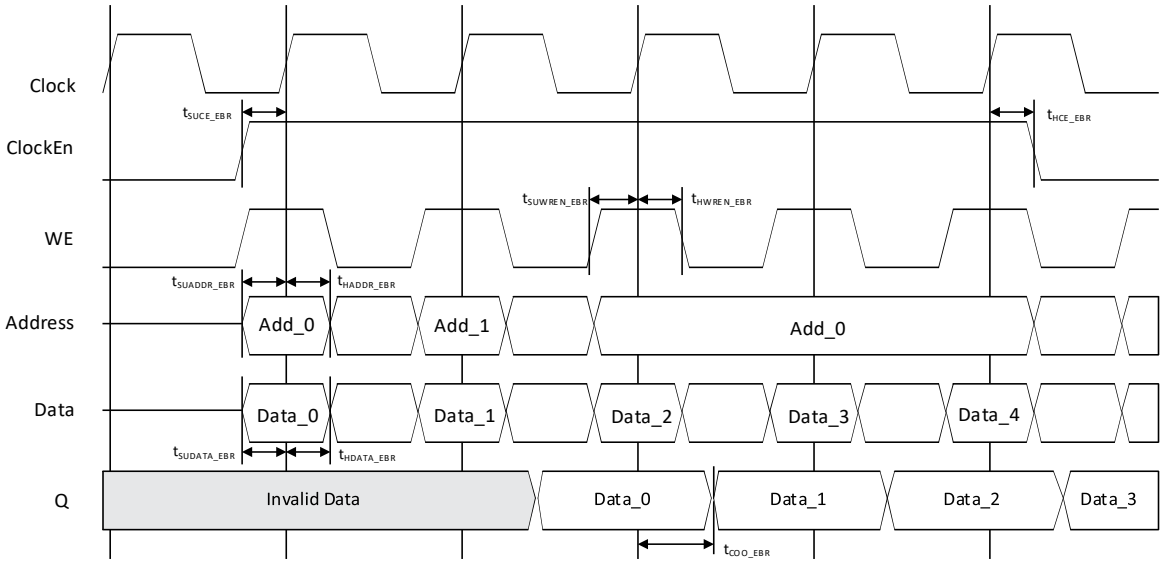

Figure 4.7. Single-Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

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

The EBR blocks in MachXO4 devices can be configured as True Dual-Port RAM, RAM_DP_TRUE. IP Catalog allows you to generate the Verilog netlists for various memory sizes depending on design requirements.

IP Catalog generates the RAM_DP_TRUE memory module, as shown in Figure 4.8.

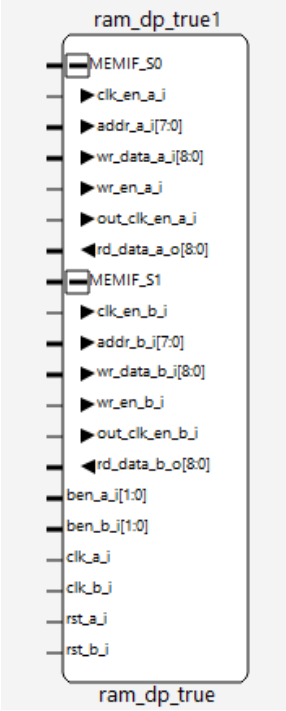

Figure 4.8.True Dual-Port Memory Module Generated by IP Catalog

Since the device has a number of EBR blocks, the generated module makes use of these EBR blocks, or primitives. The module cascades them to create the memory sizes you specify in the IP Catalog interface. 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 the 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 True Dual-Port Memory are in Table 4.2.

Table 4.2. EBR-Based True Dual-Port Memory Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
wr_data_a_i, wr_data_b_i	Input Data port A and port B	DataA, DataB	—
addr_a_i, addr_b_i	Address Bus port A and port B	AddressA, AddressB	—
clk_a_i, clk_b_i	Clock for Port A and port B	ClockA, ClockB	Rising Clock Edge
clk_en_a_i, clk_en_b_i[1, 2]	Clock Enables for Port CLKA and CLKB	ClockEnA, ClockEnB	Active High
out_clk_en_a_i[1], out_clk_en_b_i [1, 3]	Output Clock Enables for PortA and PortB	—	Active High
wr_en_a_i, wr_en_b_i	Write Enable Port A and Port B	WrA, WrB	Active High
rst_a_i, rst_b_i[4]	Reset for Port A and Port B	—	Active High
rd_data_a_o, rd_data_b_o	Output Data Port A and Port B	QA, QB	—
ben_a_i[1], ben_b_i[1, 5]	Byte Enable Port A and Port B	—	Active High
ERROR[1]	Error Correction Code	—	Active High

Notes:

1. Denotes optional ports.
2. clk_en_a_i and clk_en_b_i are used as clock enable for all the input registers.
3. out_clk_en_a_i and out_clk_en_b_i can be used as clock enable for the optional output registers. This allows the full pipeline of data to be output, including the last word.
4. rst_a_i and rst_b_i reset only the optional output registers of the RAM. They do not reset the input registers or the contents of memory.
5. ben_a_i and ben_b_i can be used to mask the input data so that only specific bytes of memory are overwritten.

The True Dual-Port RAM, RAM_DP_TRUE, can be configured as NORMAL, READ BEFORE WRITE, or WRITE THROUGH modes. Each of these modes affects what data comes out of the port Q of the memory during the write operation followed by the read operation at the same memory location.

IP Catalog implements the MachXO4 True Dual-Port RAM, RAM_DP_TRUE, using the DP8KC primitive.

Figure 4.9 through Figure 4.14 show the internal timing waveforms for the True Dual-Port RAM, RAM_DP_TRUE.

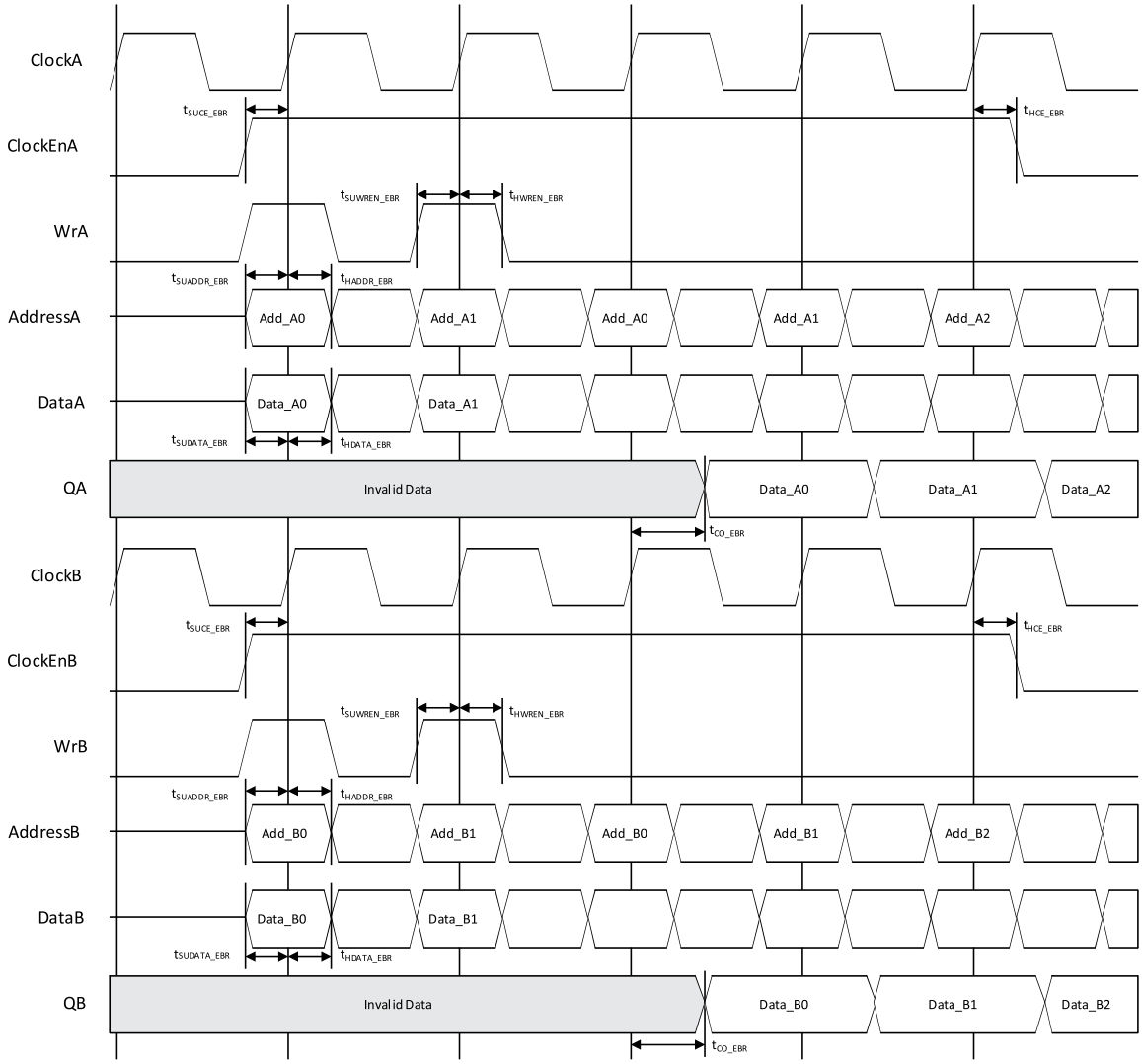

Figure 4.9. True Dual-Port RAM Timing Waveform – NORMAL Mode, without Output Registers

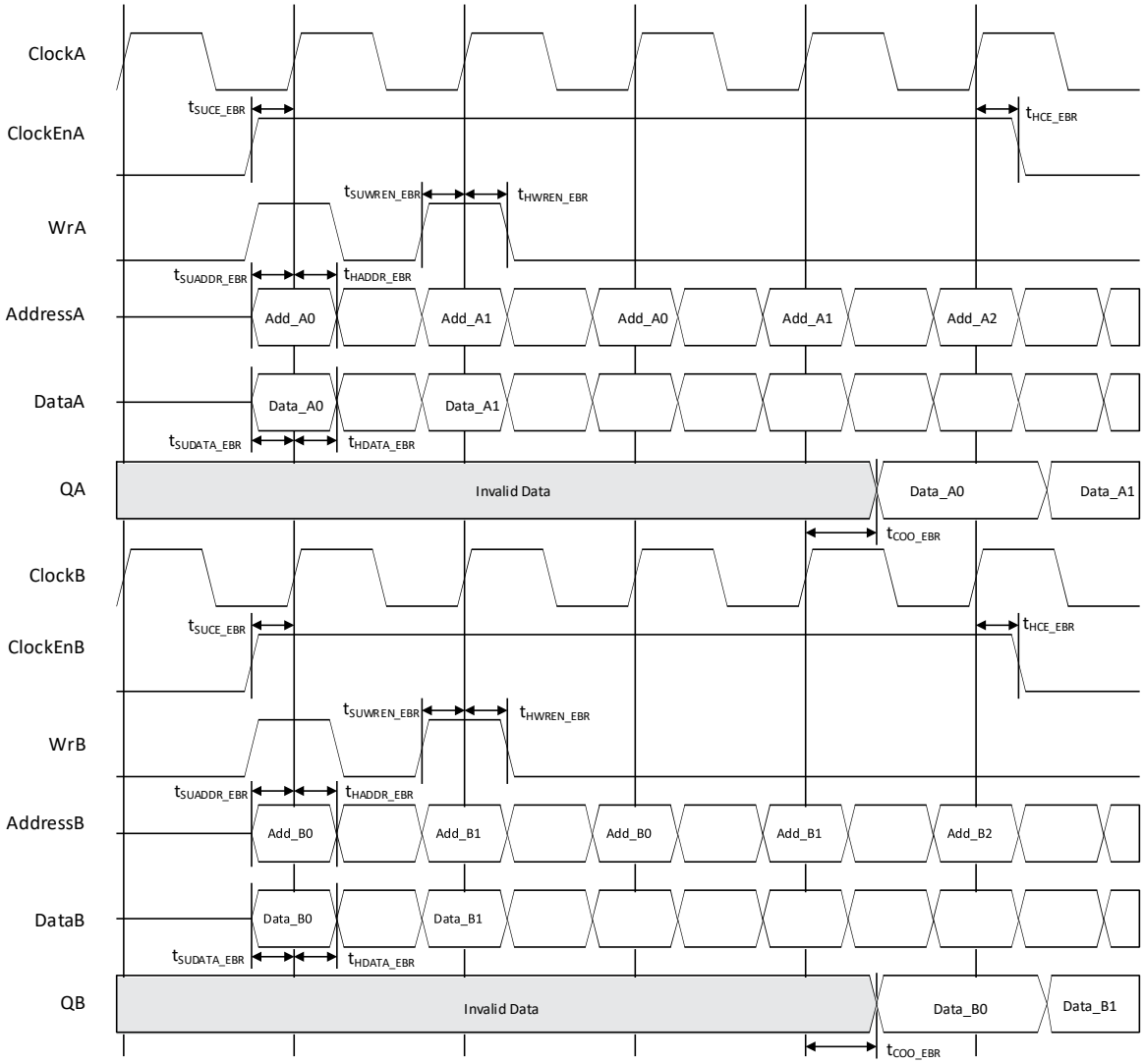

Figure 4.10. True Dual-Port RAM Timing Waveform – NORMAL Mode with Output Registers

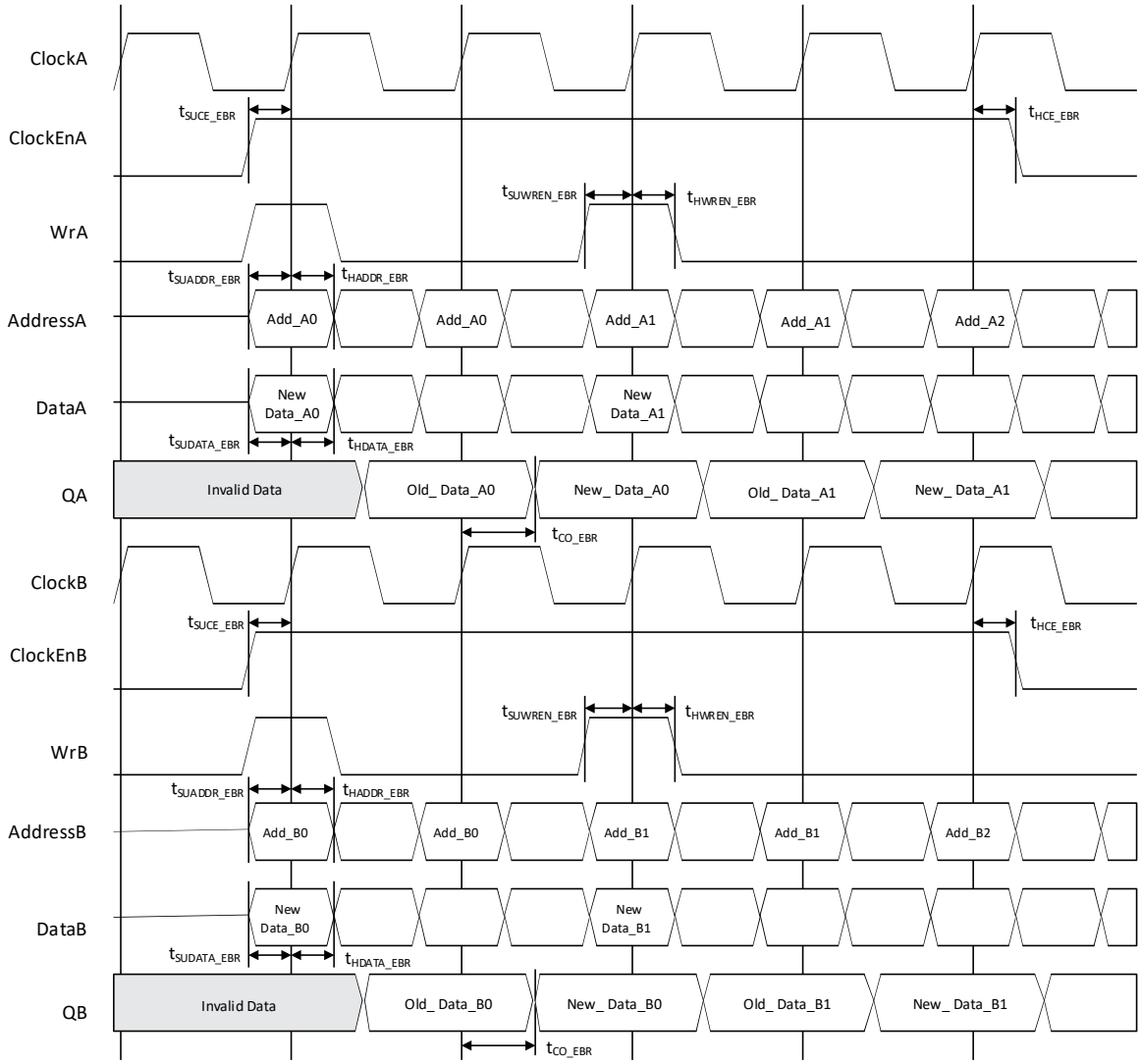

Figure 4.11. True Dual-Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers

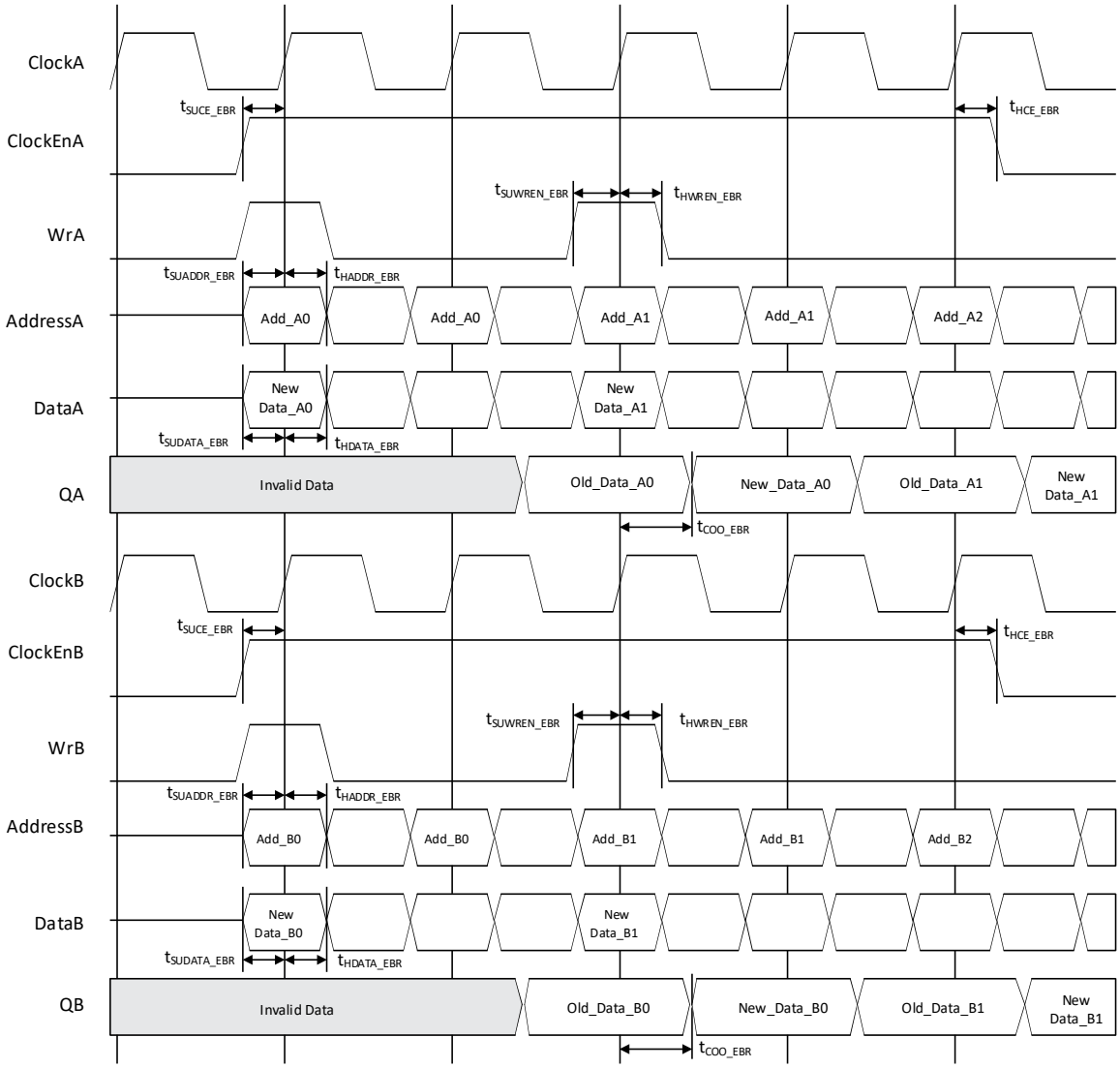

Figure 4.12. True Dual-Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers

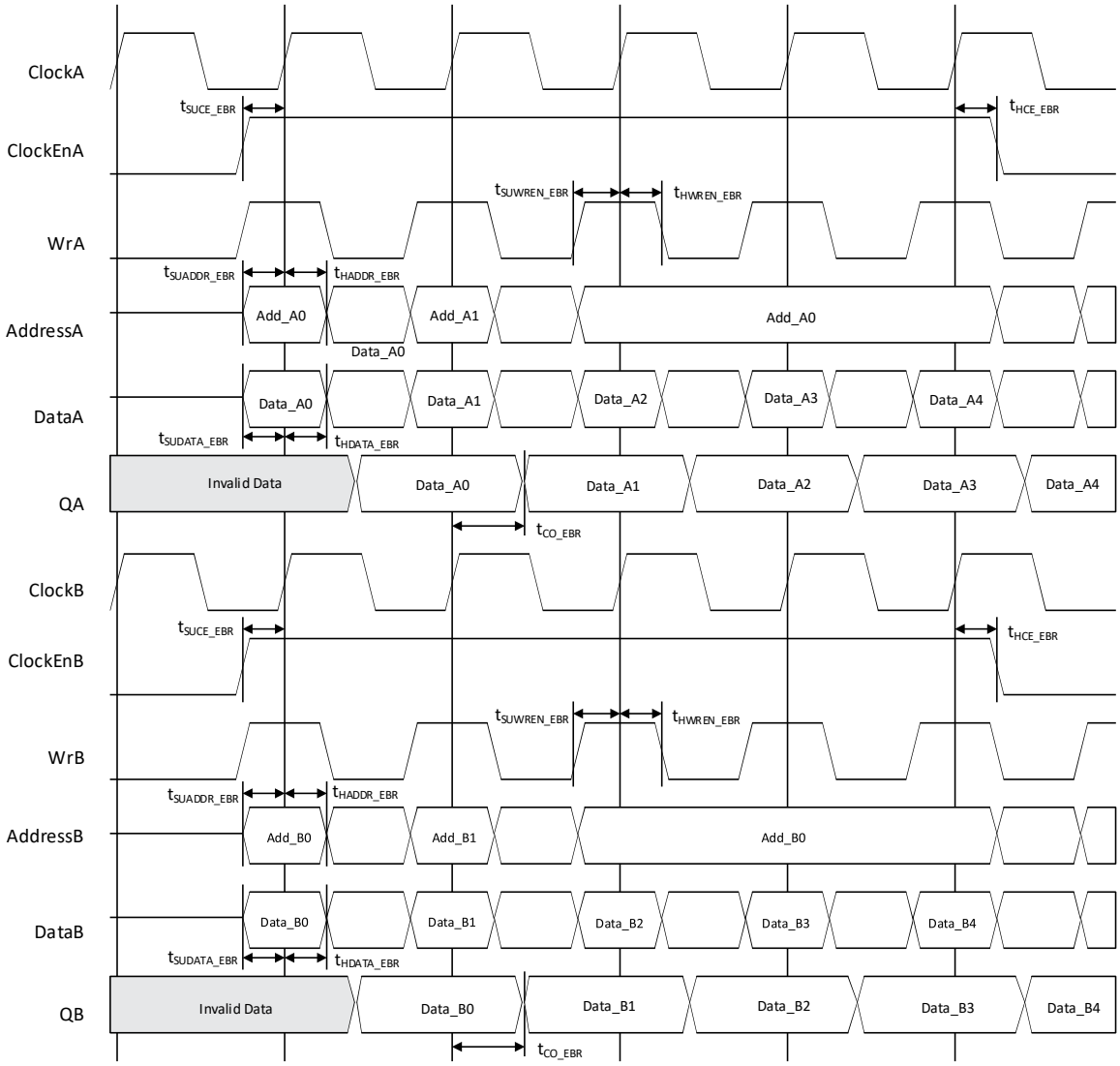

Figure 4.13. True Dual-Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

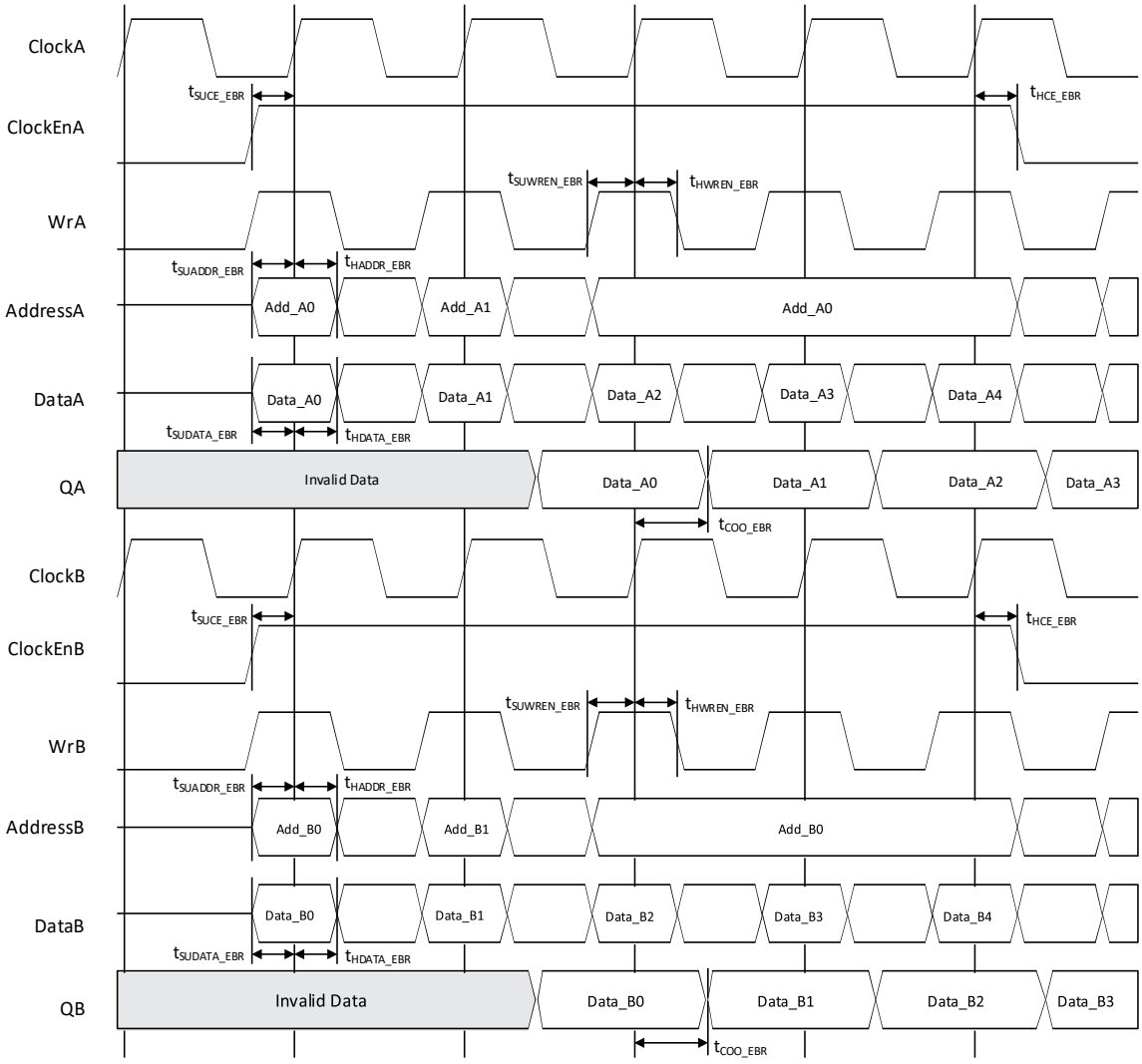

Figure 4.14. True Dual-Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

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

The EBR blocks in the MachXO4 devices can be configured as Pseudo Dual-Port RAM, RAM_DP. IP Catalog allows you to generate the Verilog-HDL or VHDL netlists for various memory sizes depending on design requirements.

IP Catalog generates the RAM_DP memory module, as shown in Figure 4.15.

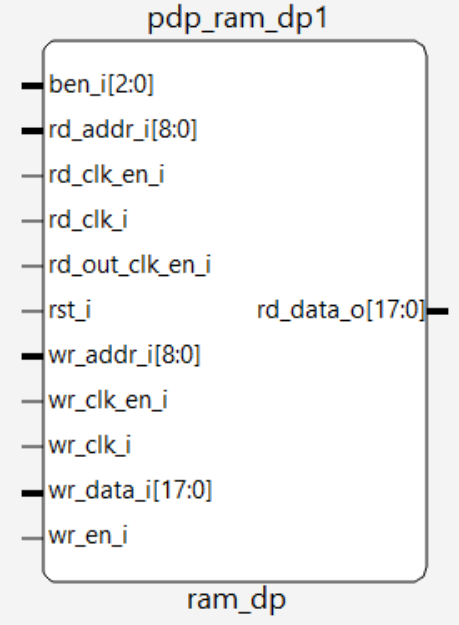

Figure 4.15. Pseudo Dual-Port Memory Module Generated by IP Catalog

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 you specify in the IP Catalog interface. 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 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 Pseudo Dual-Port Memory are listed in Table 4.3.

Table 4.3. EBR-Based Pseudo Dual-Port Memory Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
wr_addr_i	Write Address	WrAddress	—
rd_addr_i	Read Address	RdAddress	—
wr_data_i	Write Data	Data	—
ben_i[1, 2]	Byte Enable	—	Active High
wr_en_i	Write Enable	—	Active High
rd_clk_i	Read Clock	RdClock	Rising Edge
rd_clk_en_i[3]	Read Clock Enable	RdClockEn	Active High
rd_out_clk_en_i[1, 4]	Read Output Clock Enable	—	Active High
rst_i[5]	Reset	—	Active High
wr_clk_i	Write Clock	WrClock	Rising Edge
wr_clk_en_i[3]	Write Clock Enable	WrClockEn	Active High
rd_data_o	Read Data	Q	—
ERROR[1]	Error Check Code	—	Active High

Notes:

1. Denotes optional ports.
2. ben_i can be used to mask the input data so that only specific bytes of memory are overwritten.
3. rd_clk_en_i and wr_clk_en_i are used as clock enable for all the input registers.
4. rd_out_clk_en_i can be used as clock enable for the optional output registers. This allows the full pipeline of data to be output, including the last word.
5. rst_i resets only the optional output registers of the RAM. It does not reset the input registers or the contents of memory.

IP Catalog implements the MachXO4 Pseudo Dual-Port RAM RAM_DP using the PDPW8KC primitive, or the DP8KC primitive in configurations with narrow data port of 9 bits or less.

Figure 4.16 and Figure 4.17 show the internal timing waveforms for the Pseudo Dual-Port RAM, RAM_DP.

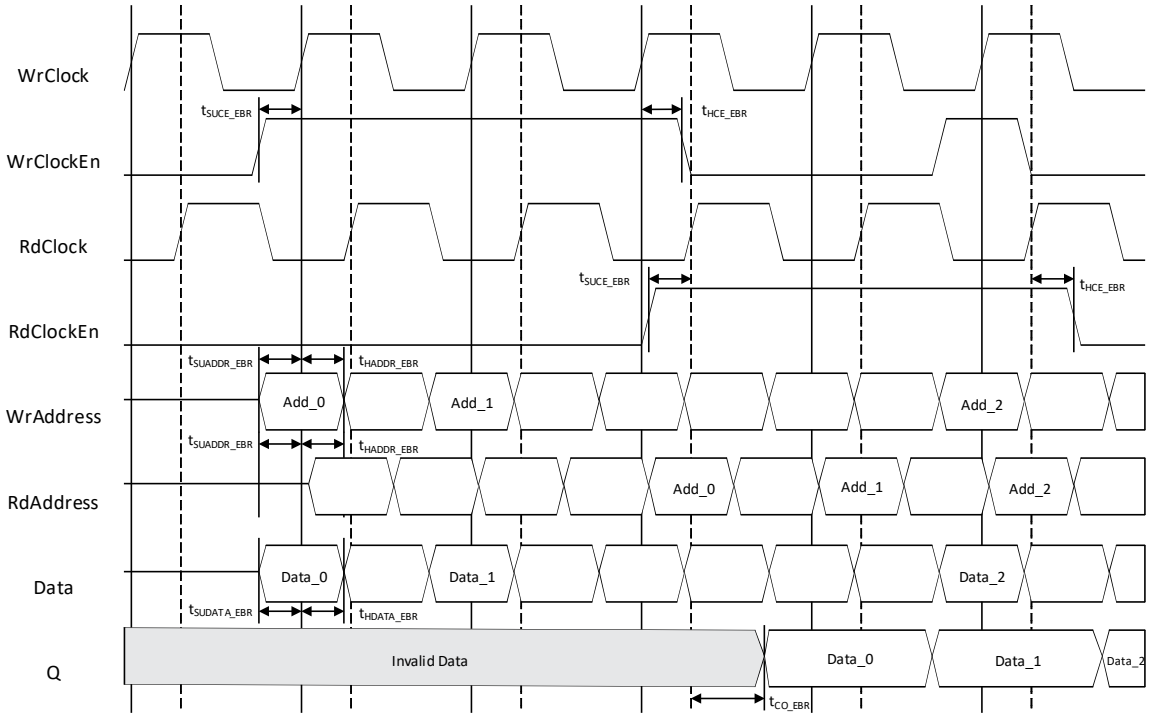

Figure 4.16. Pseudo Dual-Port RAM Timing Diagram – without Output Registers

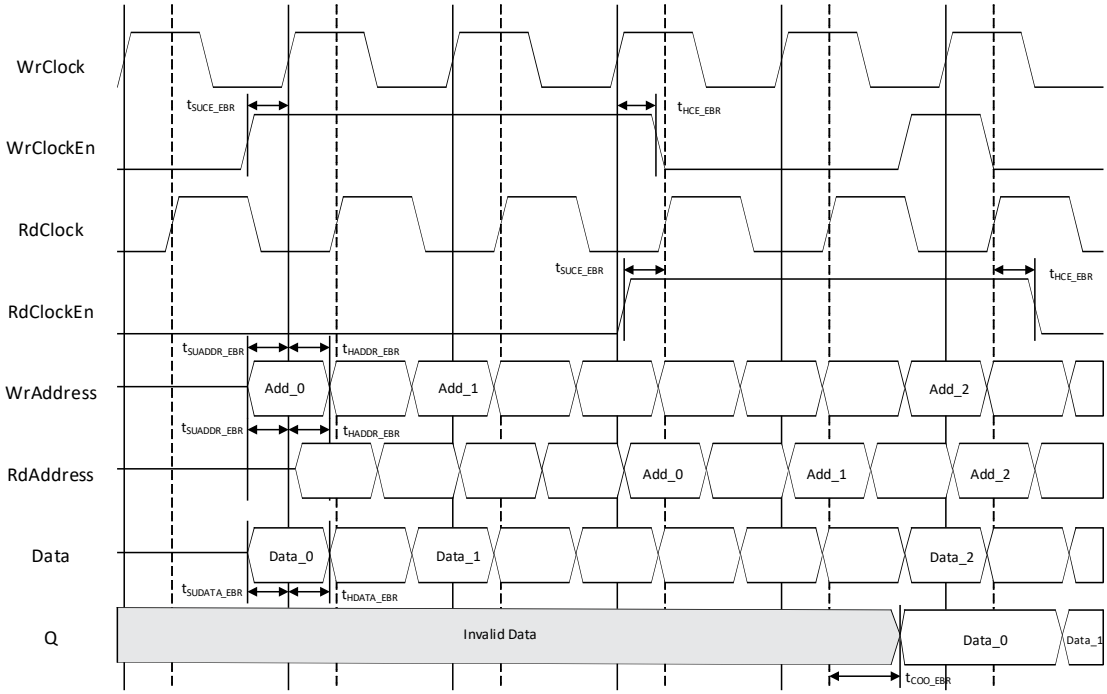

Figure 4.17. Pseudo Dual-Port RAM Timing Diagram – with Output Registers

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

The EBR blocks in the MachXO4 devices can be configured as Read-Only Memory (ROM). IP Catalog allows you to generate the Verilog-HDL or VHDL netlist for various memory sizes depending on design requirements. You are required to provide the ROM memory content in the form of an initialization file.

IP Catalog generates the ROM memory module, as shown in Figure 4.18.

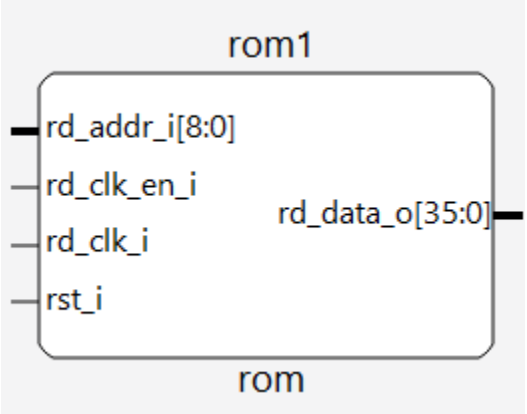

Figure 4.18. ROM – Read-Only Memory Module Generated by IP Catalog

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 you specify in the IP Catalog interface. 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.

The various ports and their definitions for the ROM module are listed in Table 4.4.

Table 4.4. EBR-Based ROM Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
rd_addr_i	Read Address	Address	—
rd_clk_i	Clock	OutClock	Rising Clock Edge
rd_clk_en_i[2]	Clock Enable	OutClockEn	Active High
rst_i[3]	Reset	—	Active High
rd_data_o	Read Data	Q	—
ERROR[1]	Error Check Code	—	Active High

Notes:

1. Denotes optional port.
2. rd_clk_en_i can be used as clock enable for the optional output registers.
3. rst_i resets only the optional output registers of the ROM. It does not reset the contents of the memory.

While generating the ROM using IP Catalog, 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 or Hex formats. The initialization files are discussed in detail in the Initializing Memory section of this document.

IP Catalog implements the MachXO4 Read-Only Memory, ROM, using an appropriately configured DP8KC primitive with write-enables tied low.

Figure 4.19 and Figure 4.20 show the internal timing waveforms for the Read-Only Memory, ROM.

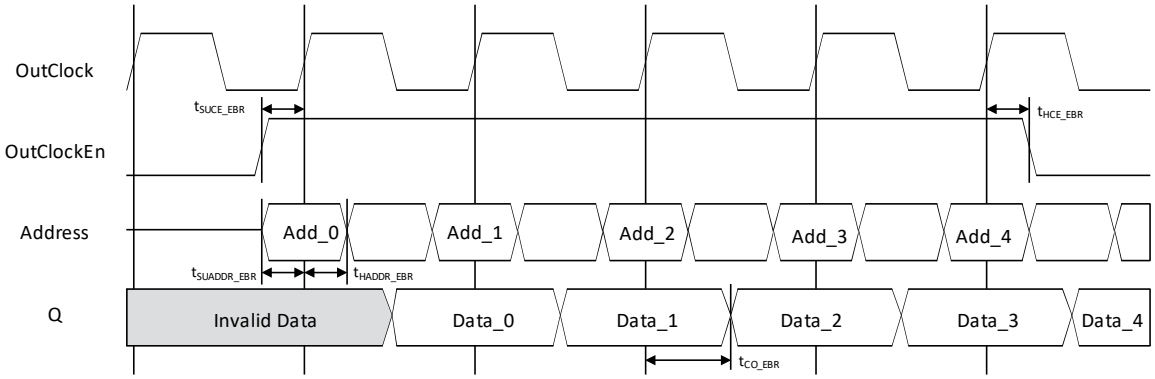

Figure 4.19. ROM Timing Waveform – without Output Registers

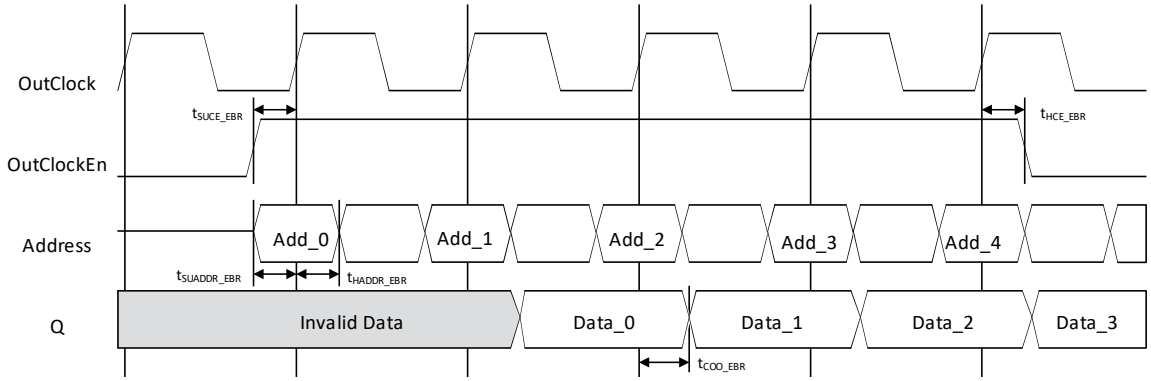

Figure 4.20. ROM Timing Waveform – with Output Registers

4.5. First In First Out (FIFO_DC) – EBR-Based

The EBR blocks in MachXO4 devices can be configured as Dual-Clock First-In First-Out Memory, FIFO_DC. IP Catalog allows you to generate the Verilog-HDL or VHDL netlist for various memory sizes depending on design requirements.

IP Catalog generates the FIFO_DC memory module, as shown in Figure 4.21.

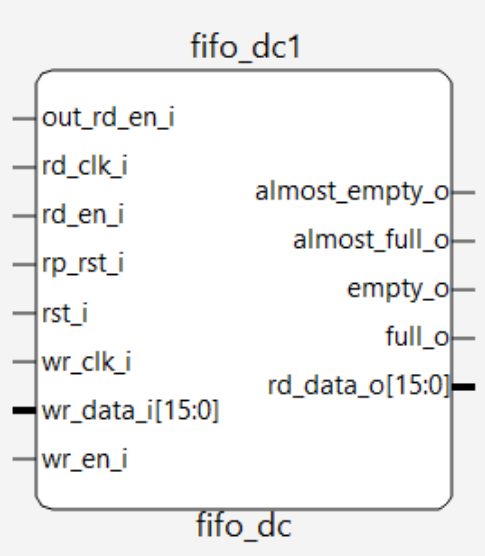

Figure 4.21. FIFO Module Generated by IP Catalog

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 you specify in the IP Catalog interface. 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 FIFO_DC mode, the input data 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 FIFO_DC are listed in Table 4.5.

Table 4.5. EBR-Based FIFO_DC Memory Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
wr_clk_i	Write Port Clock	WrClock	Rising Clock Edge
rd_clk_i	Read Port Clock	RdClock	Rising Clock Edge
wr_data_i	Data Input	Data	—
wr_en_i	Write Enable	WrEn	Active High
rd_en_i	Read Enable	RdEn	Active High
out_rd_en_i[1, 2]	Output Read Enable	—	Active High
rst_i[3]	Reset	Reset	Active High
rp_rst_i[4]	Read Pointer Reset	RPReset	Active High
rd_data_o	Data Output	Q	—
empty_o	Empty Flag	Empty	Active High
full_o	Full Flag	Full	Active High
almost_empty_o	Almost Empty Flag	Almost Empty	Active High
almost_full_o	Almost Full Flag	Almost Full	Active High
ERROR[1]	Error Check Code	—	Active High

Notes:

1. Denotes optional ports.
2. out_rd_en_i can be used as clock enable for the optional output registers. This allows the full pipeline of data to be output, including the last word.
3. rst_i resets only the optional output registers, pointer circuitry and flags of the FIFO. It does not reset the input registers or the contents of memory.
4. rp_rst_i resets only the read pointer. See additional discussion in the following sub-sections.

IP Catalog implements the MachXO4 Dual-Clock First-In First-Out Memory (FIFO_DC) using the FIFO8KB primitive.

4.5.1. FIFO_DC Flags

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

The program ranges for the four FIFO_DC flags are specified in Table 4.6.

Table 4.6. FIFO_DC Flag Settings

Port Name in the Generated Module	Description	Active State
full_o	Full flag setting	1 to (2^N -1)
almost_full_o	Almost full setting	1 to (FULL -1)
almost_empty_o	Almost empty setting	1 to (FULL -1)
empty_o	Empty setting	0

The value of Empty is fixed at 0. When coming out of reset, the active high flags Empty and Almost Empty are set to high, since they are true.

You should specify the absolute value of the address at which the Almost Empty and Almost Full flags go true. For example, if the Almost Full flag is required to go true at the address location 500 for a FIFO of depth 512, you should specify a value of 500 in IP Catalog.

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

At reset, both the write and read counters are pointing to address zero. After reset is deasserted, data can be written into the FIFO_DC to the address pointed to by the write counter at the positive edge of the write clock when the write enable is asserted. This write should occur at least two cycles after reset deassertion.

Similarly, data can be read from the FIFO_DC from the address pointed to by the read counter at the positive edge of the read clock when read enable is asserted.

Read Pointer Reset (rp_rst_i) is used to facilitate a retransmit operation and is more commonly used in packetize communications. Asserting rp_rst_i causes the internal read pointer to be reset to zero. It is typically used in conjunction with the assertion of Reset prior to each new packet, which resets both read and write pointers to zero. In this application, you must keep careful track of when a packet is written into or read from the FIFO_DC. To avoid the possible corruption of memory, rp_rst_i should not be asserted until the prior read cycle is complete, that is, rd_en_i is deasserted for one clock period. Upon the deassertion of rp_rst_i, the Empty and Almost Empty flags assume their correct state after one read clock cycle – this is a regular condition known as the boundary cycle latency.

In addition, in the case where the read pointer reset is asserted while write data is beyond the user specified address depth, data loss may occur.

The data output of the FIFO_DC can be registered or non-registered through a selection in IP Catalog. The output registers are enabled by read enable.

4.5.2. FIFO_DC Operation

If the output registers are not enabled, it takes one clock cycle to read the first word out.

If you enable the output registers, the output register causes an extra clock delay during the first data out as they are clocked by the read clock and enabled by the read enable.

1. First rd_en_i and Clock Cycle propagate the EF internally and generate internal Read Enable into the DPRAM.
2. Second rd_en_i and Clock Cycle get the data out of the output registers.

Figure 4.22 and Figure 4.23 show the internal timing waveforms for the Dual-Clock FIFO, FIFO_DC.

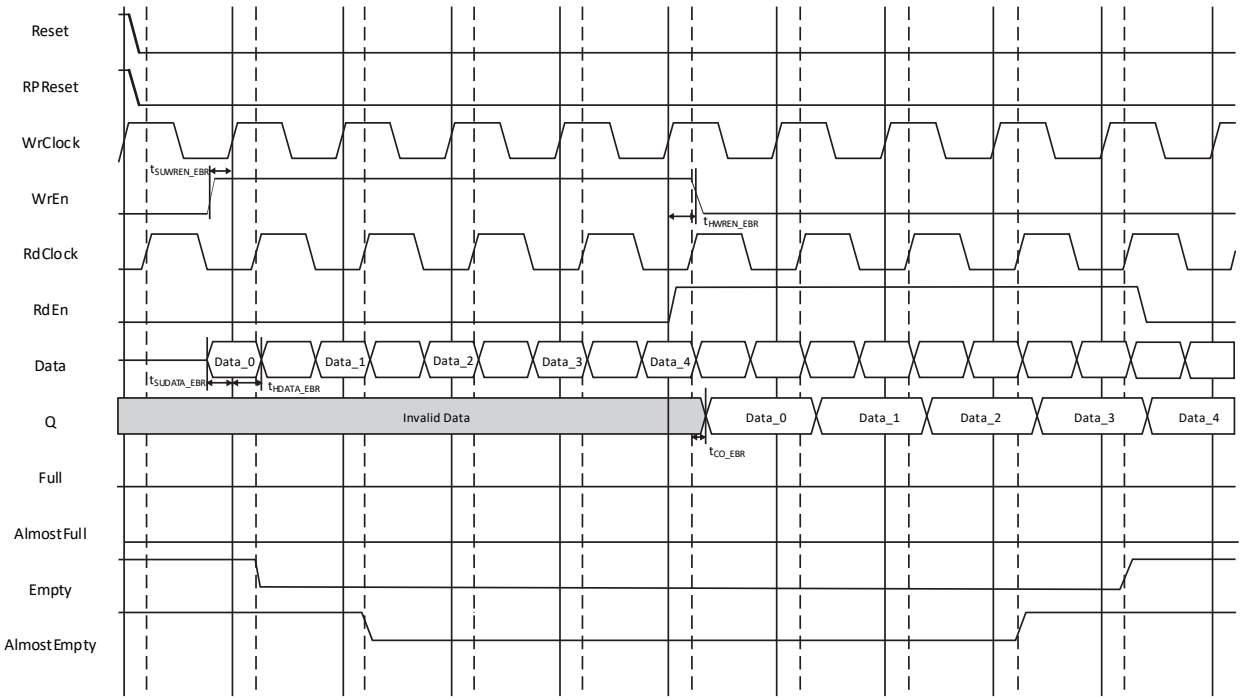

Figure 4.22. FIFO_DC without Output Registers – Non-Pipelined

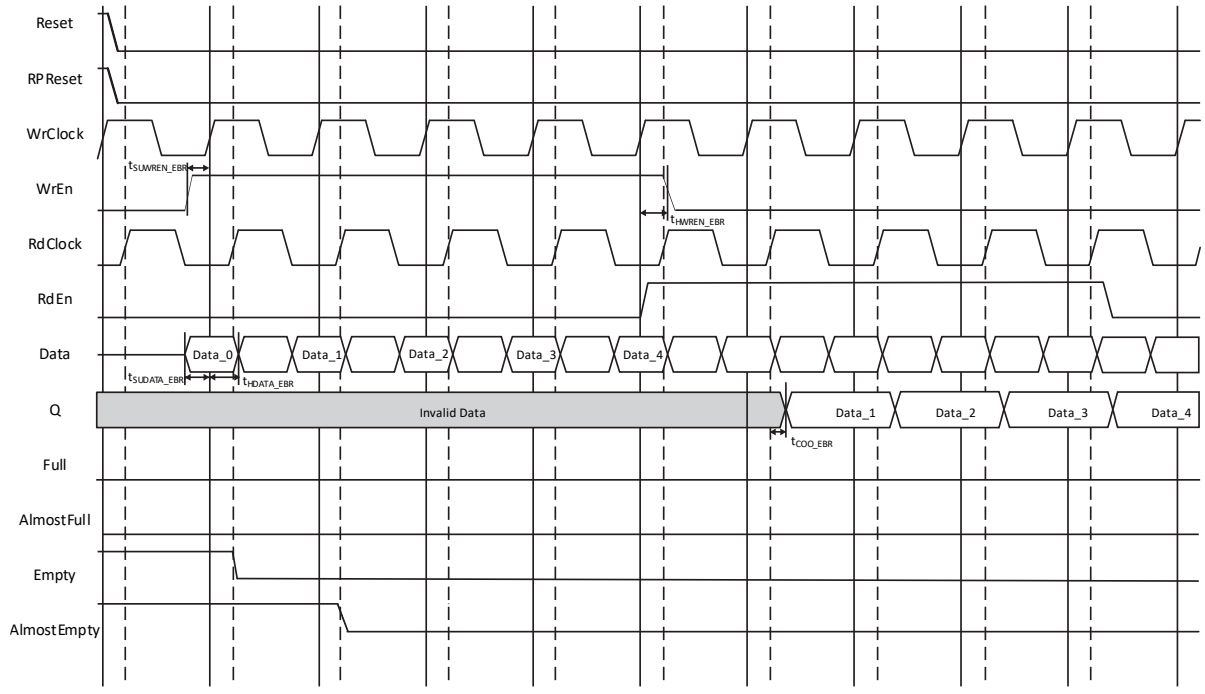

Figure 4.23. FIFO_DC with Output Registers – Pipelined

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

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

Figure 4.24 shows the Distributed Single-Port RAM module generated by IP Catalog.

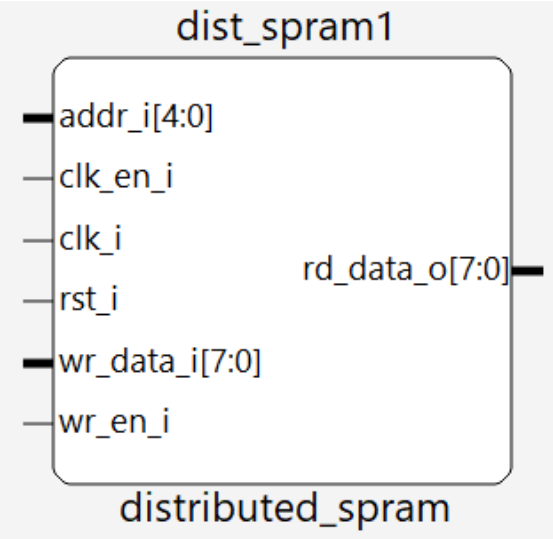

Figure 4.24.Distributed Single-Port RAM Module Generated by IP Catalog

The generated module makes use of the 4-input LUTs available in the PFU. Additional decode logic is generated by utilizing the resources available in the PFU.

The various ports and their definitions for the Memory are as listed in Table 4.7.

Table 4.7. PFU-Based Distributed Single-Port RAM Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
addr_i	Address	Address	—
wr_data_i	Data In	Data	—
clk_i	Clock	Clock	Rising Clock Edge
wr_en_i	Write Enable	WE	Active High
clk_en_i	Clock Enable	ClockEn	Active High
rst_i*	Reset	—	Active High
rd_data_o	Data Out	—Q	—

***Note**: rst_i is available only when Output Registers are enabled.

Figure 4.25 and Figure 4.26 show the internal timing waveforms for the Distributed Single-Port RAM, Distributed_SPRAM.

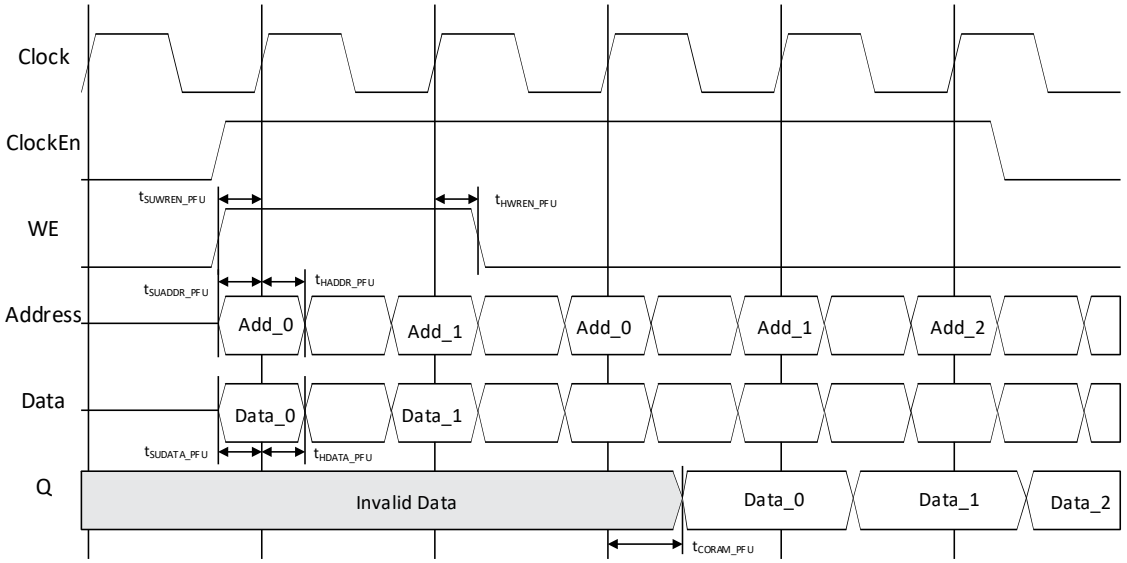

Figure 4.25. PFU-Based Distributed Single-Port RAM Timing Waveform – without Output Registers

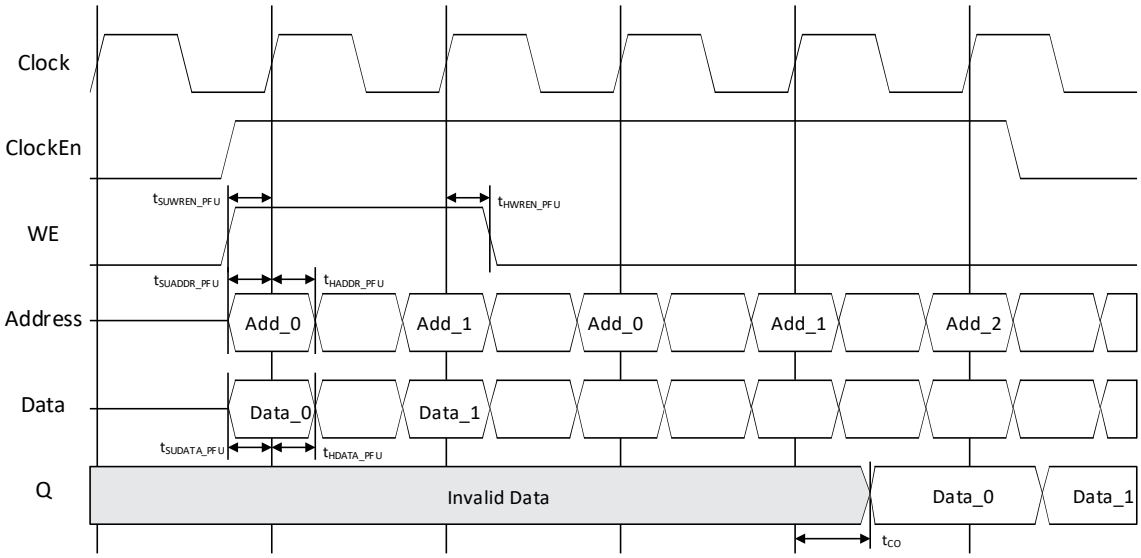

Figure 4.26. PFU-Based Distributed Single-Port RAM Timing Waveform – with Output Registers

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

PFU-based Distributed Dual-Port RAM is created using the 4-input LUTs available in the PFU.

These LUTs can be cascaded to create larger Distributed Memory sizes.

Figure 4.27 shows the Distributed Dual-Port RAM module generated by IP Catalog.

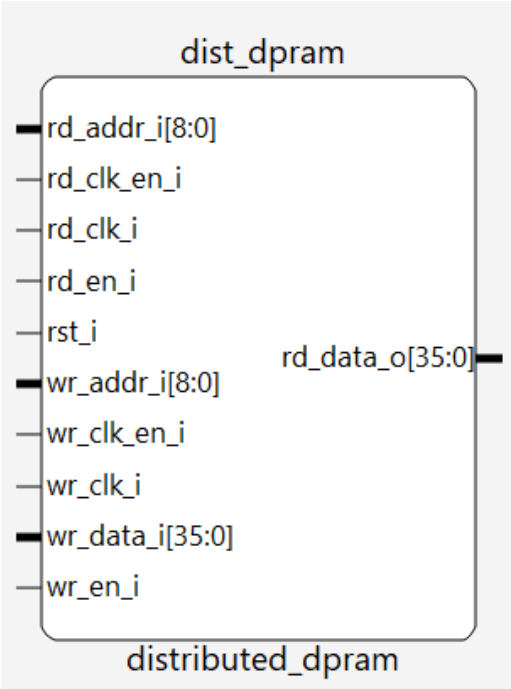

Figure 4.27. Distributed Dual-Port RAM Module Generated by IP Catalog

The generated module makes use of the 4-input LUTs available in the PFU. Additional decode logic is generated by utilizing the resources available in the PFU.

The various ports and their definitions are listed in Table 4.8.

Table 4.8. PFU-Based Distributed Dual-Port RAM Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
wr_addr_i	Write Address	WrAddress	—
wr_data_i	Data Input	Data	—
wr_clk_i	Write Clock	WrClock	Rising Clock Edge
wr_en_i	Write Enable	WE	Active High
wr_clk_en_i	Write Clock Enable	WrClockEn	Active High
rd_addr_i	Read Address	RdAddress	—
rd_clk_i*	Read Clock	RdClock	Rising Clock Edge
rd_clk_en_i*	Read Clock Enable	RdClockEn	Active High
rd_en_i	Read Enable	—	Active High
rst_i*	Reset	—	Active High
rd_data_o	Data Out	Q	—

***Note**: Denotes optional ports.

The optional ports Read Clock (rd_clk_i) and Read Clock Enable (rd_clk_en_i) are not available in the hardware primitive. These are generated by IP Catalog if you want to enable the output registers in the IP Catalog configuration.

Figure 4.28 and Figure 4.29 show the internal timing waveforms for the Distributed Dual-Port RAM, Distributed_DPRAM.

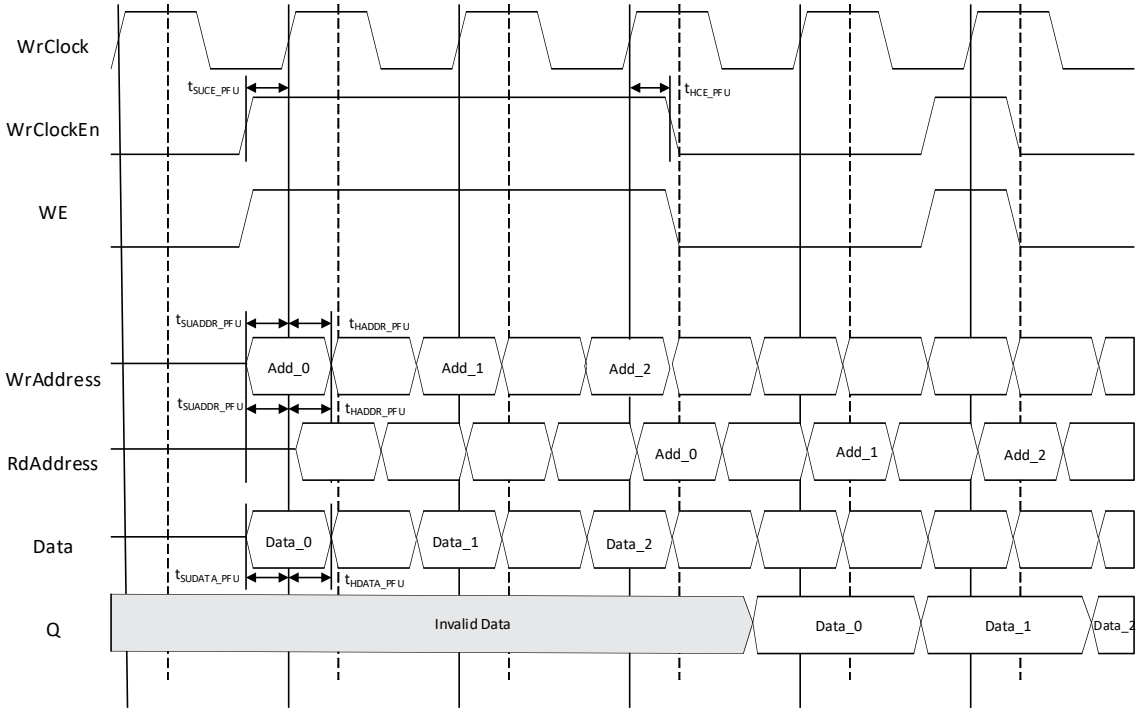

Figure 4.28. PFU-Based Distributed Dual-Port RAM Timing Waveform – without Output Registers

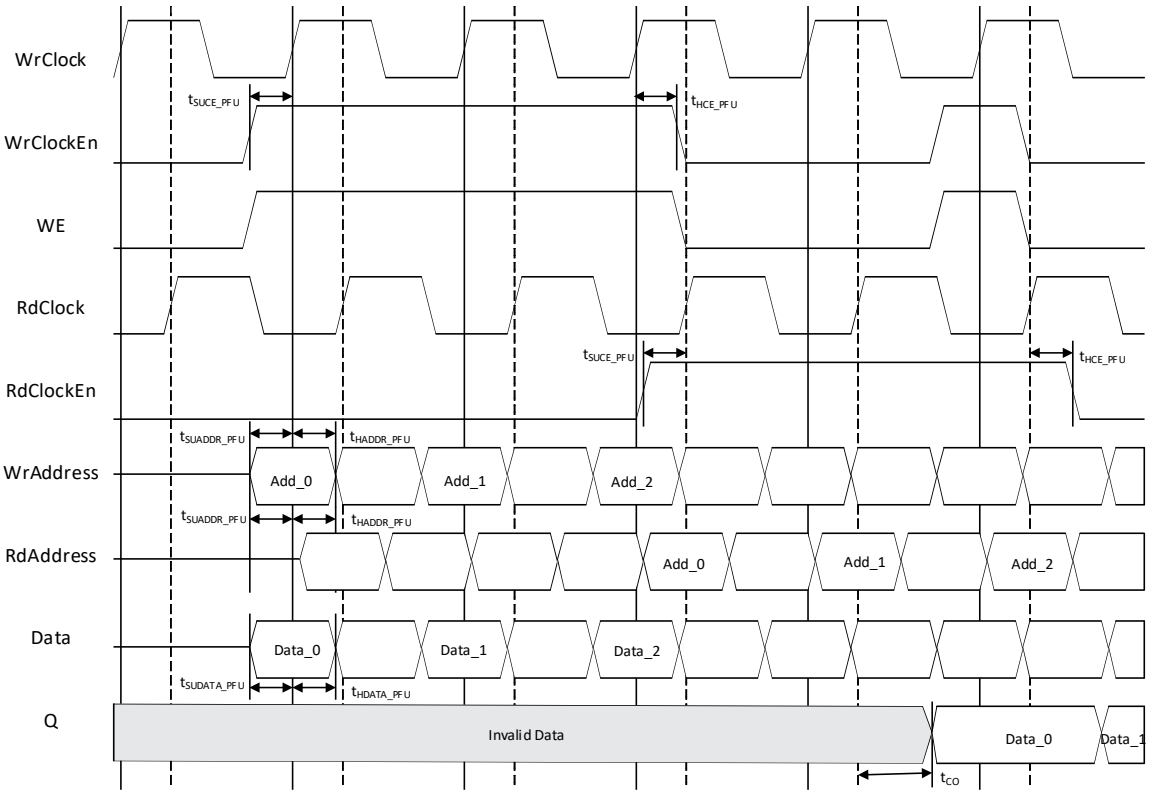

Figure 4.29. PFU-Based Distributed Dual-Port RAM Timing Waveform – with Output Registers

4.8. Distributed ROM (Distributed_ROM) – PFU-Based

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

Figure 4.30 shows the Distributed ROM module generated by IP Catalog.

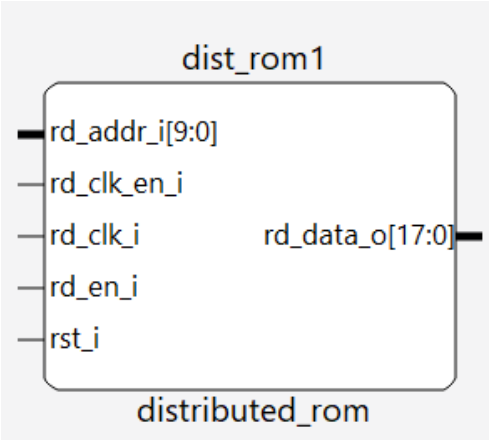

Figure 4.30.Distributed ROM Generated by IP Catalog

The generated module makes use of the 4-input LUTs available in the PFU. Additional decode logic is generated by utilizing the resources available in the PFU.

The various ports and their definitions are listed in Table 4.9.

Table 4.9. PFU-Based Distributed ROM Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
rd_addr_i	Address	Address	—
rd_clk_i*	Out Clock	OutClock	Rising Clock Edge
rd_clk_en_i*	Out Clock Enable	OutClockEn	Active High
rd_en_i	Out Enable	—	Active High
rst_i*	Reset	—	Active High
rd_data_o	Data Out	Q	—

***Note:** Denotes optional ports.

The optional ports Out Clock (rd_clk_i) and Out Clock Enable (rd_clk_en_i) are not available in the hardware primitive. These are generated by the IP Catalog when you want to enable the output registers in the IP Catalog configuration.

Figure 4.31 and Figure 4.32 show the internal timing waveforms for the Distributed ROM.

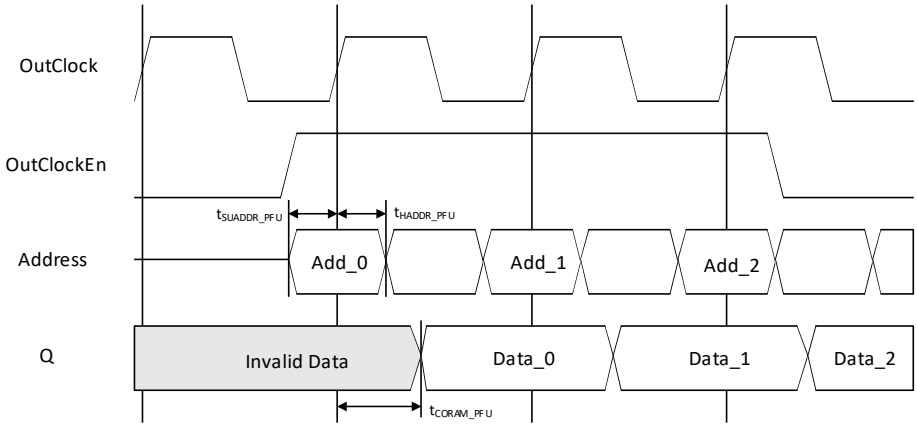

Figure 4.31. PFU-Based ROM Timing Waveform – Without Output Registers

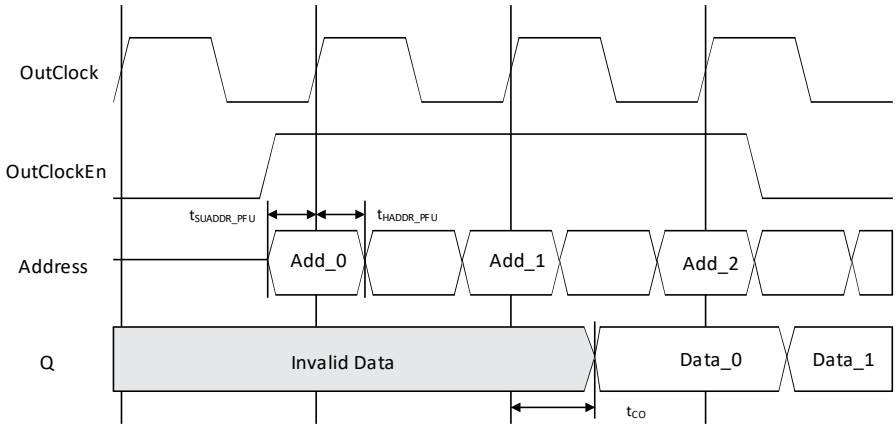

Figure 4.32. PFU-Based ROM Timing Waveform – With Output Registers

4.9. RAM-Based Shift Register

The Distributed SPRAM blocks in the MachXO4 devices, in combination with LUT-based logic, can be configured as a RAM-based Shift Register. IP Catalog allows you to generate the Verilog-HDL or VHDL netlist for the Shift Register length, upon design requirements.

IP Catalog generates the Shift Register module, as shown in Figure 4.33.

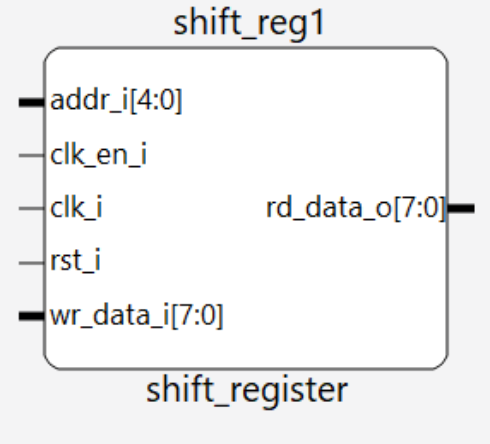

Figure 4.33. RAM-Based Shift Register Generated by IP Catalog

The generated module makes use of the 4-input LUTs available in the PFU. Additional logic is generated by utilizing the resources available in the PFU.

The various ports and their definitions are listed in Table 4.10.

Table 4.10. RAM-Based Shift Register Port Definitions

Port Name in the Generated Module	Description	Waveform Signal Name	Active State
wr_data_i	Data In	Din	—
addr_i*	Address	—	—
clk_i	Clock	Clock	Rising Clock Edge
clk_en_i	Clock Enable	ClockEn	Active High
rst_i	Reset	—	Active High
rd_data_o	Data Out	Q	—

***Note:** Denotes optional port.

The optional Address port is available only when the Variable Length type is selected. It is generated by IP Catalog when you want to enable the Variable Length operation in the IP Catalog configuration. Figure 4.34 and Figure 4.35 show the internal timing waveforms for the RAM-Based Shift Register.

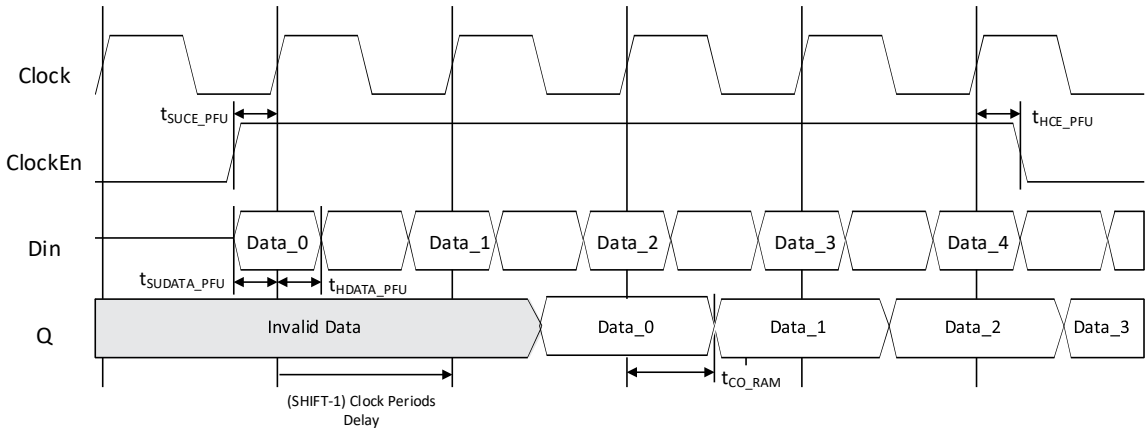

Figure 4.34. RAM-Based Shift Register Timing Waveform – without Output Registers, Shift = 2

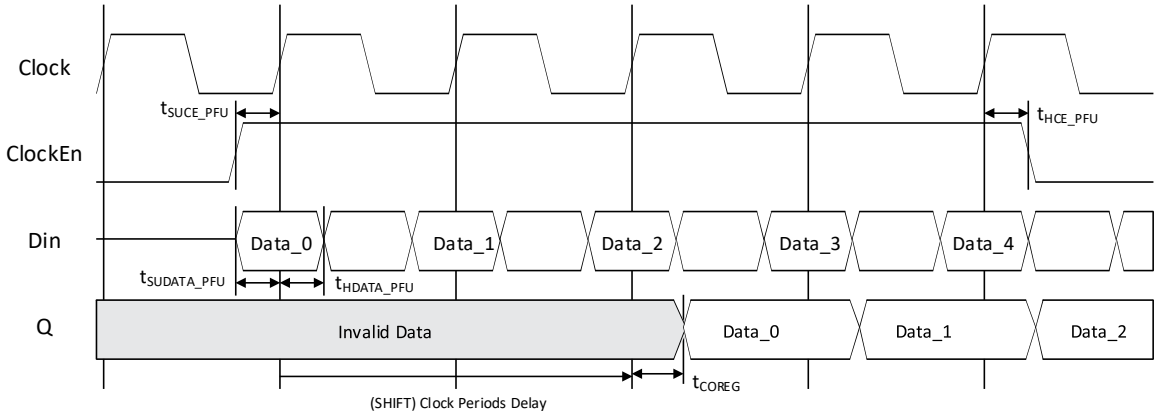

Figure 4.35. RAM-Based Shift Register Timing Waveform – with Output Registers, Shift = 2

5. MachXO4 Primitives

5.1. Single-Port RAM (SP8KC) – EBR-Based

The Single-Port RAM primitive is shown in Figure 5.1.

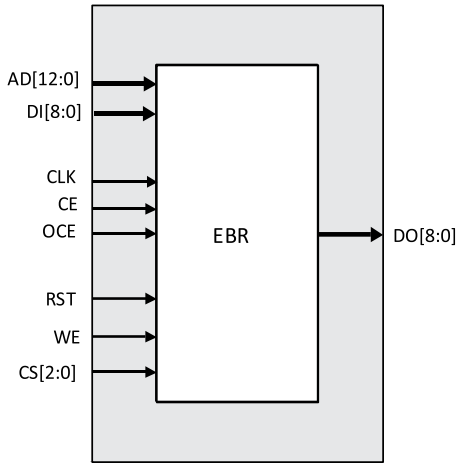

Figure 5.1. Single-Port RAM (SP8KC)

Table 5.1. EBR-Based Single-Port Memory Port Definitions

Port Name in the EBR Block Primitive (SP8KC)	Description	Active State
AD	Address Bus	—
DI	Data In	—
CLK	Clock	Rising Clock Edge
CE	Clock Enable	Active High
OCE	Output Clock Enable	Active High
RST	Reset	Active High
WE	Write Enable	Active High
CS[2:0]	Chip Select	—
DO	Data Out	—

Each SP8KC primitive consists of 9,216 bits of RAM. The possible values for address depth and data width for the SP8KC primitive are listed in Table 5.2.

Table 5.2. Single-Port Memory Sizes for 9K Memories in MachXO4

Single-Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
8K x 1	DI	DO	AD[12:0]
4K x 2	DI[1:0]	DO[1:0]	AD[12:1]
2K x 4	DI[3:0]	DO[3:0]	AD[12:2]
1K x 9	DI[8:0]	DO[8:0]	AD[12:3]

Table 5.3 shows the various attributes available for the SP8KC. Some of these attributes are user-selectable through the IP Catalog interface. For detailed attribute definitions, refer to Appendix A. Attribute Definitions.

Table 5.3. Single-Port RAM Attributes for MachXO4 (SP8KC)

Attribute	Description	Values	Default Value	User-Selectable through IP Catalog
DATA_WIDTH	Data Word Width	1, 2, 4, 9	9	Yes
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	Yes
RESETMODE	Selects Reset Type.	ASYNC, SYNC	SYNC	Yes
CSDECODE	Chip Select Decode	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111, 0b000	0b000	No
WRITEMODE	Read/Write Behavior	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	Yes
GSR	Enables Global Set Reset.	Reset ENABLED, DISABLED	DISABLED	No
INITVAL_00 .. INITVAL_1F	Initialization Value	0x0000000000000000000000 000000000000000000000000 000000000000000000000000 0000000000 .... 0xFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFF (80- character hex strings)	0x00000000000 0000000000000 0000000000000 0000000000000 0000000000000 0000000000000 0000	No
ASYNC_RESET_RELEASE	Reset Release	ASYNC, SYNC	SYNC	Yes
INIT_DATA	Init Values Status	STATIC, DYNAMIC	STATIC	Yes

5.2. True Dual-Port RAM (DP8KC) – EBR-Based

The True Dual-Port RAM primitive is shown in Figure 5.2.

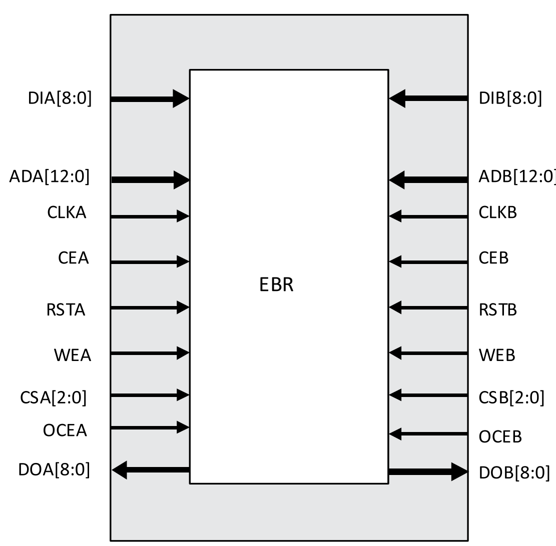

Figure 5.2. True Dual-Port RAM (DP8KC)

Table 5.4. EBR-Based True Dual-Port Memory Port Definitions

Port Name in the EBR Block Primitive (DP8KC)	Description	Active State
DIA, DIB	Input Data port A and port B	—
ADA, ADB	Address Bus port A and port B	—
CLKA, CLKB	Clock for Port A and Port B	Rising Clock Edge
CEA, CEB	Clock Enables for Port CLKA and CLKB	Active High
RSTA, RSTB	Reset for Port A and Port B	Active High
WEA, WEB	Write enable port A and port B	Active High
CSA[2:0], CSB[2:0]	Chip Selects for each port	—
OCEA, OCEB	Output Clock Enables for Port A and Port B	Active High
DOA, DOB	Output Data port A and port B	—

Each DP8KC primitive consists of 9,216 bits of RAM. The possible values for address depth and data width for the DP8KC primitive are listed in Table 5.5.

Table 5.5. True Dual-Port Memory Sizes for 9K Memory in MachXO4 Devices

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

Table 5.6 shows the various attributes available for the True Dual-Port Memory, RAM_DP_TRUE. Some of these attributes are user-selectable through the IP Catalog interface. For detailed attribute definitions, refer to Appendix A. Attribute Definitions.

Table 5.6. True Dual-Port RAM Attributes for MachXO4 (DP8KC)

Attribute	Description	Values	Default Value	User-Selectable through IP Catalog
DATA_WIDTH_A	Data Word Width Port A	1, 2, 4, 9	9	Yes
DATA_WIDTH_B	Data Word Width Port B	1, 2, 4, 9	9	Yes
REGMODE_A	Register Mode Pipelining for Port A	NOREG, OUTREG	NOREG	Yes
REGMODE_B	Register Mode Pipelining for Port B	NOREG, OUTREG	NOREG	Yes
RESETMODE	Selects the Reset type.	ASYNC, SYNC	SYNC	Yes
CSDECODE_A	Chip Select Decode for Port A	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
CSDECODE_B	Chip Select Decode for Port B	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
WRITEMODE_A	Read/Write Mode for Port A	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	Yes
WRITEMODE_B	Read/Write Mode for Port B	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	Yes
GSR	Enables Global Set Reset.	ENABLE, DISABLE	DISABLED	No

Attribute	Description	Values	Default Value	User-Selectable through IP Catalog
INITVAL_00 .. INITVAL_1F	Initialization Value	0x00000000000000000000000000000000000000000000000000000000000000000000000000000000 .... 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF (80-character hex strings)	0x00000000000000000000000000000000000000000000000000000000000000000000000000000000	No
ASYNC_RESET_RELEASE	Reset Release	ASYNC, SYNC	SYNC	Yes
INIT_DATA	Init Values Status	STATIC, DYNAMIC	STATIC	Yes

5.3. Pseudo Dual-Port RAM (PDPW8KC) – EBR-Based

The Pseudo Dual-Port RAM primitive is shown in Figure 5.3.

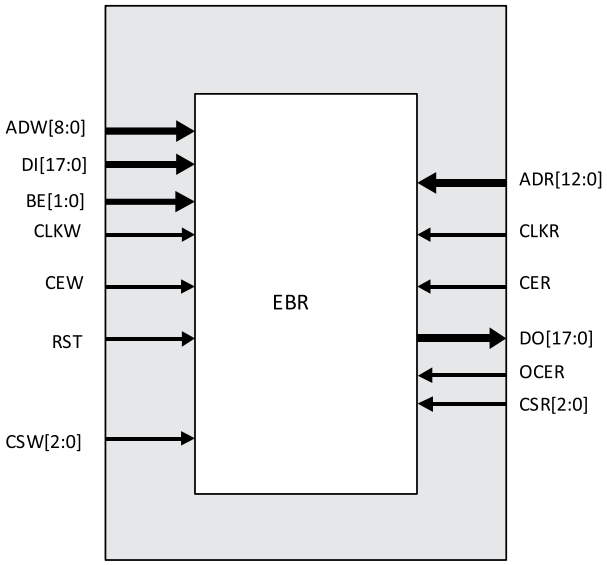

Figure 5.3. Pseudo Dual-Port RAM (PDPW8KC)

Table 5.7. EBR-Based Pseudo Dual-Port Memory Port Definitions

Port Name in the EBR Block Primitive (PDPW8KC)	Description	Active State
ADW	Write Address	—
DI	Write Data	—
BE	Byte Enable	Active High
CLKW	Write Clock	Rising Edge
CEW	Write Clock Enable	Active High
RST	Reset	Active High
CSW	Write Chip Select	—
ADR	Read Address	—
CLKR	Read Clock	Rising Edge
CER	Read Clock Enable	Active High

Port Name in the EBR Block Primitive (PDPW8KC)	Description	Active State
DO	Read Data	—
OCER	Read Output Clock Enable	Active High
CSR	Read Chip Select	—

Each PDPW8KC primitive consists of 9,216 bits of RAM. The possible values for address depth and data width for the PDPW8KC primitive are listed in Table 5.8.

Table 5.8. Pseudo Dual-Port Memory Sizes for 9K Memory in MachXO4 Devices

Pseudo-Dual Read Port Memory Size	Write Data Port	Read Data Port	Read Address Port [MSB:LSB]	Write Address Port [MSB:LSB]
8K x 1	DI[17:0]	DO	ADR[12:0]	ADW[8:0]
4K x 2	DI[17:0]	DO[1:0]	ADR[12:1]	ADW[8:0]
2K x 4	DI[17:0]	DO[3:0]	ADR[12:2]	ADW[8:0]
1K x 9	DI[17:0]	DO[8:0]	ADR[12:3]	ADW[8:0]
512 x 18	DI[17:0]	DO[17:0]*	ADR[12:4]	ADW[8:0]

*__Note__: High and low bytes are swapped with regard to DI word.

Table 5.9 shows the various attributes available for the Pseudo Dual-Port Memory, RAM_DP. Some of these attributes are user-selectable through the IP Catalog interface. For detailed attribute definitions, refer to Appendix A. Attribute Definitions.

Table 5.9. Pseudo Dual-Port RAM Attributes for MachXO4 (PDPW8KC)

Attribute	Description	Values	Default Value	User-selectable through IP Catalog
DATA_WIDTH_W	Write Data Word Width	18	18	Yes
DATA_WIDTH_R	Read Data Word Width	1, 2, 4, 9, 18	9	Yes
REGMODE	Register Mode Pipelining	NOREG, OUTREG	NOREG	Yes
RESETMODE	Selects the Reset type.	ASYNC, SYNC	SYNC	Yes
CSDECODE_W	Chip Select Decode for Write	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
CSDECODE_R	Chip Select Decode for Read	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
GSR	Enables Global Set Reset.	ENABLE, DISABLE	DISABLE	No
INITVAL_00 .. INITVAL_1F	Initialization Value	0x000000000000000000000000000000000000000000000000000000000000000000000000000000 .... 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFF (80- character hex strings)	0x000000000000000000000000000000000000000000000000000000000000000000000000000000	No
ASYNC_RESET_RELEASE	Reset Release	ASYNC, SYNC	SYNC	Yes
INIT_DATA	Init Values Status	STATIC, DYNAMIC	STATIC	Yes

5.4. Dual-Clock FIFO (FIFO8KB) - EBR-Based

The Dual-Clock FIFO RAM primitive is shown in Figure 5.4.

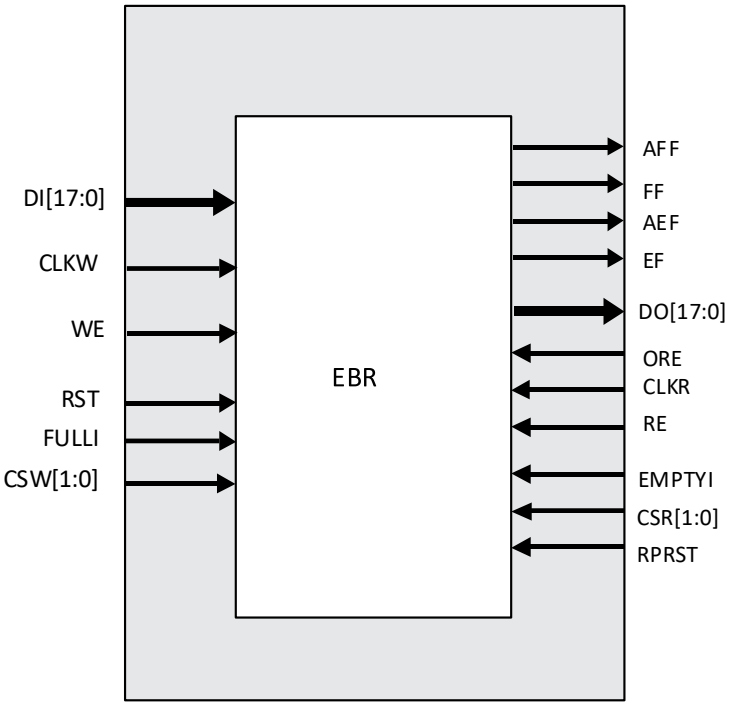

Figure 5.4. FIFO_DC Primitive (FIFO8KB)

Table 5.10. EBR-Based Dual-Clock FIFO Memory Port Definitions

Port Name in Primitive(FIFO8KB)	Description	Active State
DI	Data Input	—
CLKW	Write Port Clock	Rising Clock Edge
WE	Write Enable	Active High
FULLI	Write inhibit	Active High
CSW	Write Chip Select	Active High
AFF	Almost Full Flag	Active High
FF	Full Flag	Active High
AEF	Almost Empty	Active High
EF	Empty Flag	Active High
DO	Data Output	—
ORE	Output Read Enable	Active High
CLKR	Read Port Clock	Rising Clock Edge
RE	Read Enable	Active High
EMPTYI	Read inhibit	Active High
CSR	Read Chip Select	Active High
RPRST	Read Pointer Reset	Active High

Each FIFO8KB primitive consists of 9,216 bits of RAM. The possible values for address depth and data width for the FIFO8KB primitive are listed in Table 5.11.

Table 5.11. MachXO4 Dual-Clock FIFO Data Widths Sizes

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

Table 5.12 shows the various attributes available for the FIFO_DC. Some of these attributes are user-selectable through the IP Catalog interface. For detailed attribute definitions, refer to Appendix A. Attribute Definitions.

Table 5.12. Dual-Clock FIFO Attributes for MachXO4 (FIFO8KB)

Attribute	Description	Values	Default Value	User-Selectable through IP Catalog
DATA_WIDTH_W	Data Width Write Mode	1, 2, 4, 9, 18	18	YES
DATA_WIDTH_R	Data Width Read Mode	1, 2, 4, 9, 18	18	YES
REGMODE	Register Mode	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects Reset Type.	ASYNC, SYNC	ASYNC	YES
CSDECODE_W	Chip Select Decode for Write Mode	0b00, 0b01, 0b10, 0b11	0b00	NO
CSDECODE_R	Chip Select Decode for Read Mode	0b00, 0b01, 0b10, 0b11	0b00	NO
GSR	Enables Global Set Reset.	ENABLED, DISABLED	DISABLED	NO
AEPOINTER	Almost Empty Pointer	0b00000000000000, ..., 0b01111111111111	—	YES
AFPOINTER	Almost Full Pointer	0b00000000000000, ..., 0b01111111111111	—	YES
FULLPOINTER	Full Pointer	0b00000000000000, ..., 0b10000000000000	—	YES
FULLPOINTER1	Full Pointer minus 1	0b00000000000000, ..., 0b01111111111111	—	NO
AFPOINTER1	Almost Full Pointer minus 1	0b00000000000000, ..., 0b01111111111110	—	NO
AEPOINTER1	Almost Empty Pointer plus 1	0b00000000000000, ..., 0b10000000000000	—	NO
ASYNC_RESET_RELEASE	Reset Release	ASYNC, SYNC	SYNC	Yes
INIT_DATA	Init Values Status	STATIC, DYNAMIC	STATIC	Yes

5.4.1. FIFO_DC Flags

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

The program ranges for the four FIFO_DC flags are specified in Table 5.13.

Table 5.13. FIFO_DC Flag Settings

Module Flag Name	FIFO Attribute Name	Description	Programming Range	Program Bits
Full	FULLPOINTER	Full setting	1 to (2N - 1)	14

Module Flag Name	FIFO Attribute Name	Description	Programming Range	Program Bits
	FULLPOINTER1	Full – 1	1 to (FULL-1)	14
AlmostFull	AFPOINTER	Almost full setting	1 to (FULL -1)	14
	AFPOINTER1	Almost full – 1	1 to (FULL -1)	14
	AEPOINTER1	Almost empty + 1	1 to (FULL -1)	14
AlmostEmpty	AEPOINTER	Almost empty setting	1 to (FULL -1)	14
Empty	—	Empty setting	0	—

The value of Empty is fixed at 0. When coming out of reset, the active high flags Empty and Almost Empty are set to high, since they are true.

Careful attention is required to set the Pointer attributes to match the desired behavior. Refer to Appendix B. Setting FIFO_DC Pointer Attributes.

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

5.5. Distributed SPRAM (SPR16X4C) – PFU-Based

The PFU based distributed Single-Port RAM primitive is shown below.

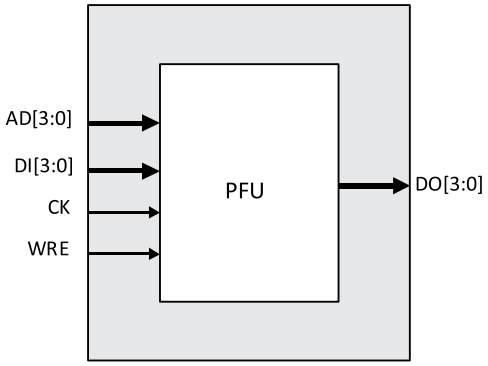

Figure 5.5. Distributed_SPRAM Primitive (SPR16X4C)

Table 5.14. PFU-based Distributed Single-Port RAM Port Definitions

Port Name in the PFU Primitive	Description	Active State
AD[3:0]	Address	—
DI[3:0]	Data Input	—
CK	Clock	Rising Clock Edge
WRE	Write Enable	Active High
DO[3:0]	Data Out	—

5.6. Distributed DPRAM (DPR16X4C) – PFU-Based

The PFU-based distributed Pseudo Dual-port RAM primitive is below.

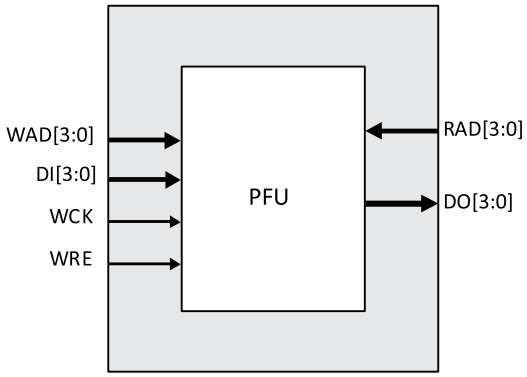

Figure 5.6. Distributed_DPRAM Primitive (DPR16X4C)

Table 5.15. PFU-based Distributed Dual-Port RAM Port Definitions

Port Name in the EBR Block Primitive	Description	Active State
WAD[3:0]	Write Address	—
DI[3:0]	Data Input	—
WCK	Write Clock	Rising Clock Edge
WRE	Write Enable	Active High
RAD[3:0]	Read Address	—
DO[3:0]	Data Out	—

5.7. Distributed ROM (ROMnnnX1A) – PFU-Based

The PFU based distributed ROM primitives are shown in Figure 5.8.

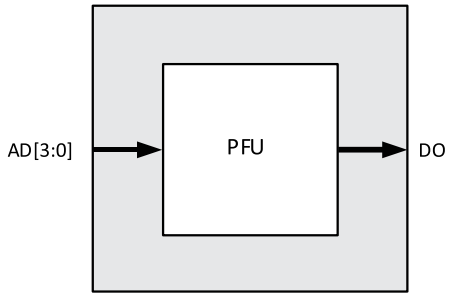

Figure 5.7. Distributed_ROM Primitive (ROM16X1A)

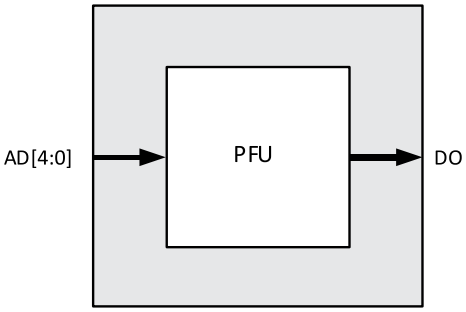

Figure 5.8. Distributed_ROM Primitive (ROM32X1A)

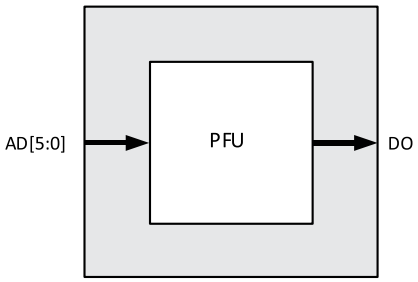

Figure 5.9. Distributed_ROM Primitive (ROM64X1A)

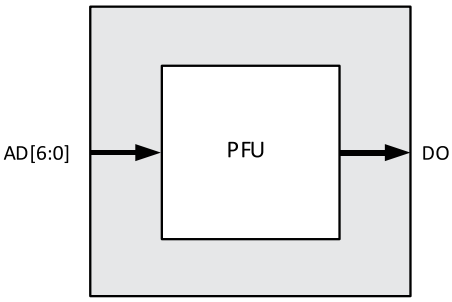

Figure 5.10. Distributed_ROM Primitive (ROM128X1A)

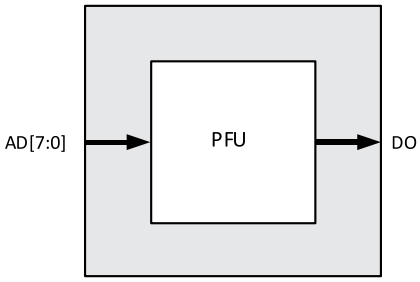

Figure 5.11. Distributed_ROM Primitive (ROM256X1A)

Table 5.16. PFU-Based Distributed ROM Port Definitions

Port Name in the PFU Block Primitive	Description
AD[n:0]	Address
DO	Data Out

6. 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 the two values: 0 or 1.

6.1. Initialization File Format

The initialization file is an ASCII file, which can be created or edited using any ASCII editor. IP Catalog supports three different types of memory file formats:

- Binary file
- Hex File

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, respectively.

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

6.1.1. Binary File

The file is a text file of 0's and 1's. The rows indicate the number of words and columns indicate the width of the memory. In the example below, the memory size is 20 x 32, which is 20 addresses with each word of 32-bit length.

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

6.1.2. Hex File

The Hex file is 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. In

In the example below, the memory size is 8x16, which is eight addresses with each word of 16-bit length.

```
A001
0B03
1004
CE06
0007
040A
0017
02A4
```

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 you 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, is 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 the chip select decode for write and CSDECODE_R is the 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 H and the other tries to write L.

It is recommended that you 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 or Reset attribute, is used to enable or disable the global set or reset for the RAM element.

A.7. ASYNC_RESET_RELEASE

When RESETMODE is set to ASYNC, the ASYNC_RESET_RELEASE attribute allows you to select how the reset is deasserted or released. When set to SYNC, the reset is deasserted synchronously to the clock. When set to ASYNC, the memory reset is released asynchronously, without relation to the clock.

A.8. INIT_DATA

The INIT_DATA attribute allows you 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. When set to DYNAMIC, the initialization values are not compressed, and stored in a user-accessible, fixed location in UFM.

Appendix B. Setting FIFO_DC Pointer Attributes

The FIFO_DC uses pointer attributes to control the Full, Almost Full, and Almost Empty flags.

The values for the pointer attributes are set according to the following table and equations:

Table B.1. Pointer Attribute Setting Equations

Flag	Trip Vale	Attribute	Port Width	Equation
Full	ff	FULLPOINTER	wrw[1]	[(ff - 1) * wrw] + 1
		FULLPOINTER1		[(ff - 2) * wrw] + 1
Almost Full	Aff	AFPOINTER	wrw[1]	[(aff - 1) * wrw] + 1
		AFPOINTER1		[(aff - 2) * wrw] + 1
Almost Empty	aef	AEPOINTER	rdw[1]	[(aef) * rdw] + rdw - 1
		AEFOINTER1		[(aef + 1) * rdw] + rdw - 1

Note: Set the Write Port Width wrw and the Read Port Width rdw upon Table B.2.

Table B.2. Port Width Values

Attribute: Data_width_w, Data_width_r	Port Width: wrw, rdw
1	1
2	2
4	4
9	8
18	16

You should specify the absolute value of the address at which the Almost Empty and Almost Full flags go true. For example, if the Almost Full flag is required to go true at the address location 500 for a FIFO of depth 512, you should specify aff = 500.

Worked Example:

```
Write Data Width: 18 => use wrw=16
Read Data Width: 4 => use rdw = 4
Full: ff =16 ( (16) 18-bit words)
Almost Full: aff = 14
Almost Empty: aef = 8 ( (8) 4-bit words, which corresponds to (2) 16-bit writes)
Empty: 0 (always)
```

Calculated Values:

```
FULLPOINTER = [(ff - 1) * wrw] + 1
= [(16 – 1) * 16] + 1
= (15 * 16) + 1
= 241
=> 14’ b00_0000_1111_0001
FULLPOINTER1 = [(ff - 2) * wrw] + 1
= [(16 – 2) * 16] + 1
= (14 * 16) + 1
= 225
=> 14’ b00_0000_1110_0001
AFPOINTER = [(aff - 1) * wrw] + 1
= [(14 – 1) * 16] + 1
= (13 * 16) + 1
= 209
=> 14’ b00_0000_1101_0001
AFPOINTER1 = [(aff - 2) * wrw] + 1
= [(14 – 2) * 16] + 1
= (12 * 16) + 1
```

```
= 193
=> 14’ b00_0000_1100_0001
AEPOINTER = [(aef) * rdw] + rdw - 1
= (8 * 4) + 4 - 1
= (8 * 4) + 3
= 35
=> 14’ b00_0000_0010_0011
AEFOINTER1 = [(aef + 1) * rdw] + rdw - 1
= [(8+1) * 4] + 4 - 1
= (9 * 4) + 3
= 39
=> 14’ b00_0000_0010_0111
```

References

- MachXO4 Devices Web Page
- MachXO4 Family Data Sheet (FPGA-DS-02125)
- MachXO4 Hardened Control Functions User Guide (FPGA-TN-02403)
- Lattice Insights for Lattice Semiconductor training courses 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 1.0, December 2025

Section	Change Summary
All	Updated the document version from 0.80 to 1.0.

Revision 0.80, December 2025

Section	Change Summary
All	Initial preliminary release.

www.latticesemi.com