ECP5 and ECP5-5G Memory Usage Guide

Technical Note

FPGA-TN-02204-1.4

June 2021

Disclaimers

Contents

Figures

Tables

1. Introduction

This technical note discusses memory usage for the ECP5™ and ECP5-5G™ family of FPGA devices. It is intended to be used by design engineers as a guide to integrating the EBR- and PFU-based memories for this device family in Lattice Diamond®.

The architecture of these devices provides many resources for memory-intensive applications. The sysMEM™ Embedded Block RAM (EBR) compliments its distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM and ROM memories can be constructed using the EBR. The LUTs and PFUs can implement Distributed Single-Port RAM, Dual-Port RAM and ROM. Look-up Tables (LUTs) within PFUs can implement Distributed Single-Port RAM, Dual-Port RAM and ROM.

The capabilities of the EBR Block RAM and PFU RAM are referred in this document. Designers can utilize the memory primitives in three separate ways:

- Via **Clarity Designer** – The Clarity Designer GUI allows users to specify the memory type and size required. The Clarity Designer takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.
- Via **PMI (Parameterizable Module Inferencing)** – PMI allows experienced users to skip the graphical interface and utilize the Configurable memory primitives on-the-fly from the Lattice Diamond® project navigator. The parameters and the control signals needed either in Verilog or VHDL can be set. The top-level design will have the parameters defined and signals declared so the interface can automatically generate the black box during synthesis.
- Via the **Instantiation of Memory Primitives** – Memory primitives are called directly by the top-level module and instantiated in the user’s design. This is an advanced method and requires a thorough understanding of memoryhook-ups and design interfaces.

The remainder of this document discusses these approaches.

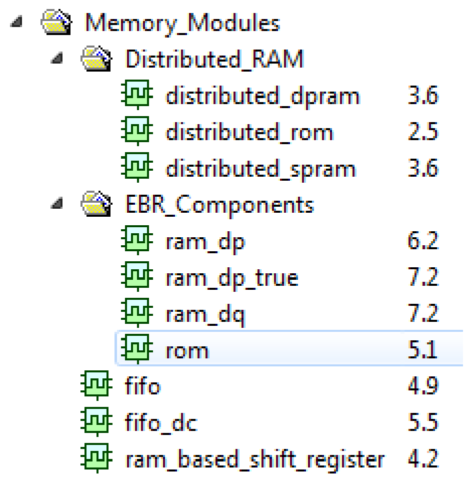
2. Memory Generation

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

The available modules in the Clarity Designer are:

- Distributed Memory Modules
  - Distributed Dual Port RAM (Distributed_DPRAM)
  - Distributed ROM (Distributed_ROM)
  - Distributed Single Port RAM (Distributed_SPRAM)
- EBR Components (or EBR based Modules)
  - Dual PORT RAM (RAM_DP_TRUE)
  - Pseudo Dual Port RAM (RAM_DP)
  - Single Port RAM (RAM_DQ)
  - Read Only Memory (ROM)
- First In First Out Memory (FIFO and FIFO_DC)
- RAM Based Shift Register

Figure 2.1 shows the memory modules under Clarity Designer in Lattice Diamond software.



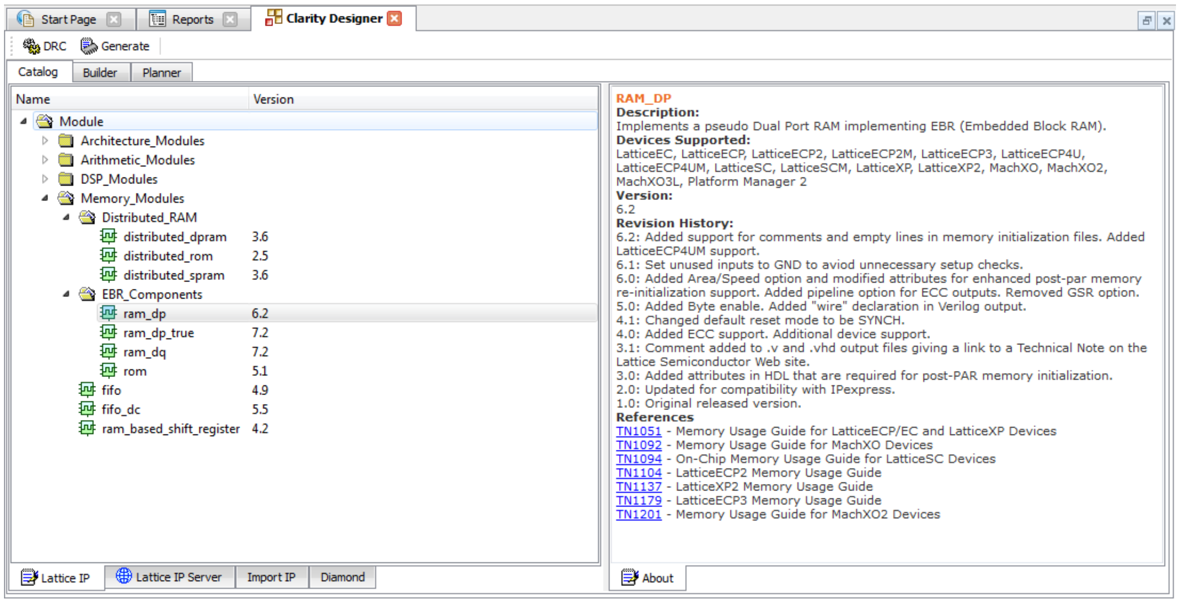
```
Memory_Modules
  Distributed_RAM
    distributed_dpram        3.6
    distributed_rom          2.5
    distributed_spram        3.6
  EBR_Components
    ram_dp                   6.2
    ram_dp_true              7.2
    ram_dq                   7.2
    rom                      5.1
  fifo                       4.9
  fifo_dc                    5.5
  ram_based_shift_register   4.2
```

Figure 2.1. Memory Modules Available in Clarity Designer

2.1. Clarity Designer Flow

Clarity Designer allows users to generate, create (or open) any of the above modules for ECP5 and ECP5-5G devices. From the Lattice Diamond software, select Tools > Clarity Designer.

Alternatively, you can click on the button in the toolbar. This opens the Clarity Designer window as shown in Figure 2.2.



Name	Version
Module	
Architecture_Modules	
Arithmetic_Modules	
DSP_Modules	
Memory_Modules	
Distributed_RAM	
distributed_dpram	3.6
distributed_rom	2.5
distributed_spram	3.6
EBR_Components	
ram_dp	6.2
ram_dp_true	7.2
ram_dq	7.2
rom	5.1
fifo	4.9
fifo_dc	5.5
ram_based_shift_register	4.2

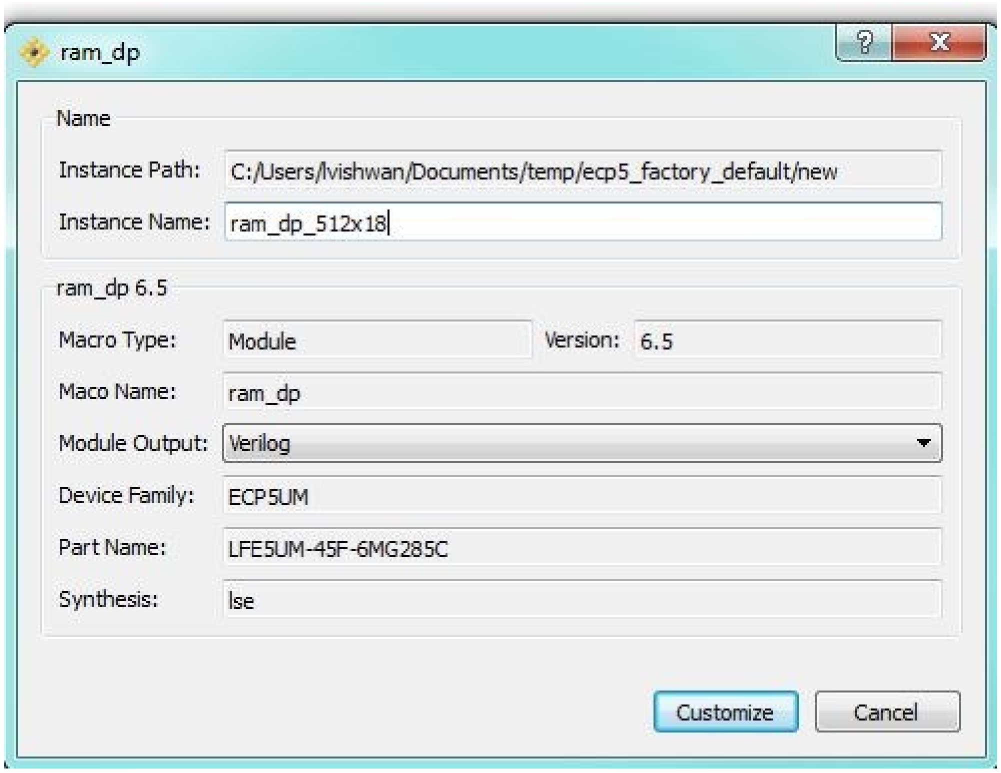
RAM_DP
Description:
Implements a pseudo Dual Port RAM implementing EBR (Embedded Block RAM).
Devices Supported:
LatticeEC, LatticeECP, LatticeECP2, LatticeECP2M, LatticeECP3, LatticeECP4U, LatticeECP4UM, LatticeSC, LatticeSCM, LatticeXP, LatticeXP2, MachXO, MachXO2, MachXO3L, Platform Manager 2
Version:
6.2
Revision History:
6.2: Added support for comments and empty lines in memory initialization files. Added LatticeECP4UM support.
6.1: Set unused inputs to GND to aviod unnecessary setup checks.
6.0: Added Area/Speed option and modified attributes for enhanced post-par memory re-initialization support. Added pipeline option for ECC outputs. Removed GSR option.
5.0: Added Byte enable. Added "wire" declaration in Verilog output.
4.1: Changed default reset mode to be SYNCH.
4.0: Added ECC support. Additional device support.
3.1: Comment added to .v and .vhd output files giving a link to a Technical Note on the Lattice Semiconductor Web site.
3.0: Added attributes in HDL that are required for post-PAR memory initialization.
2.0: Updated for compatibility with IPexpress.
1.0: Original released version.
References
TN1051 - Memory Usage Guide for LatticeECP/EC and LatticeXP Devices
TN1092 - Memory Usage Guide for MachXO Devices
TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices
TN1104 - LatticeECP2 Memory Usage Guide
TN1137 - LatticeXP2 Memory Usage Guide
TN1179 - LatticeECP3 Memory Usage Guide
TN1201 - Memory Usage Guide for MachXO2 Devices

Figure 2.2. Clarity Designer in Lattice Diamond Software

The left section of the Clarity Designer window includes the Module Tree. The Memory Modules are categorized as Distributed RAM, EBR Components and FIFOs. The right section of the window shows the description of the module selected and links to the documentation to find more details about it.

Let us look at an example of generating an EBR based Pseudo Dual Port RAM of size 512 x 18.

Double-click ram_dp under the EBR_ Components. Fill out the information of the module to generate and click the Customize button. This is shown in Figure 2.3.



Name

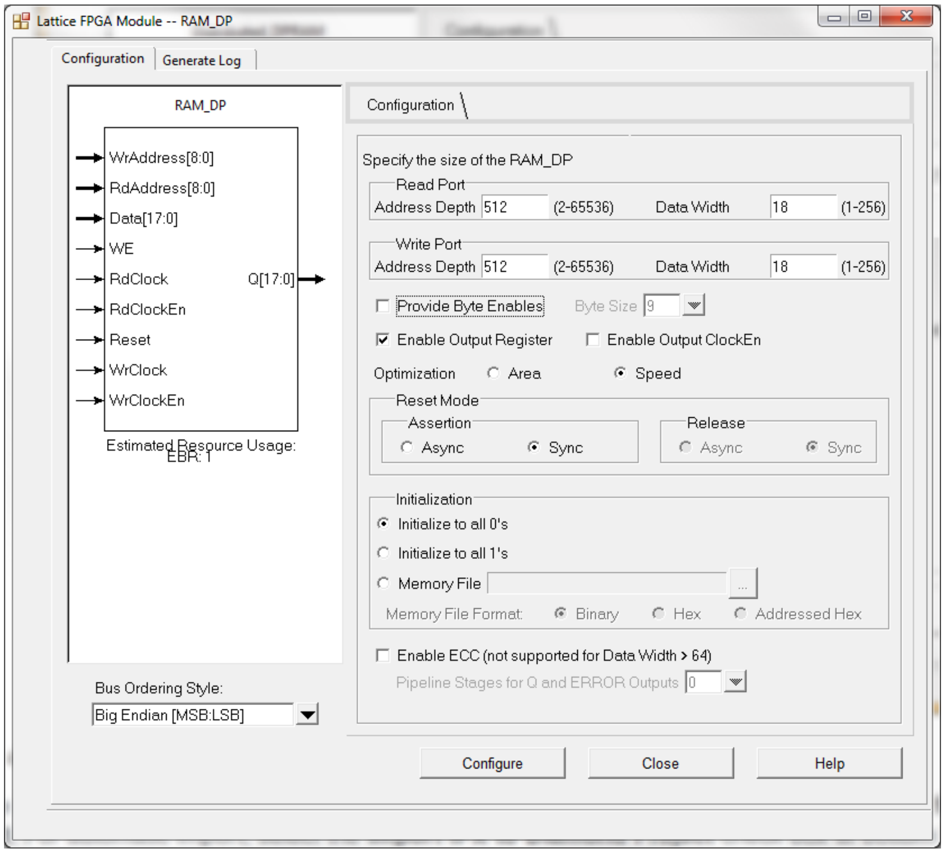
Instance Path: C:/Users/lvishwan/Documents/temp/ecp5_factory_default/new
Instance Name: ram_dp_512x18

ram_dp 6.5

Macro Type: Module
Version: 6.5
Maco Name: ram_dp
Module Output: Verilog
Device Family: ECP5UM
Part Name: LFE5UM-45F-6MG285C
Synthesis: lse

Figure 2.3. Example: Generating Pseudo Dual Port RAM (RAM_DP) Using Clarity Designer

Clicking Customize opens another window, as shown in Figure 2.4, where you can customize the Distributed DPRAM.



Specify the size of the RAM_DP

Read Port: Address Depth 512 (2-65536), Data Width 18 (1-256)

Write Port: Address Depth 512 (2-65536), Data Width 18 (1-256)

☐ Provide Byte Enables — Byte Size 9

☑ Enable Output Register — ☐ Enable Output ClockEn

Optimization: Area [✓ Speed]

Reset Mode — Assertion: Async [✓ Sync]; Release: Async [✓ Sync]

Initialization: [✓ Initialize to all 0's] Initialize to all 1's; Memory File

Memory File Format: [✓ Binary] Hex; Addressed Hex

☐ Enable ECC (not supported for Data Width > 64)

Pipeline Stages for Q and ERROR Outputs 0

Bus Ordering Style: Big Endian [MSB:LSB]

Figure 2.4. Example: Generating Pseudo Dual Port RAM (RAM_DP) Module Customization – General Options

When all the options of the module being generated are filled in, click Generate.

This module, once in the Diamond project, can be instantiated within other modules.

2.2. Utilizing PMI

The parameters and control signals needed can be set in either Verilog or VHDL. The top-level design includes the defined memory parameters and declared signals. The interface can then automatically generate the black box during synthesis and Diamond 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 on-line help system.

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

2.3. Utilizing Direct instantiation of Memory Primitives

Another way to use the memories in the designs is by directly instantiating the memory primitives for the ECP5 and ECP5-5G devices. When instantiating the primitives, users have to work at the EBR block level. In case there is a need to have a memory that spans multiple modules, users are required to create the cascading memory on their own.

Lattice provides a detailed list of all the primitives in a VHDL/ Verilog file under the cae_library/synthesis folder in Lattice Diamond software installation folder.

3. Memory Features

The RAMs can be generated with Error Correction and Byte Enables that mask selective bits. These features are available in the EBR based RAM modules.

ECC in Memory Modules

An error-correcting code (ECC) code is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors were introduced, either during the process of transmission, or on storage.

EBR based Memory modules Clarity Designer allows users to implement ECC. There is a check box to enable ECC in the Configuration tab for the module.

Enable ECC check box allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths higher than 64 bits.

The two bits indicate the error if any, and the following shows you what each of these bits mean:

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

ECC is adding 0~2 cycle of delay to both the data and ERROR outputs base on the Clarity Designer GUI setting. The data and ERROR output are always in sync.

One of the things to note is that the ECC is added in the PFU logic, and hence there logic implemented outside the EBR block and can cause some speed impact. The effective speed at which the memory runs should be determined by running the Trace report.

3.1. Byte Enable

Byte Enable is a feature available in the selected RAM modules where users can mask the bytes written in the RAM. The Byte Enable control can be per 8-bits or 9-bits; the selection can be made in Clarity Designer while generating the module.

Note that the Byte Enable option is available for the memory widths higher than 8-bits (1 byte).

Each bit of the BE signal corresponds to the corresponding 8-bit or 9-bit selection, starting from LSB side. For example, if we add Byte Enable to an 18-bit wide RAM, then Table 3.1 explains how the written data (Data In) is masked for an 8-bit or 9-bit Byte Size.

Table 3.1. Masked Data in Bits for an 8-Bit or 9-Bit Byte Size

Byte Enable Bit	Data In Bits that Get Masked (with 8-bit Byte Size)	Data In bits that Get Masked (with 9-Bit Byte Size)
ByteEn(0)	Data(7:0)	Data(8:0)
ByteEn(1)	Data(13:8)	Data(15:9)
ByteEn(2)	Data(19:14)	Data(19:16)

It is to be noted that the ByteEn and ECC are mutually exclusive and they cannot be used together.

4. Memory Modules

The following sections discuss the different modules, the size of memory that each EBR block or the Distributive primitive can support and any special options for the module.

When a user specifies the width and depth of the memory in the Clarity Designer, the tool generates the memory by depth cascading and/ or width cascading or EBR blocks or Distributed RAM primitives. Clarity Designer automatically allows users to create memories larger than the width and depth supported for each primitive.

4.1. Memory Cascading

For any memory sizes smaller than that can fit in a single EBR block or the Distributed primitive, the module will utilize the complete block or primitive.

For memory sizes larger than that of a single module, the multiple modules will be cascaded (either in depth or width) to create a larger module.

4.1.1. Input & Output Register

The architecture of the EBR blocks in ECP5 and ECP5-5G devices are designed such that the inputs that go into the memory are always registered. This means that the input data and address are always registered at the input of the memory array. The output data of the memory is optionally registered at the output. The user can choose this option by selecting the Enable Output Register check box in Clarity Designer while customizing the module.

Control signals like WE and Byte Enable are also registered going in to the EBR block.

4.1.2. Reset

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

4.1.3. Timing

In order to correctly write into a memory cell in the EBR block, the correct address has to be registered by the logic. Hence it is important to note that while running the trace on the EBR blocks, there should be no setup and hold time violations on the address registers (address). Failing to meet these requirements can result in incorrect addressing and hence corruption of memory contents.

During a read cycle, a similar issue can occur that the correct contents are not read if the address is not correctly registered in the memory.

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

4.2. Single Port RAM (RAM_DQ) – EBR Based

ECP5 and ECP5-5G FPGAs supports all the features of Single Port Memory Module or RAM_DQ. Clarity Designer allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement.

Clarity Designer generates the memory module, as shown in Figure 4.1.

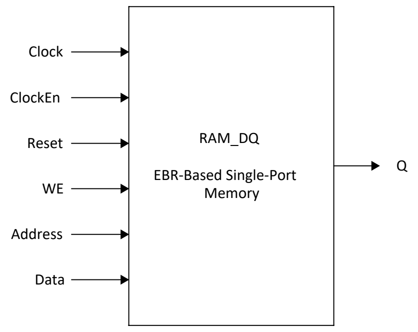


Figure 4.1. Single-Port Memory Module Generated by Clarity Designer

Figure 4.2 provides the primitive that can be instantiated for the Single Port RAM. Primitive name is SP16KD and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 18K bits of memory; if the memory required is larger than 18K, then cascading can be used, using the CS port (CSA and CSB in this case). See Appendix A. Attribute Definitions CSDECODE section on the cascading of multiple EBR primitives.

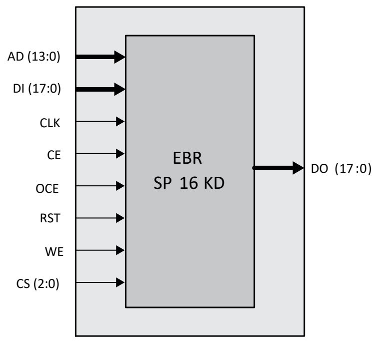


Figure 4.2. Single Port RAM Primitive for ECP5 and ECP5-5G Devices

The various ports and their definitions for Single-Port Memory are listed in Table 4.1. The table lists the corresponding ports for the module generated by Clarity Designer and for the EBR RAM_DQ primitive.

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

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

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

Table 4.2. Single-Port Memory Sizes for 18K Memories in ECP5 and ECP5-5G Devices

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

The ones that do not have selectable option in Clarity Designer are handled by the engine. However, users working with the direct primitive instantiation can access these options.

Table 4.3. Single-Port Memory Sizes for 18K Memories in ECP5 and ECP5-5G Devices

Configuration Tab Attributes	Description	Values	Default Value
Address Depth	Address depth of the read and write port.	2—<Max that can fit in the device>	512
Data Width	Data word width of the Read and write port.	1 — 256	36
Enable Output Register	Data Out port (Q) can be registered or not using this selection.	TRUE, FALSE	TRUE
Enable Output ClockEn	Clock Enable for the output clock (this option requires Enabling Output Register).	TRUE, FALSE	FALSE
Byte Enables	Allows users to select Byte Enable options.	TRUE, FALSE	FALSE
Byte Size	Byte Size selection when Byte Enable option is selected.	9, 8	9
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	ASYNC, SYNC	SYNC
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock.	ASYNC, SYNC	SYNC
Optimization	Design optimizations to configure EBR for Speed or Area.	Area, Speed	Speed
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file.	0's, 1's, File	0's
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.	—	—
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats).	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE
Advanced Tab Attributes	**Description**	**Values**	**Default Value**
Write Mode	Option to select different write modes. Check Timing waveforms for the behavior of RAMs in these modes.	NORMAL, WRITE THROUGH, READ BEFORE WRITE	NORMAL

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

Additionally, users can select to enable the output registers for RAM_DQ. The waveforms in the figures in the following pages show the internal timing waveforms for the Single Port RAM (RAM_DQ) with these options.

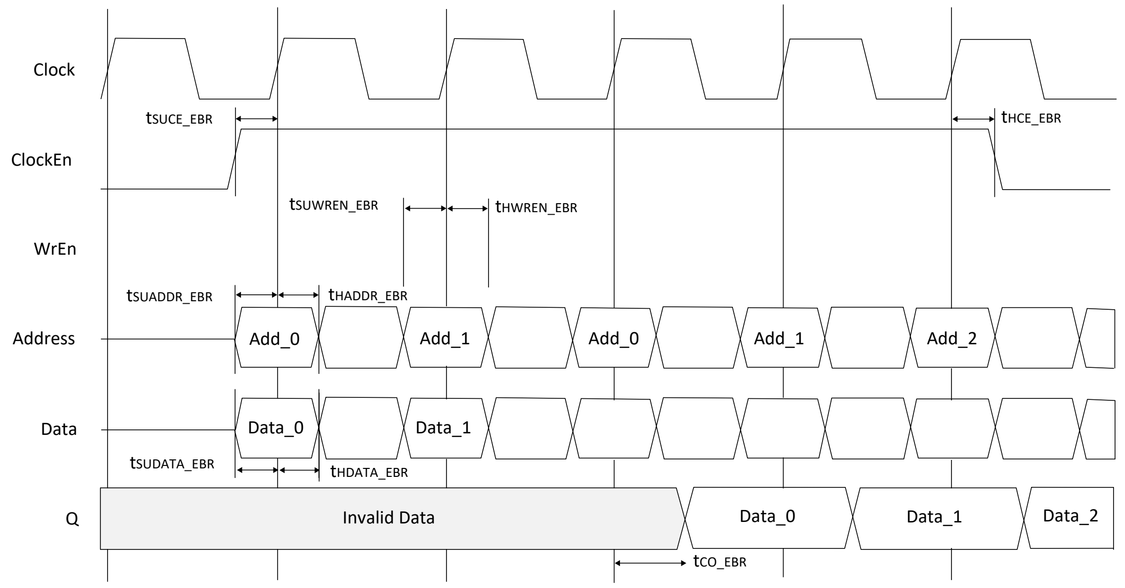


Figure 4.3. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, without Output Registers

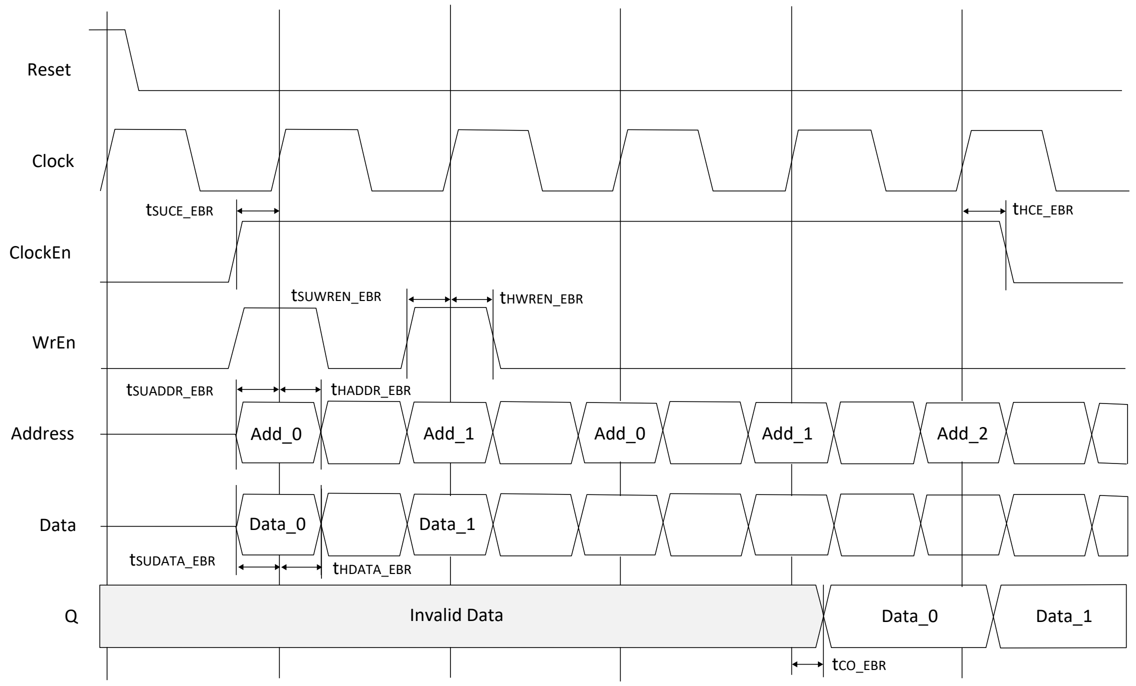


Figure 4.4. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, with Output Registers

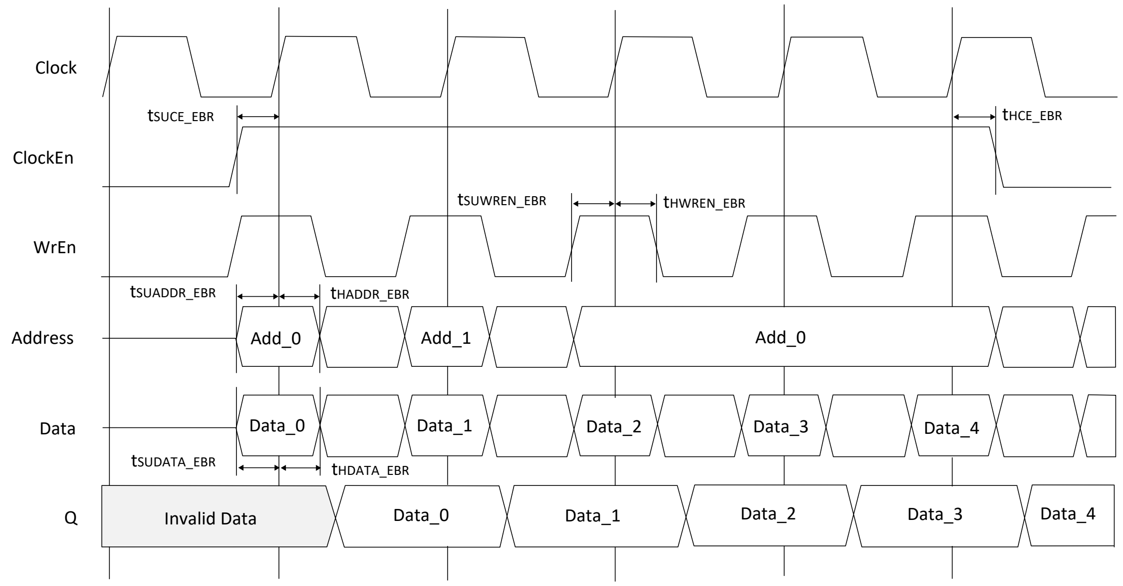


Figure 4.5. Single Port RAM Timing Waveform in Write through (WRITETHROUGH) Mode, without Output Registers

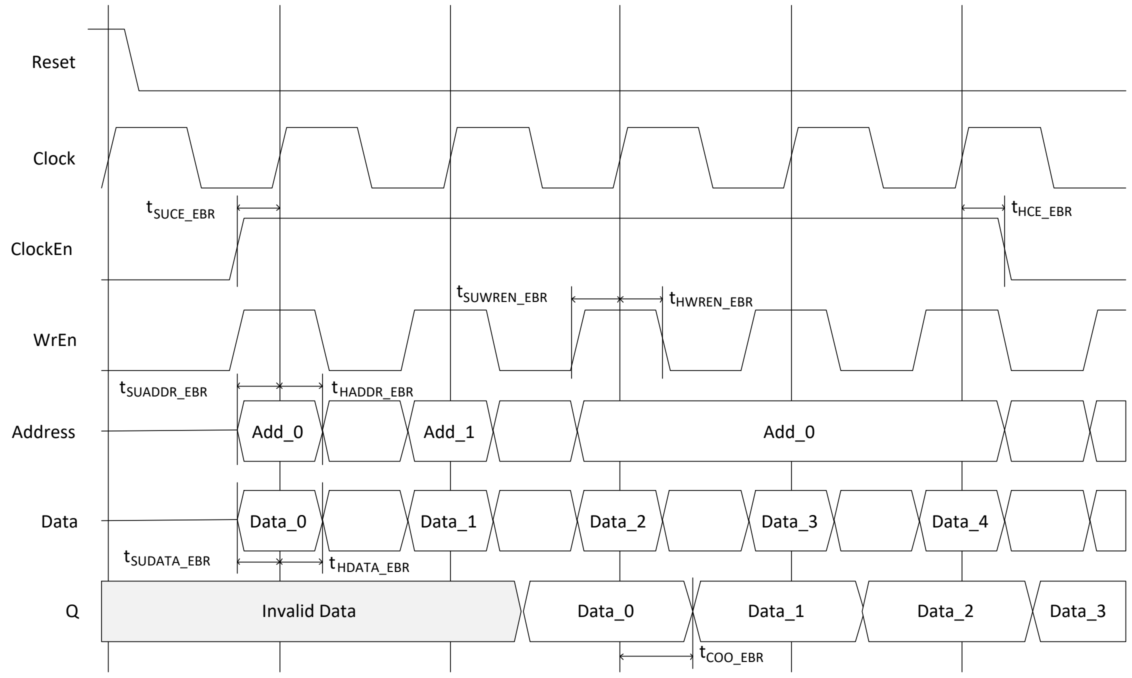


Figure 4.6. Single Port RAM Timing Waveform in Write through (WRITETHROUGH) Mode, with Output Registers

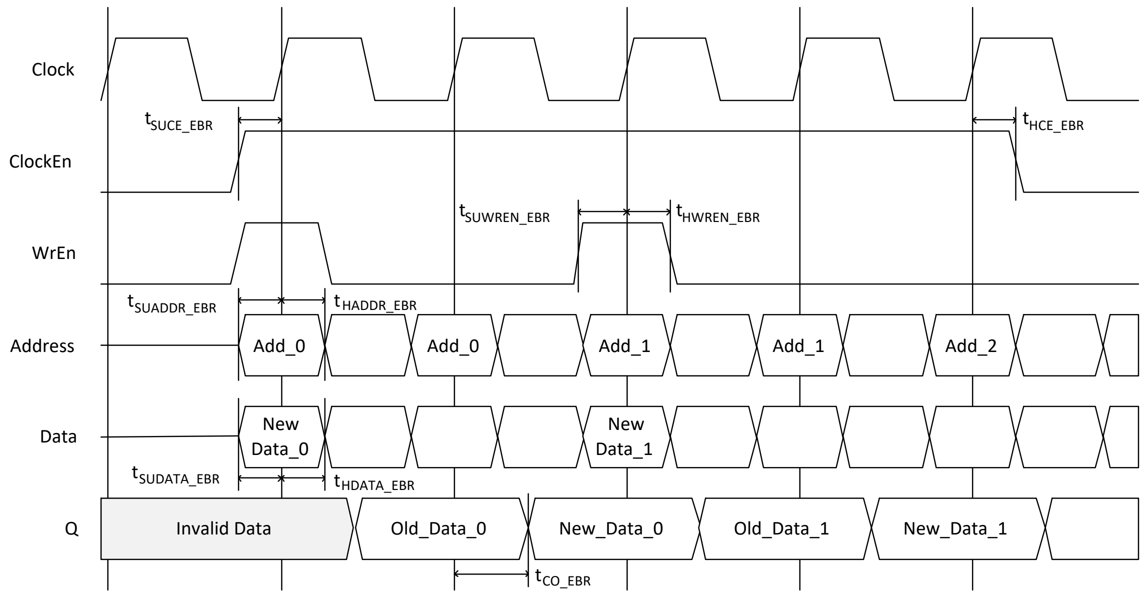


Figure 4.7. Single Port RAM Timing Waveform in Read before Write (READBEFOREWRITE) Mode, without Output Registers

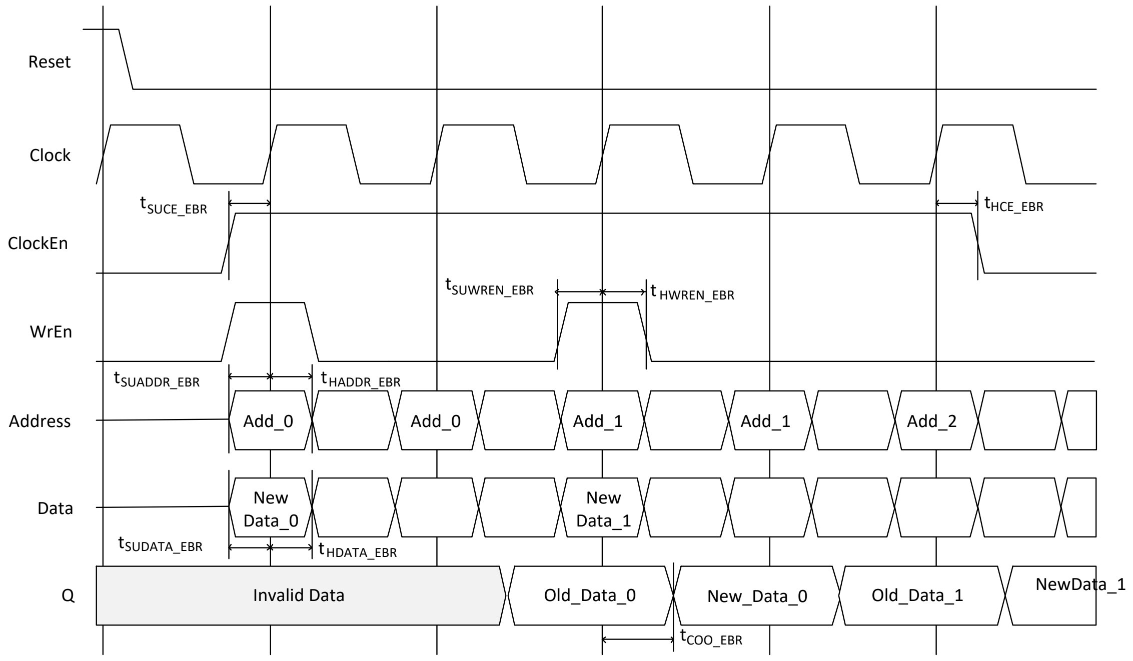


Figure 4.8. Single Port RAM Timing Waveform in Read before Write (READBEFOREWRITE) Mode, with Output Registers

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

The EBR blocks in the ECP5 and ECP5-5G devices can be configured as True-Dual Port RAM or RAM_DP_TRUE. Clarity Designer allows users to generate the Verilog-HDL, VHDL or EDIF netlists for the memory size as per design requirements.

Clarity Designer generates the memory module, as shown in Figure 4.9.

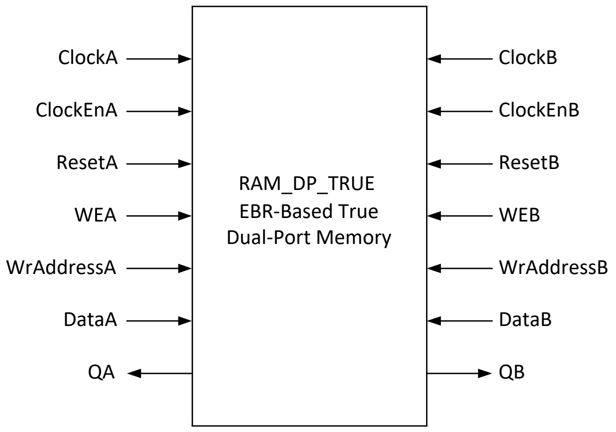


Figure 4.9. True Dual-Port Memory Module Generated by Clarity Designer

Figure 4.10 provides the primitive that can be instantiated for the True Dual Port RAM. Primitive name is DP16KD and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 18K bits of memory; if the memory required is larger than 18K, then cascading can be used, using the CS port (CSA and CSB in this case). Check Appendix A for CSDECODE section on how to do cascading of multiple EBR primitives.

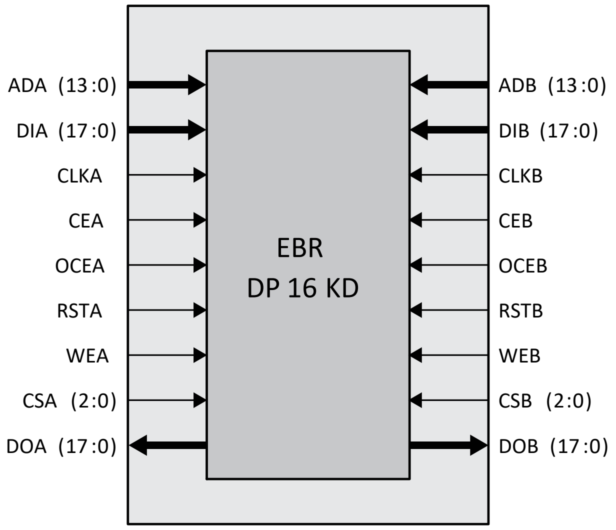


Figure 4.10. True Dual Port RAM Primitive for ECP5 and ECP5-5G Devices

The various ports and their definitions for True Dual-Port RAM are listed in Table 4.4. The table lists the corresponding ports for the module generated by Clarity Designer and for the EBR RAM_DP_TRUE primitive.

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

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
ClockA, ClockB	CLKA, CLKB	Clock for PortA and PortB
ClockEnA, ClockEnB	CEA, CEB	Clock Enables for Port CLKA and CLKB
AddressA, AddressB	ADA[w:0], ADB[x:0]	Address Bus port A and port B
DataA, DataB	DIA[y:0], DIB[z:0]	Input Data port A and port B
QA, QB	DOA[y:0], DOB[z:0]	Output Data port A and port B
WEA, WEB	WEA, WEB	Write enable port A and port B
ResetA, ResetB	RSTA, RSTB	Reset for PortA and PortB
—	CSA[2:0], CSB[2:0]	Chip Selects for each port

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

Table 4.5. Dual Port Memory Sizes for 18K Memory for ECP5 and ECP5-5G

Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Address Port A	Address Port B
16384 × 1	DataInA	DataInB	QA	QB	AddressA(13:0)	AddressB(13:0)
8192 × 2	DataInA(1:0)	DataInB(1:0)	QA(1:0)	QB(1:0)	AddressA(12:0)	AddressB(12:0)
4096 × 4	DataInA(3:0)	DataInB(3:0)	QA(3:0)	QB(3:0)	AddressA(11:0)	AddressB(11:0)
2049 × 9	DataInA(8:0)	DataInB(8:0)	QA(8:0)	QB(8:0)	AddressA(10:0)	AddressB(10:0)
1024 × 18	DataInA(17:0)	DataInB(17:0)	QA(17:0)	QB(17:0)	AddressA(9:0)	AddressB(9:0)

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

Table 4.6. True Dual-Port RAM Attributes for ECP5 and ECP5-5G Devices

Configuration Tab Attributes	**Description**	**Values**	**Default Value**
Port A Address Depth	Port A Address depth of the read and write port	2—<Max that can fit in the device>	512
Port A Data Width	Port A Data word width of the Read and write port	1 — 256	36
Port B Address Depth	Port B Address depth of the read and write port	2—<Max that can fit in the device>	512
Port B Data Width	Port B Data word width of the Read and write port	1 — 256	36
Port A Enable Output Register	Port A Data Out port (QA) can be registered or not using this selection.	TRUE, FALSE	TRUE
Port A Enable Output ClockEn	Port A Clock Enable for the output clock (this option requires Enabling Output Register)	TRUE, FALSE	FALSE
Port B Enable Output Register	Port B Data Out port (QB) can be registered or not using this selection.	TRUE, FALSE	TRUE
Port B Enable Output ClockEn	Port B Clock Enable for the output clock (this option requires Enabling Output Register)	TRUE, FALSE	FALSE
Byte Enables	Allows users to select Byte Enable options	TRUE, FALSE	FALSE
Byte Size	Byte Size selection when Byte Enable option is selected	9, 8	9
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Optimization	Design optimizations to configure EBR for Speed or Area	Area, Speed	Speed
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file.	0’s, 1’s, File	0’s
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.	—	—
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats).	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE
Advanced Tab Attributes	**Description**	**Values**	**Default Value**
Port A Write Mode	Option to select different write modes for Port A. Check Timing waveforms for the behavior of RAMs in these modes.	NORMAL, WRITE THROUGH, READ BEFORE WRITE	NORMAL
Port B Write Mode	Option to select different write modes for Port B. Check Timing waveforms for the behavior of RAMs in these modes.	NORMAL, WRITE THROUGH, READ BEFORE WRITE	NORMAL

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

Additionally, users can select to enable the output registers for RAM_DP_TRUE. Waveforms in the following figures show the internal timing waveforms for the True Dual Port RAM (RAM_DP_TRUE) with these options.

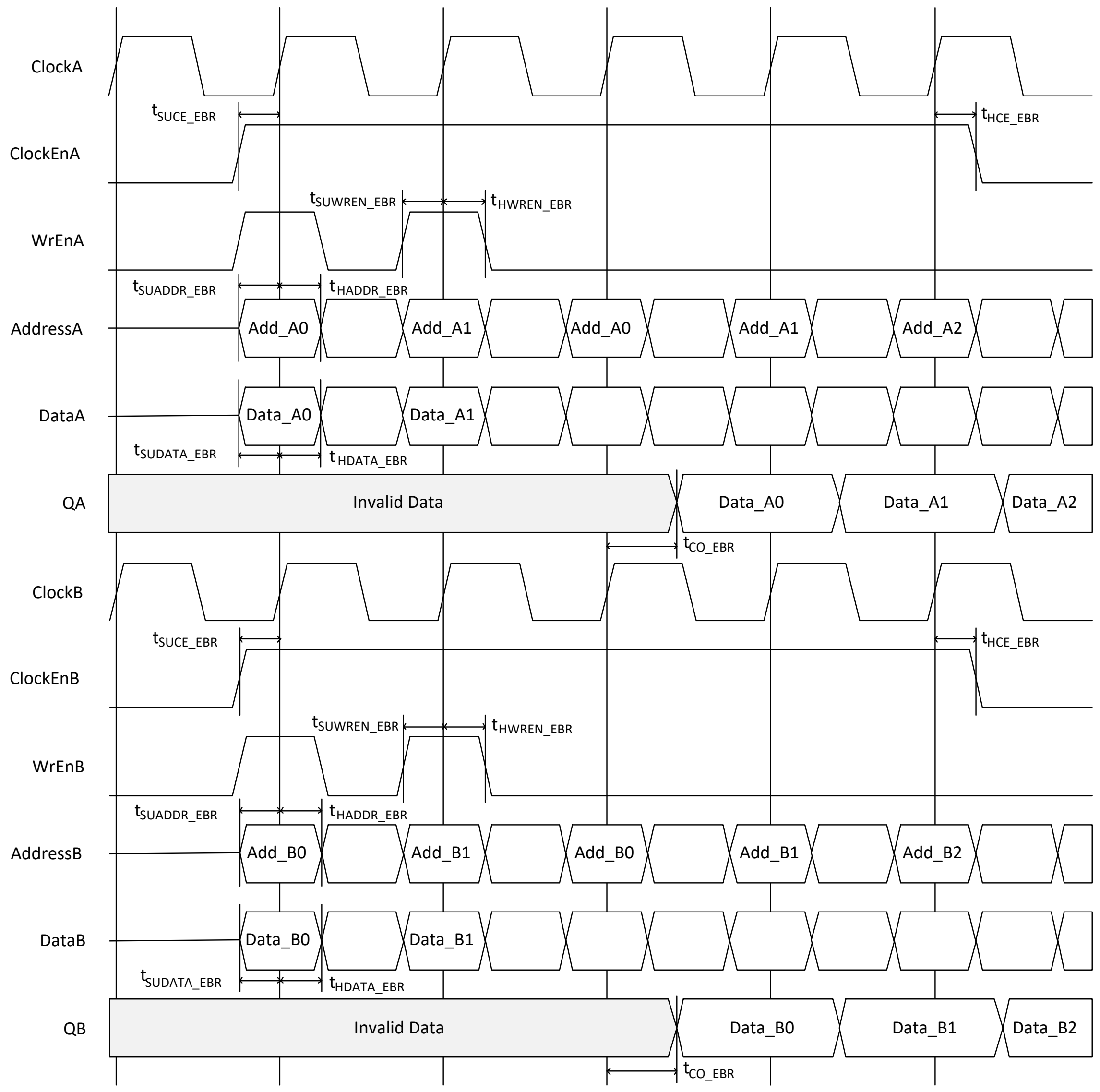


Figure 4.11. True Dual Port RAM Timing Waveform in Normal (NORMAL) Mode, without Output Registers

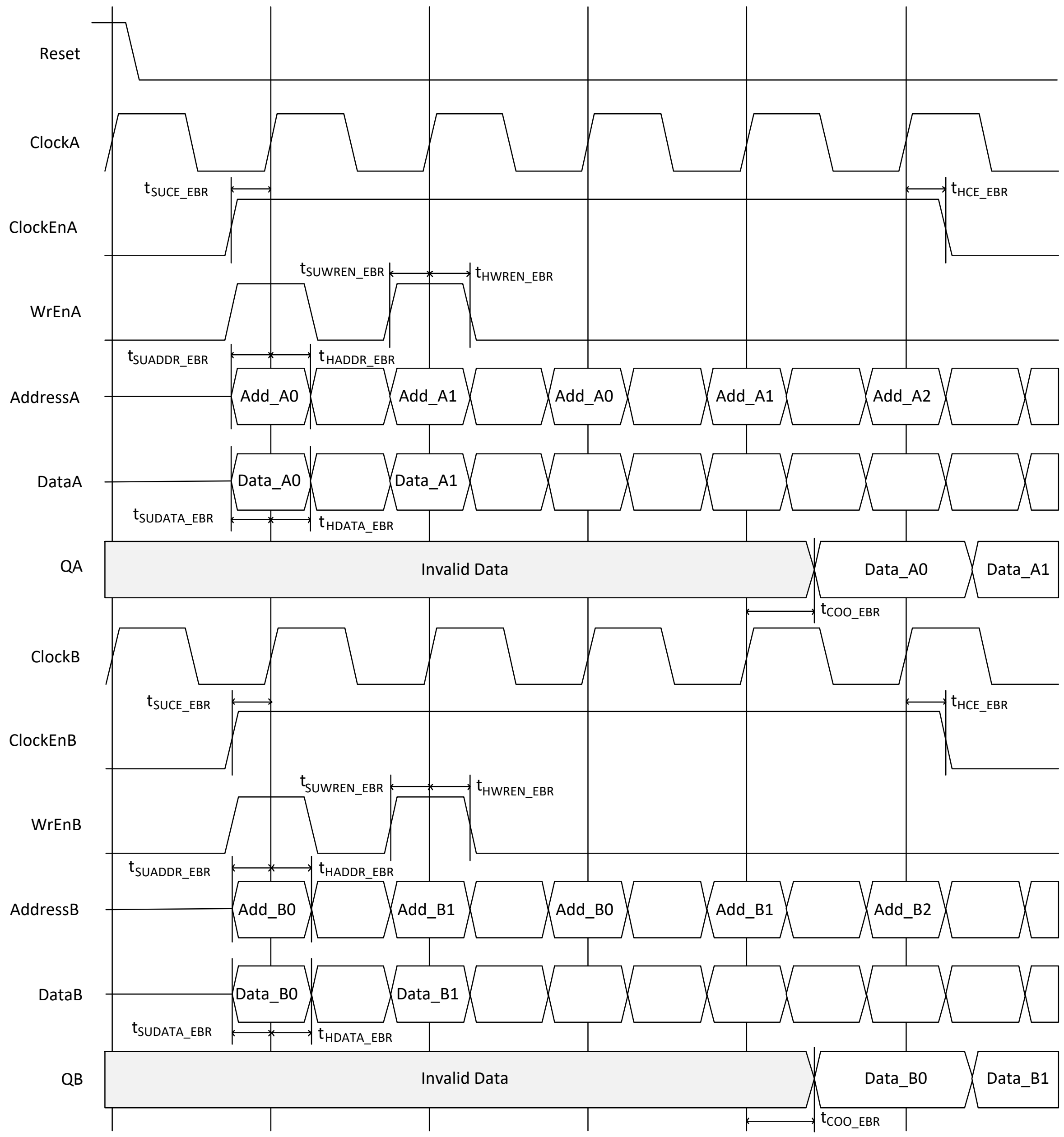


Figure 4.12. True Dual Port RAM Timing Waveform in Normal (NORMAL) Mode with Output Registers

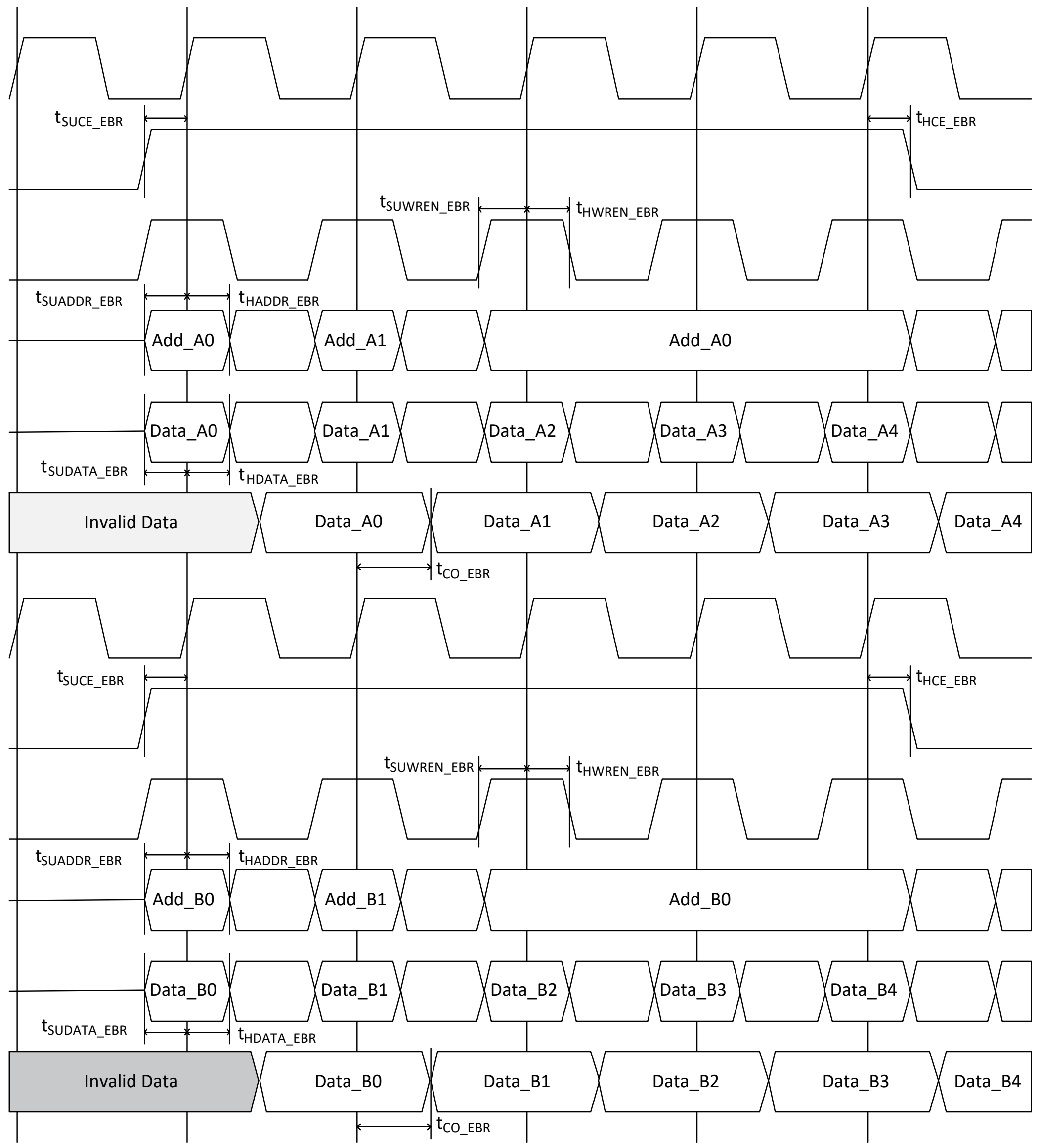


Figure 4.13. True Dual Port RAM Timing Waveform in Write through (WRITETHROUGH) Mode, without Output Registers

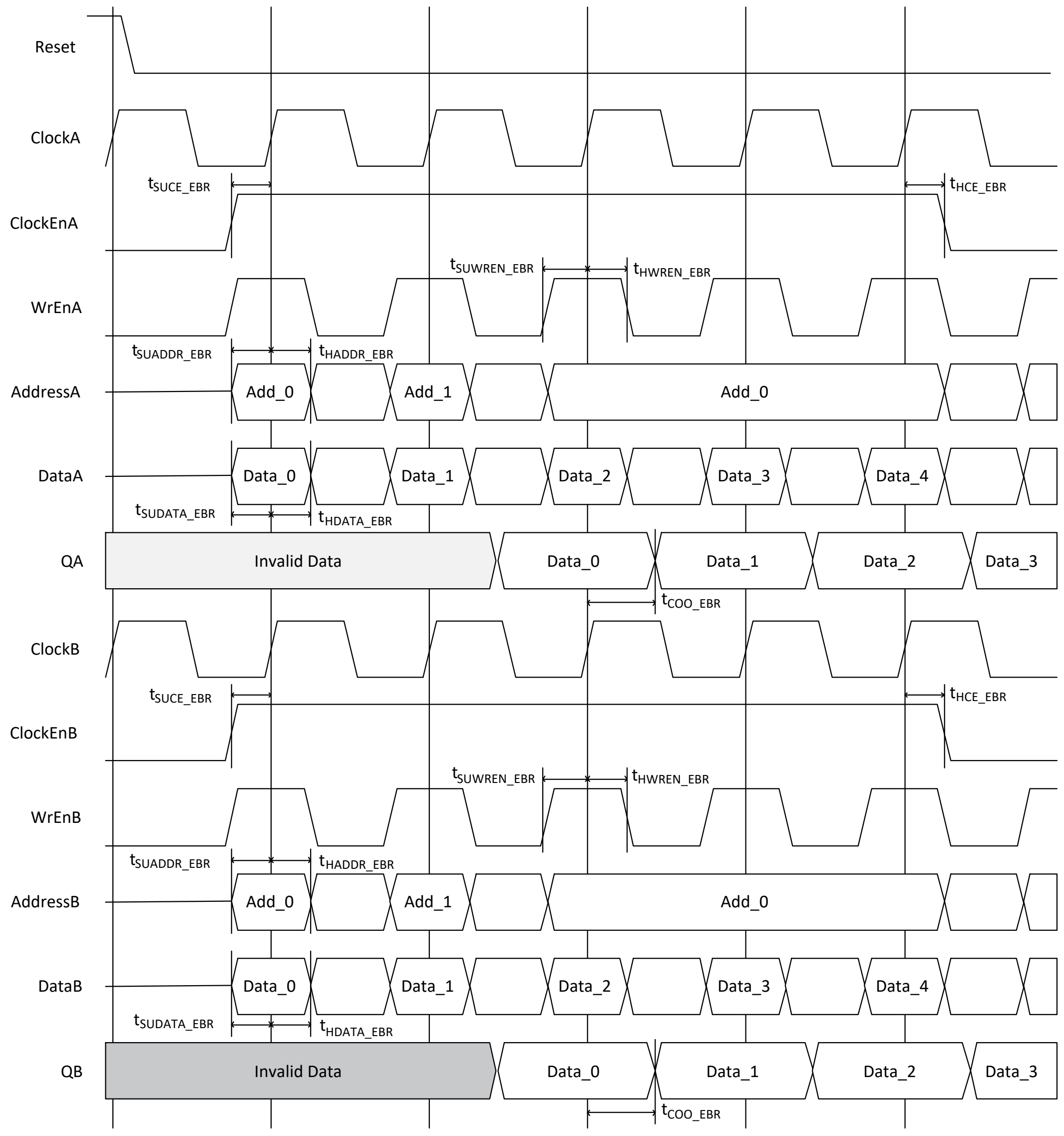


Figure 4.14. True Dual Port RAM Timing Waveform in Write through (WRITETHROUGH) Mode, with Output Registers

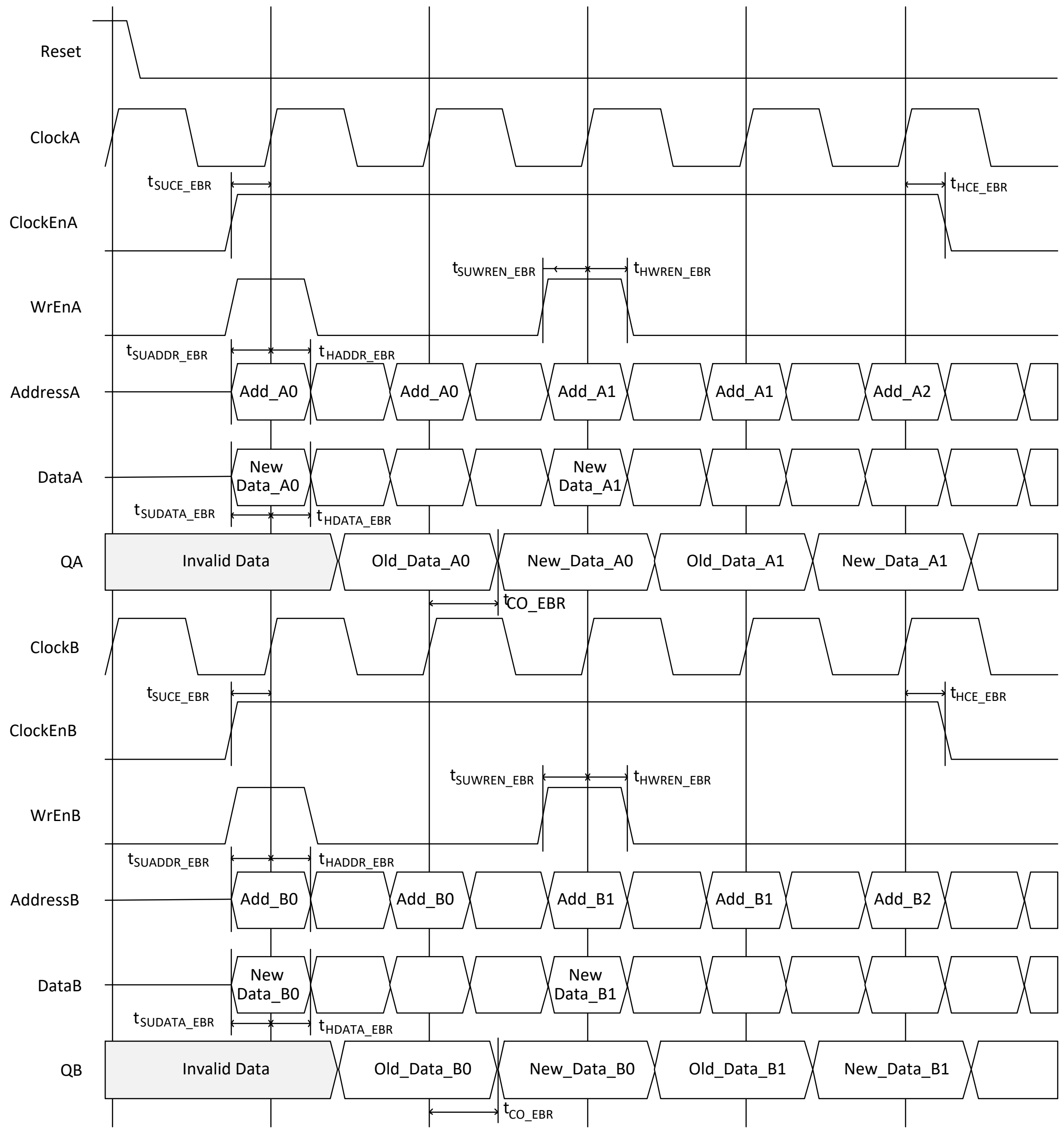


Figure 4.15. True Dual Port RAM Timing Waveform in Read before Write (READBEFOREWRITE) Mode, without Output Registers

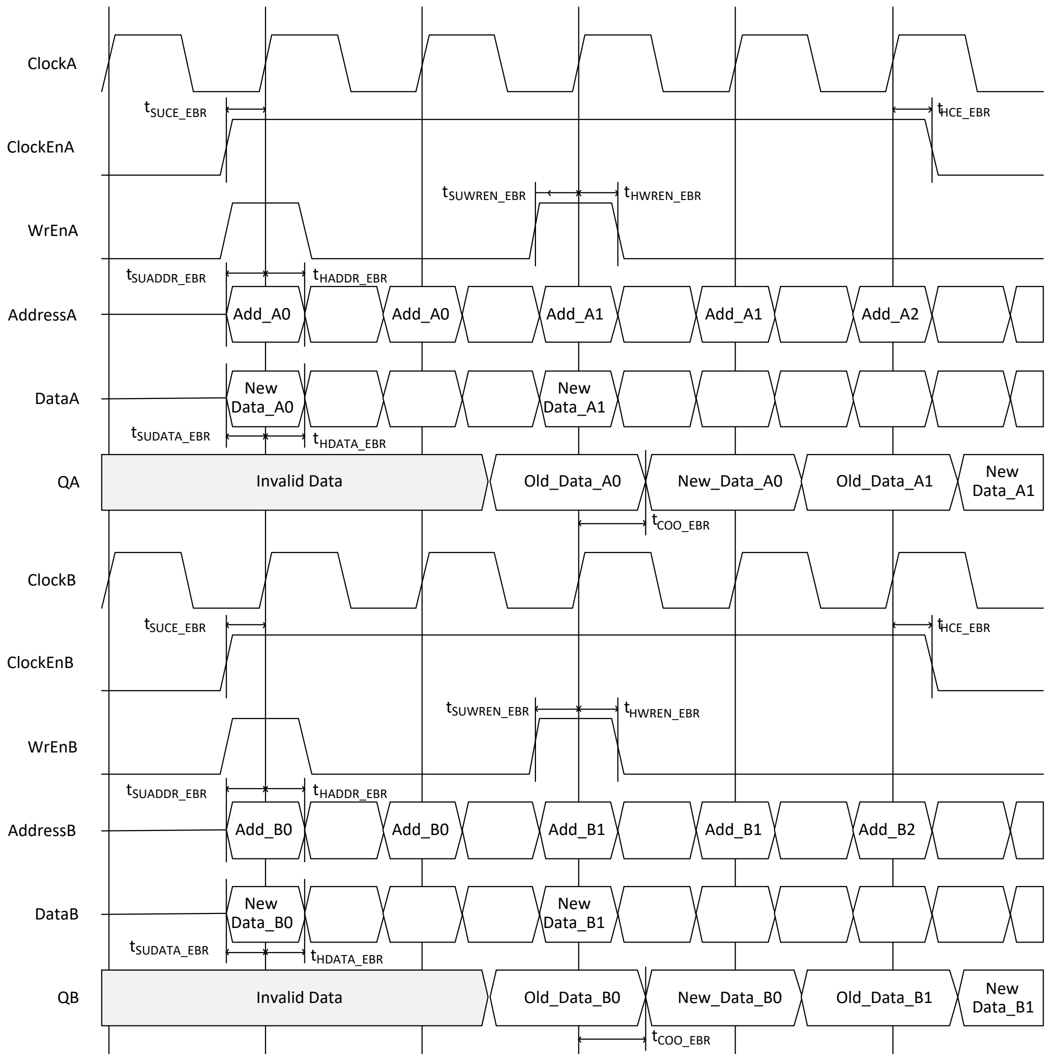


Figure 4.16. True Dual Port RAM Timing Waveform in Read before Write (READBEFOREWRITE) Mode, with Output Registers

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

ECP5 and ECP5-5G FPGAs supports all the features of Pseudo-Dual Port Memory Module or RAM_DP. Clarity Designer allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement.

Clarity Designer generates the memory module shown in Figure 4.17.

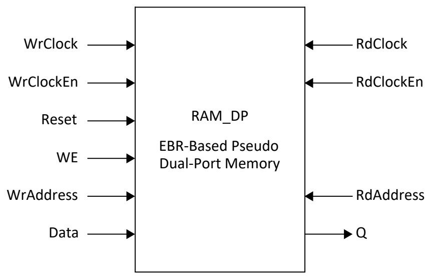


Figure 4.17. Pseudo Dual-Port Memory Module Generated by Clarity Designer

Figure 4.18 provides the primitive that can be instantiated for the Pseudo-Dual Port RAM. Primitive name is PDPW16KD and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 18K bits of memory; if the memory required is larger than 18K, then cascading can be used, using the CS port (CSA and CSB in this case). Check Appendix A for CSDECODE section on how to do cascading of multiple EBR primitives.

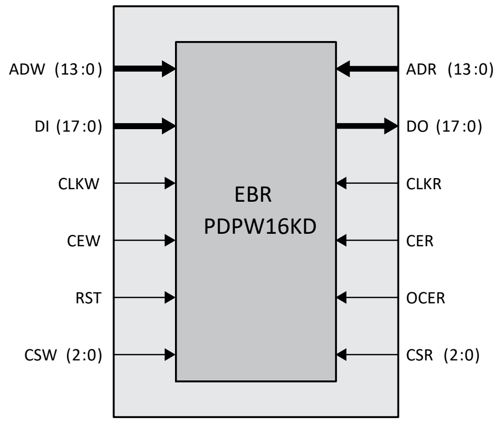


Figure 4.18. Pseudo-Dual Port RAM Primitive for ECP5 and ECP5-5G Devices

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

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

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
RdAddress	ADR[w:0]	Read Address
WrAddress	ADW[x:0]	Write Address
RdClock	CLKR	Read Clock
WrClock	CLKW	Write Clock
RdClockEn	CER	Read Clock Enable
WrClockEn	CEW	Write Clock Enable
Q	DO[y:0]	Read Data
Data	DI[z:0]	Write Data
WE	WE	Write Enable
Reset	RST	Reset
—	CS[2:0]	Chip Select

Each EBR block consists of 18,432 bits of RAM. The values for address (w and x) and data (y and z) of each EBR block are listed in Table 4.8.

Table 4.8. Pseudo-Dual Port Memory Sizes for 18K Memory for ECP5 and ECP5-5G Devices

Dual Port Memory Size	Input Data Write Port	Output Data Read Port	Address Write Port	Address Read Port
16384 × 1	Data	Q	WrAddress(13:0)	RdAddress(13:0)
8192 × 2	Data(1:0)	Q(1:0)	WrAddress(12:0)	RdAddress(12:0)
4096 × 4	Data(3:0)	Q(3:0)	WrAddress(11:0)	RdAddress(11:0)
2049 × 9	Data(8:0)	Q(8:0)	WrAddress(10:0)	RdAddress(10:0)
1024 × 18	Data(17:0)	Q(17:0)	WrAddress(9:0)	RdAddress(9:0)
512 × 36	Data(35:0)	Q(35:0)	WrAddress(8:0)	RdAddress(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 Clarity Designer GUI.

Table 4.9. Pseudo Dual-Port RAM Attributes for ECP5 and ECP5-5G Devices

Configuration Tab Attributes	Description	Values	Default Value
Read Port Address Depth	Read Port Address depth of the read and write port	2 — 65536	512
Read Port Data Width	Read Port Data word width of the Read and write port	1 — 256	35
Write Port Address Depth	Write Port Address depth of the read and write port	2 — 65536	512
Write Port Data Width	Write Port Data word width of the Read and write port	1 — 256	36
Enable Output Register	Data Out port (Q) can be registered or not using this selection.	TRUE, FALSE	TRUE
Enable Output ClockEn	Clock Enable for the output clock (this option requires Enabling Output Register)	TRUE, FALSE	FALSE
Byte Enables	Allows users to select Byte Enable options.	TRUE, FALSE	FALSE
Byte Size	Byte Size selection when Byte Enable option is selected.	9, 8	9
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Optimization	Design optimizations to configure EBR for Speed or Area	Area, Speed	Speed
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file.	0's, 1's, File	0's
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.	—	—
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats)	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE

Users have the option to enable the output registers for Pseudo-Dual Port RAM (RAM_DP). Waveforms in the following figures show the internal timing waveforms for Pseudo-Dual Port RAM (RAM_DP) with these options.

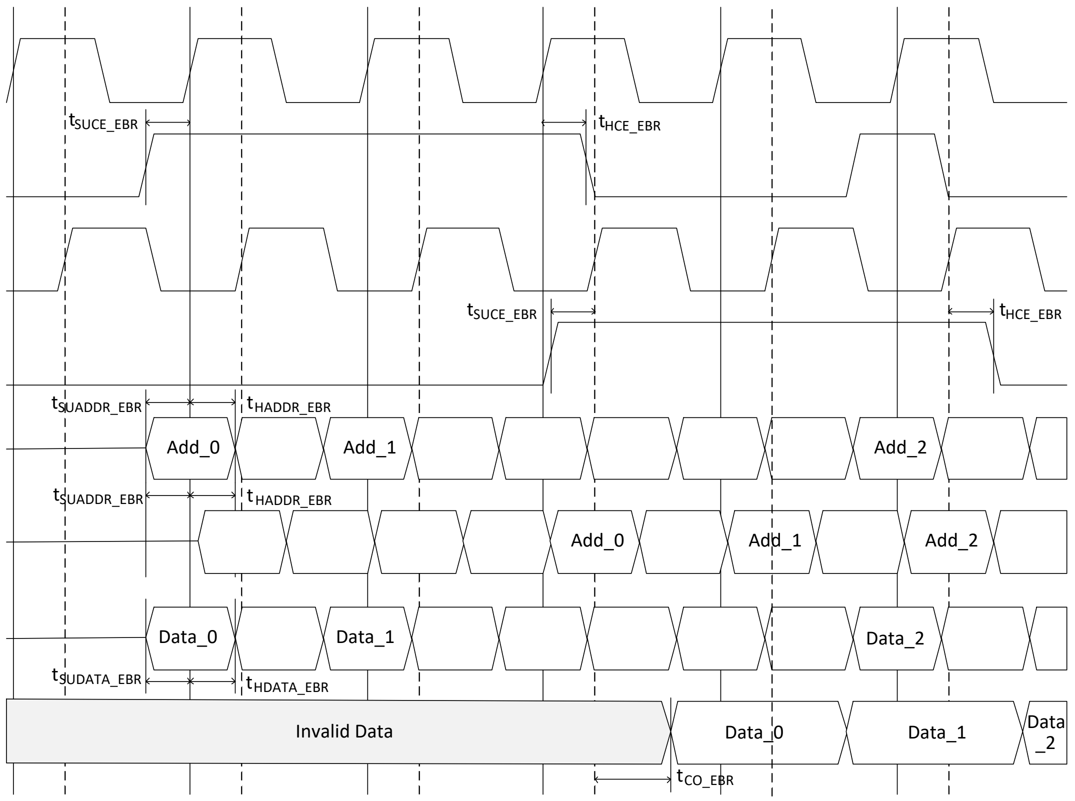


Figure 4.19. PSEUDO DUAL PORT RAM Timing Diagram - without Output Registers

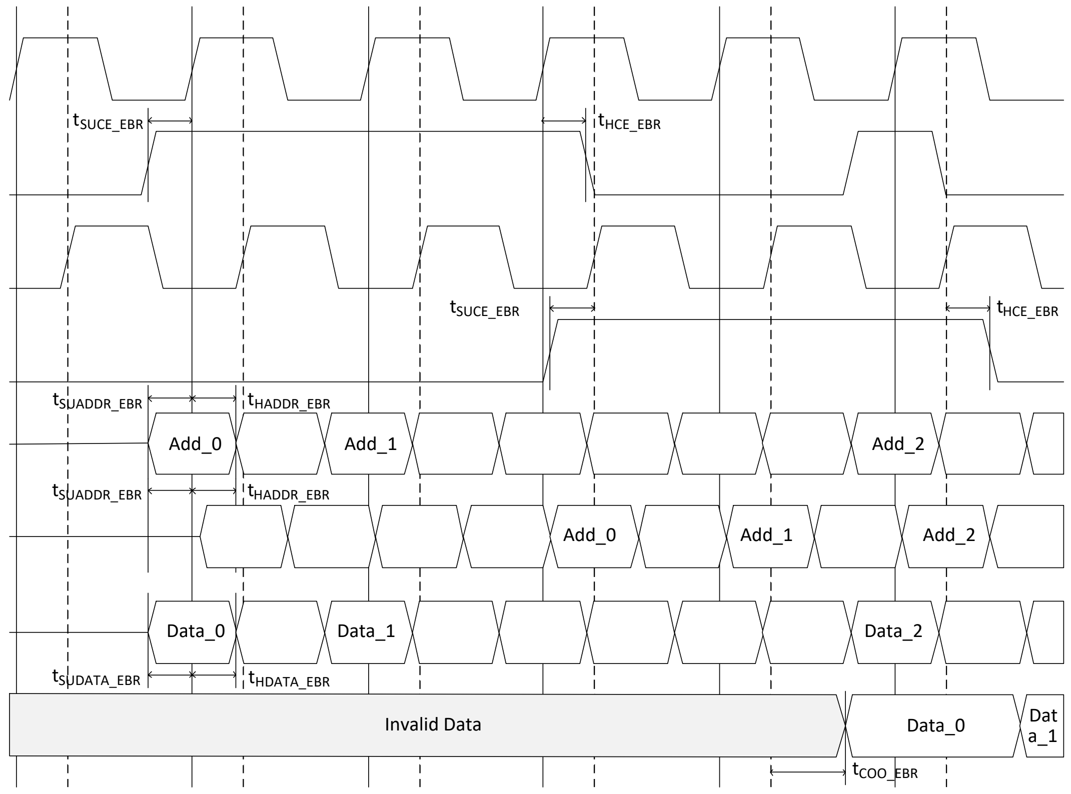


Figure 4.20. PSEUDO DUAL PORT RAM Timing Diagram - with Output Registers

4.5. Read Only Memory (ROM) – EBR Based

ECP5 and ECP5-5G FPGAs supports all the features of ROM Memory Module or ROM. Clarity Designer allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement. Users are required to provide the ROM memory content in a form of an initialization file.

Clarity Designer generates the memory module shown in Figure 4.21.

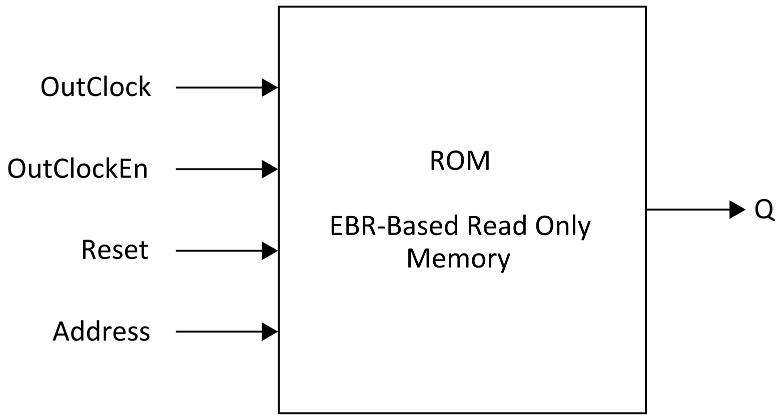


Figure 4.21. ROM – Read Only Memory Module Generated by Clarity Designer

The various ports and their definitions are listed in Table 4.10. The table lists the corresponding ports for the module generated by Clarity Designer and for the ROM primitive.

Table 4.10. EBR-Based ROM Port Definitions

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

When generating ROM using Clarity Designer, the designer must provide the initialization file to pre-initialize the contents of the ROM. These files are the *.mem files and they can be of binary, hex or addressed hex formats. The initialization files are discussed in detail in the Initializing Memory section of this document.

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 included in Table 4.11.

Table 4.11. ROM Memory Sizes for 16K Memory for ECP5 and ECP5-5G Devices

Dual Port Memory Size	Output Data Read Port	Address Write Port
16384 × 1	Q	WrAddress(13:0)
8192 × 2	Q(1:0)	WrAddress(12:0)
4096 × 4	Q(3:0)	WrAddress(11:0)
2049 × 9	Q(8:0)	WrAddress(10:0)
1024 × 18	Q(17:0)	WrAddress(9:0)
512 × 36	Q(35:0)	WrAddress(9:0)

Table 4.12 shows the various attributes available for the Read Only Memory (ROM). Some of these attributes are user-selectable through the Clarity Designer GUI. For detailed attribute definitions, refer to Appendix A.

Table 4.12. ROM Attributes for ECP5 and ECP5-5G Devices

Configuration Tab Attributes	Description	Values	Default Value
Address Depth	Address depth of the read and write port	2 — 65536	512
Data Width	Data word width of the Read and write port	1 — 256	35
Enable Output Register	Data Out port (Q) can be registered or not using this selection.	TRUE, FALSE	TRUE
Optimization	Design optimizations to configure EBR for Speed or Area	Area, Speed	Speed
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file.	0's, 1's, File	0's
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.	—	—
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats)	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE

Users have the option to enable the output registers for Read Only Memory (ROM). Figure 4.22 and Figure 4.23 show the internal timing waveforms for ROM with these options.

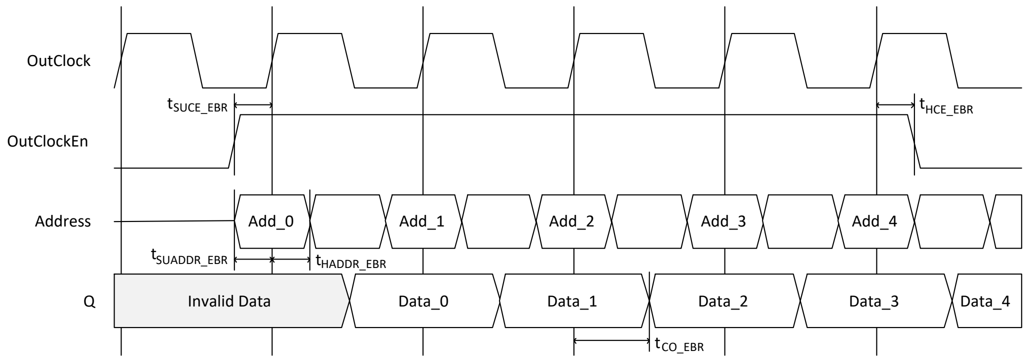


Figure 4.22. IROM Timing Waveform - without Output Registers

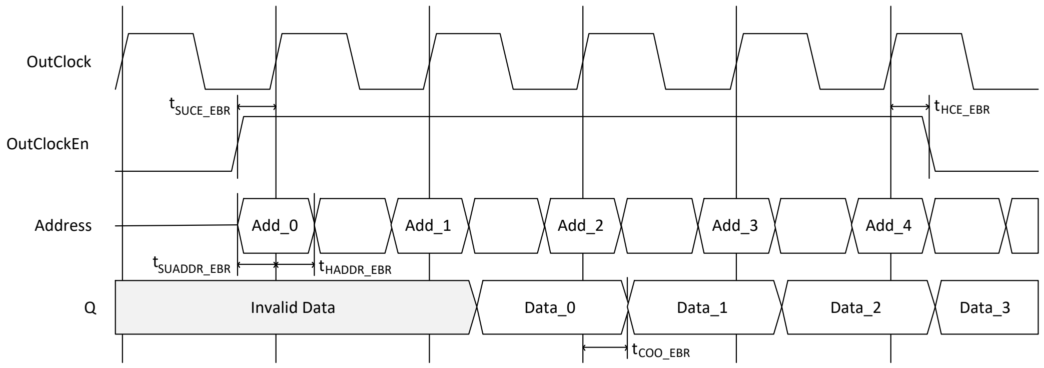


Figure 4.23. ROM Timing Waveform - with Output Registers

5. First In First Out (FIFO) Memory

ECP5 and ECP5-5G devices support two different types of FIFOs:

- Single Clock FIFO (FIFO)
- Dual Clock FIFO (FIFO_DC)

The FIFOs in ECP5 and ECP5-5G devices are emulated such that they are built using Dual Port RAMs with additional logic around them. The additional logic (address counters, flag logic and pointer comparators) are implemented using PFUs around the Dual Port RAMs.

The EBR blocks in ECP5 and ECP5-5G devices can be configured as LUT based or EBR based, as well as Single Clock First-In First-Out Memory (FIFO) or Dual-Clock First-In First-Out Memory (FIFO_DC). Clarity Designer allows users to generate the Verilog-HDL or VHDL netlist for various memory sizes depending on design requirements.

Clarity Designer generated FIFO modules and their operation are discussed in detail in the following pages.

Single Clock FIFO (FIFO) – EBR and LUT

Figure 5.1 shows the module that gets generated by the Clarity Designer for FIFO.

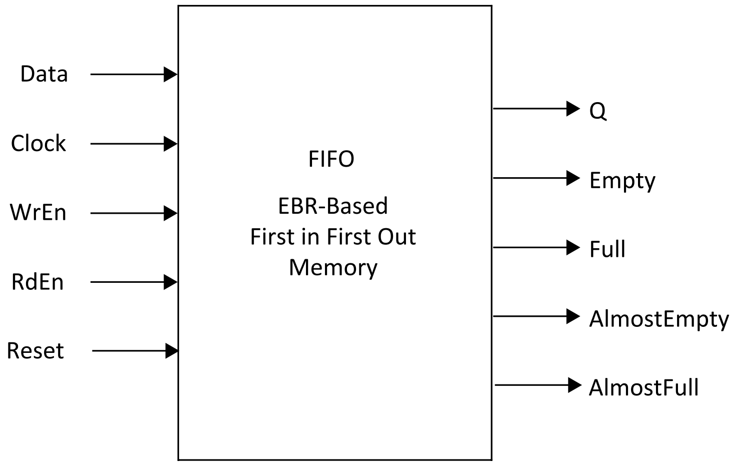


Figure 5.1. FIFO Module Generated by Clarity Designer

The various ports and their definitions for the FIFO are listed in Table 5.1.

Table 5.1. Port Names and Definitions for FIFO

Port Name in the Generated Module	Description	Active State
Data	Input Data	—
Clock	Clock	Rising Clock Edge
WrEn	Write Enable	Active High
RdEn	Read Enable	Active High
Reset[2]	Reset	Active High
Q	Data Output	—
Empty	Empty Flag Active High	
Full	Full Flag Active High	
AlmostEmpty	Almost Empty Flag	Active High
AlmostFull	Almost Full Flag	Active High
ERROR[1]	Error Check Code	Active High

Note:

1. Denotes optional port.
2. Reset resets only the optional output registers, pointer circuitry and flags (except Empty, which gets set to '1') of the FIFO. It does not reset the input registers or the contents of memory.

Let us first discuss the non-pipelined or the FIFO without output registers. Figure 5.2 shows the operation of the FIFO when it is empty and the data begins to be written into it.

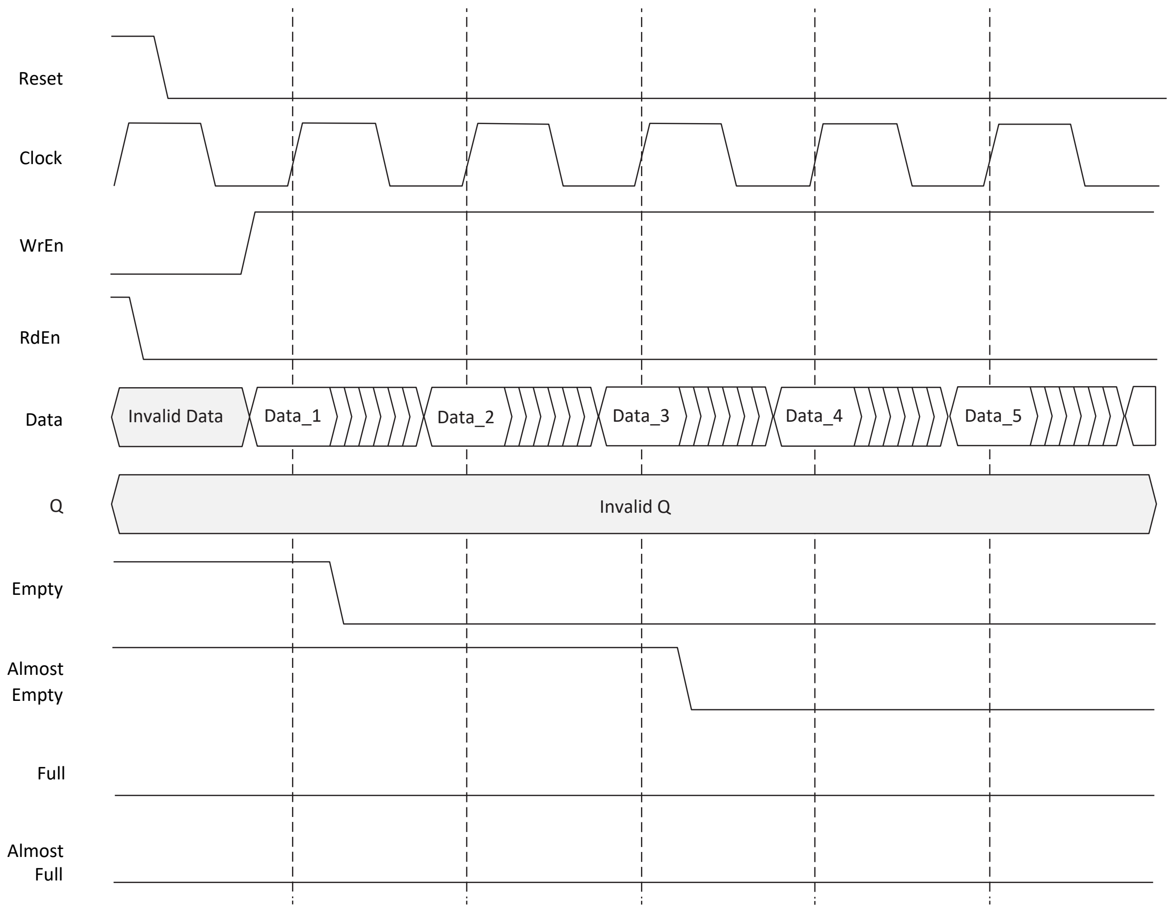


Figure 5.2. FIFO Without Output Registers, Start of data Write Cycle

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

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

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 5.3 shows the behavior of these flags. In this figure we assume that the FIFO depth is 'N'.

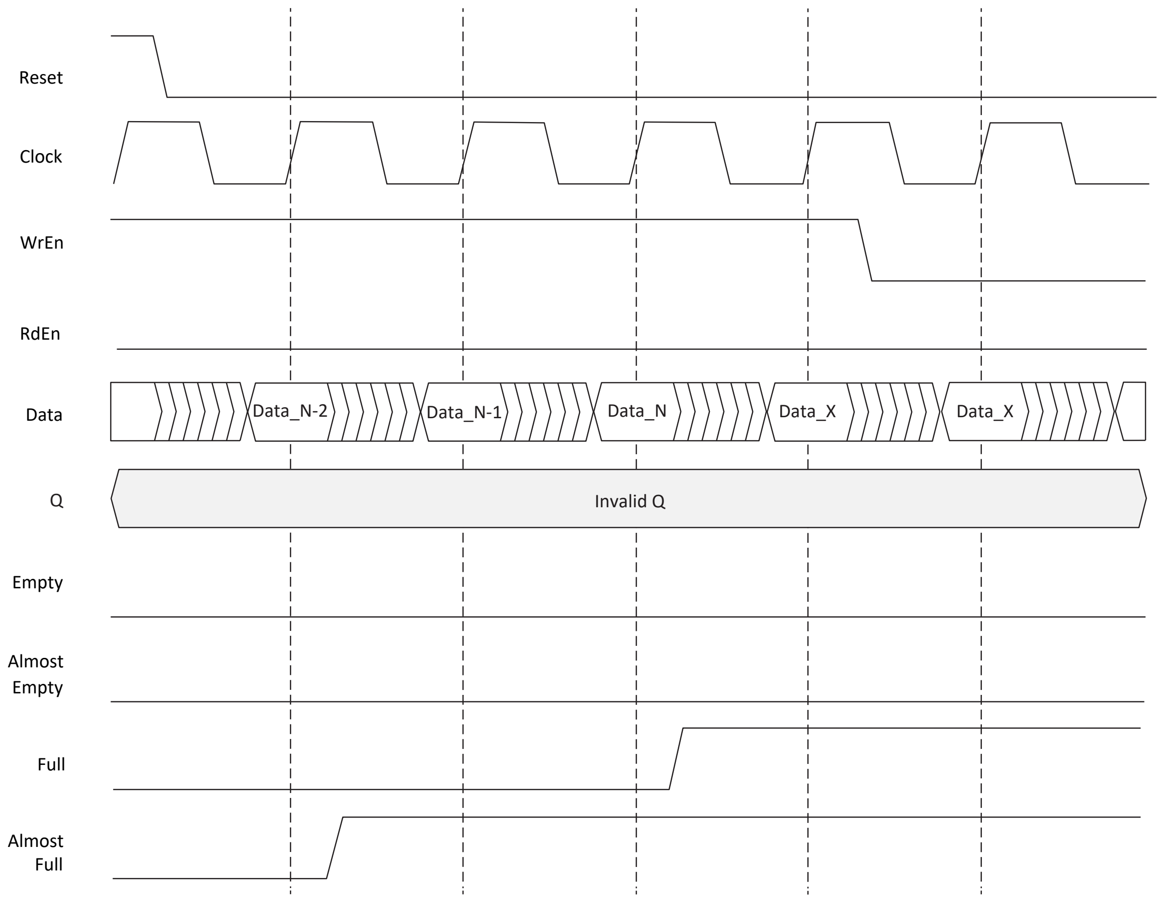


Figure 5.3. FIFO Without Output Registers, End of data Write Cycle

In Figure 5.3, the Almost Full flag is two locations before the FIFO is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO.

Data_X data inputs are not written since the FIFO is full (Full flag is high).

Now let us look at the waveforms when the contents of the FIFO are read out. Figure 5.4 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.

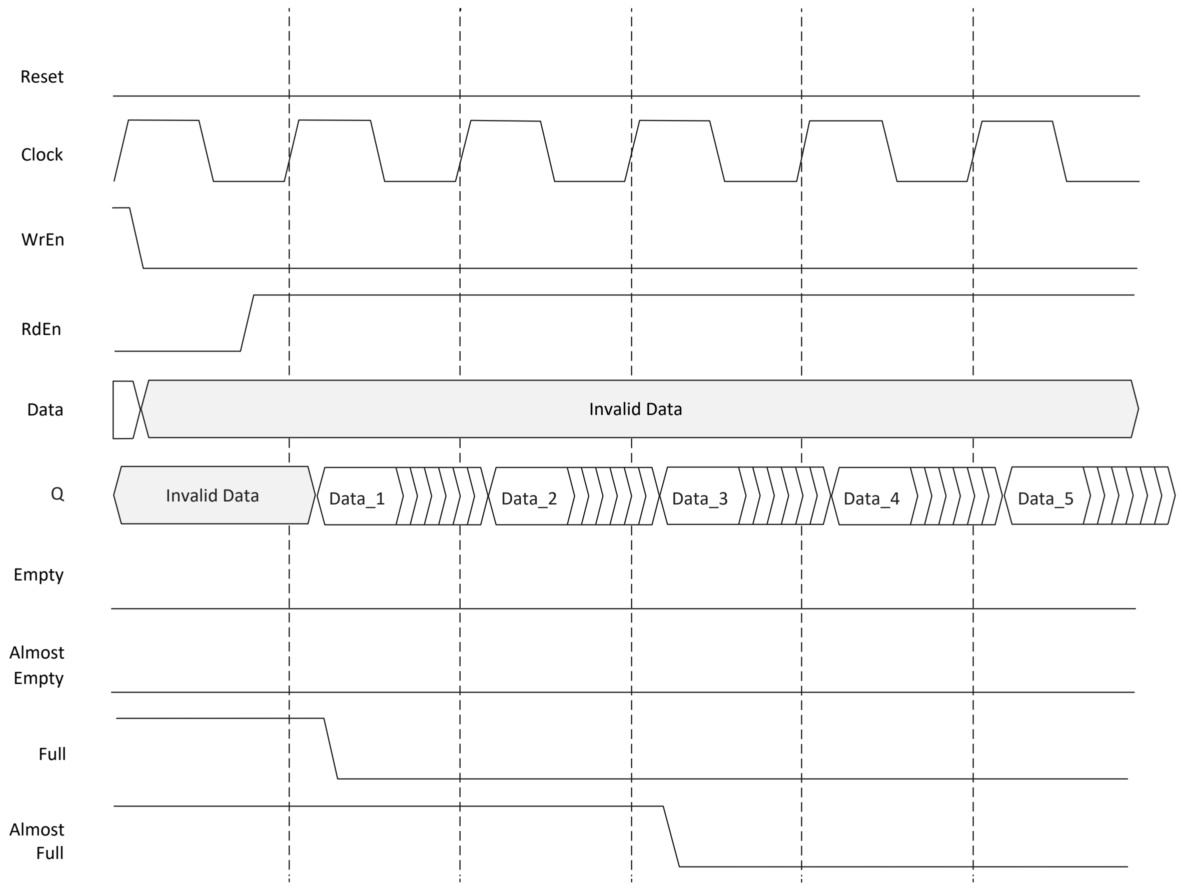


Figure 5.4. FIFO Without Output Registers, Start of Data Read Cycle

Similarly, as the data is read out and FIFO is emptied, the Almost Empty and Empty flags are asserted (see Figure 5.5).

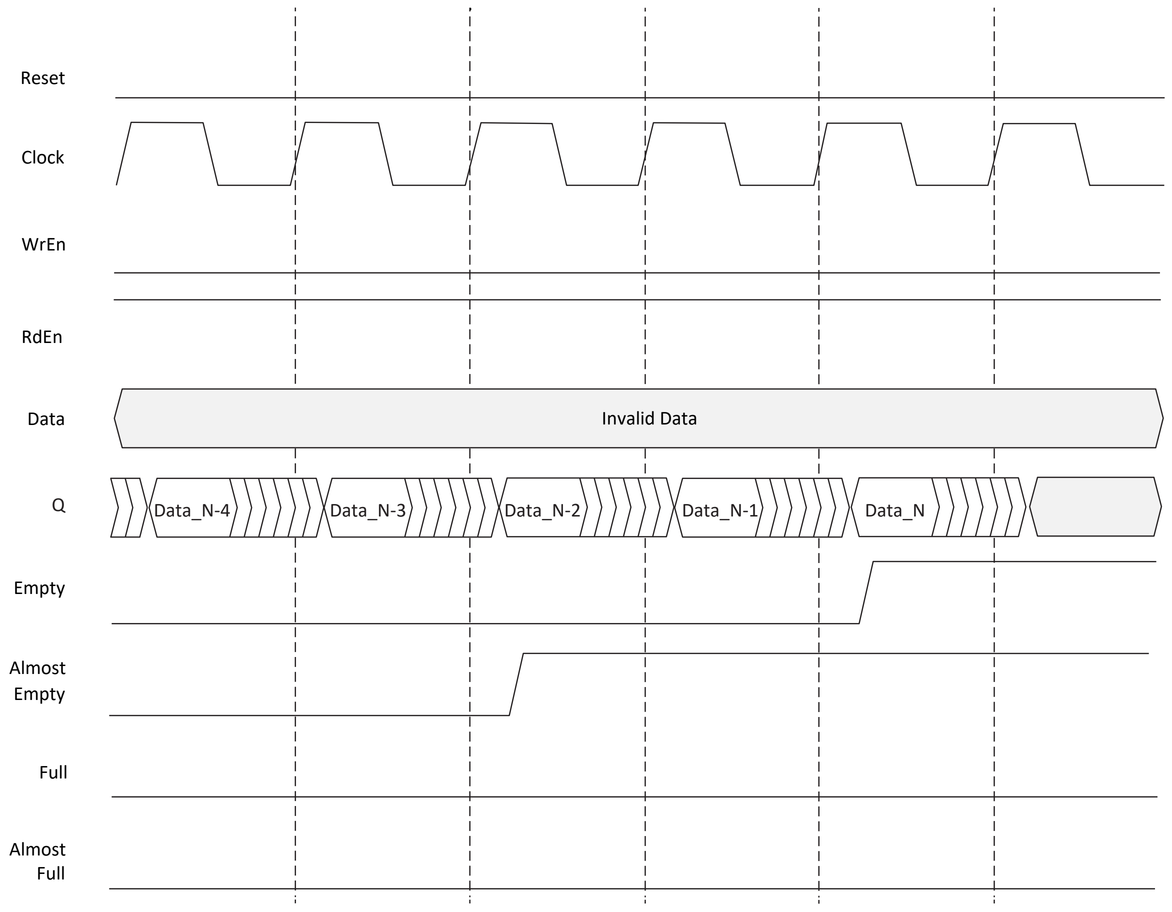


Figure 5.5. FIFO Without Output Registers, End of Data Read Cycle

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

Figure 5.6 to Figure 5.9 show similar waveforms for the FIFO with an output register and an output register enable with RdEn. Note that flags are asserted and de-asserted with similar timing to the FIFO without output registers.

Only the data out 'Q' gets delayed by one clock cycle.

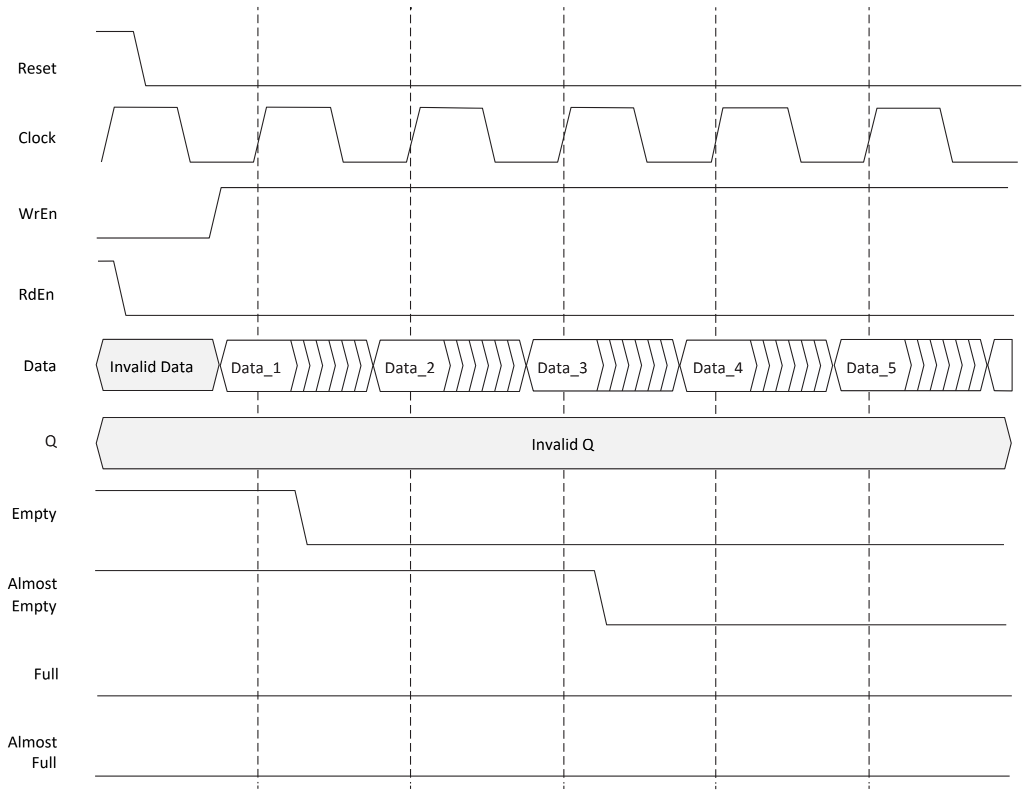


Figure 5.6. FIFO with Output Registers, Start of Data Write Cycle

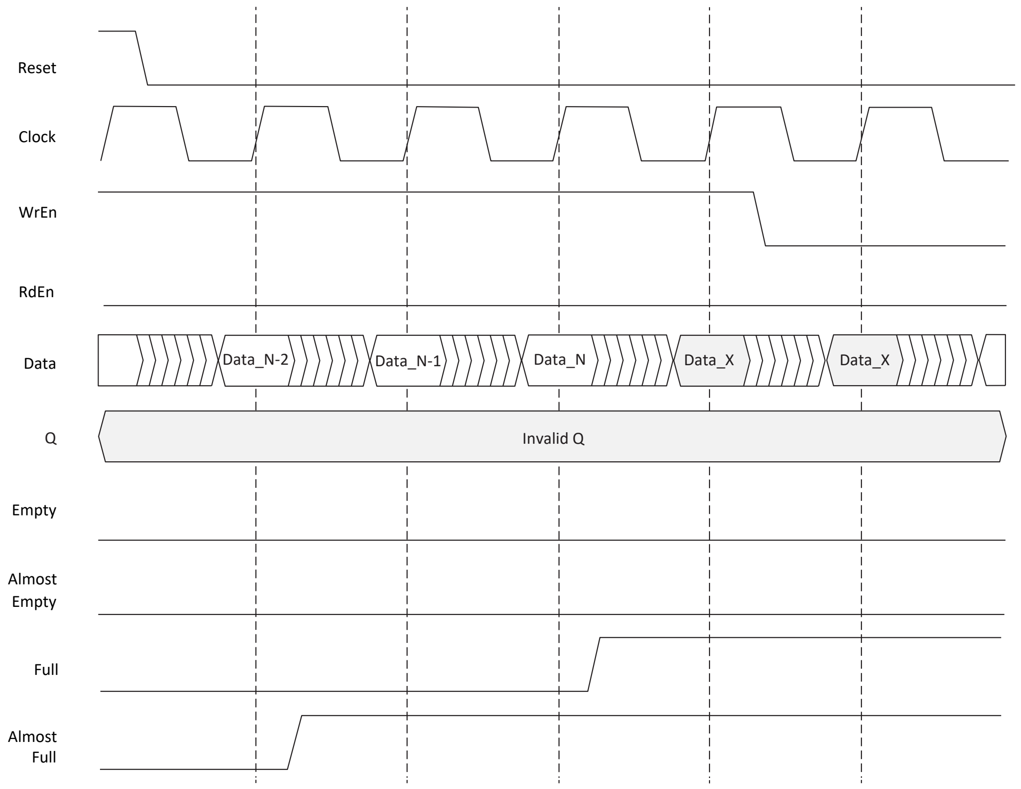


Figure 5.7. FIFO with Output Registers, End of Data Write Cycle

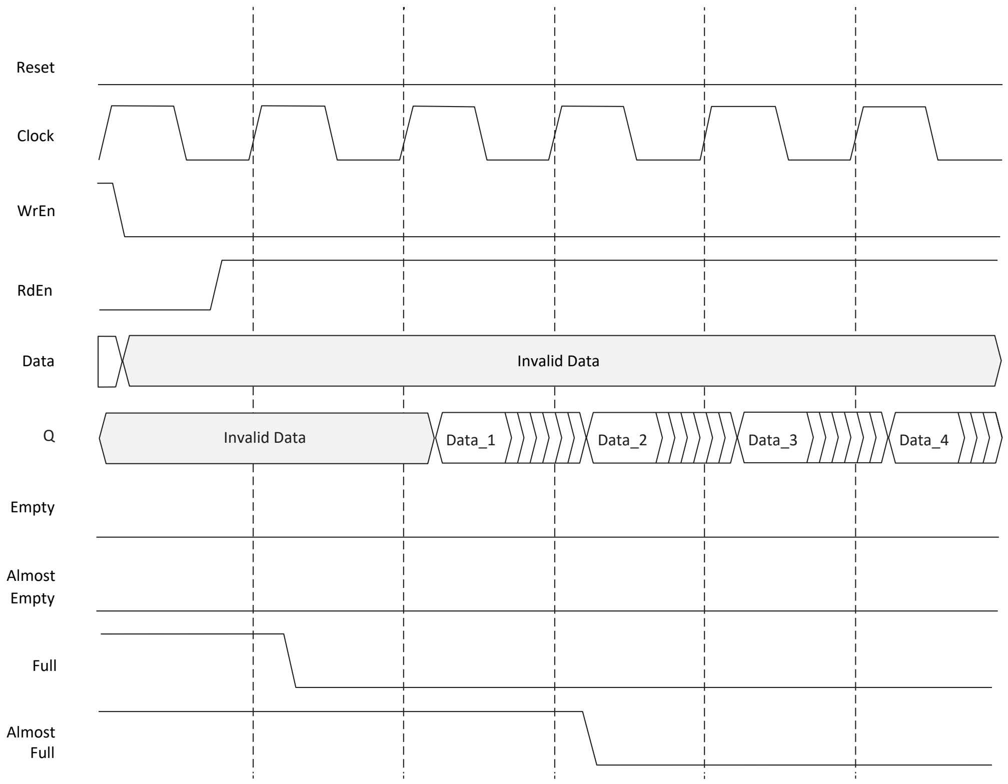


Figure 5.8. FIFO with Output Registers, Start of Data Read Cycle

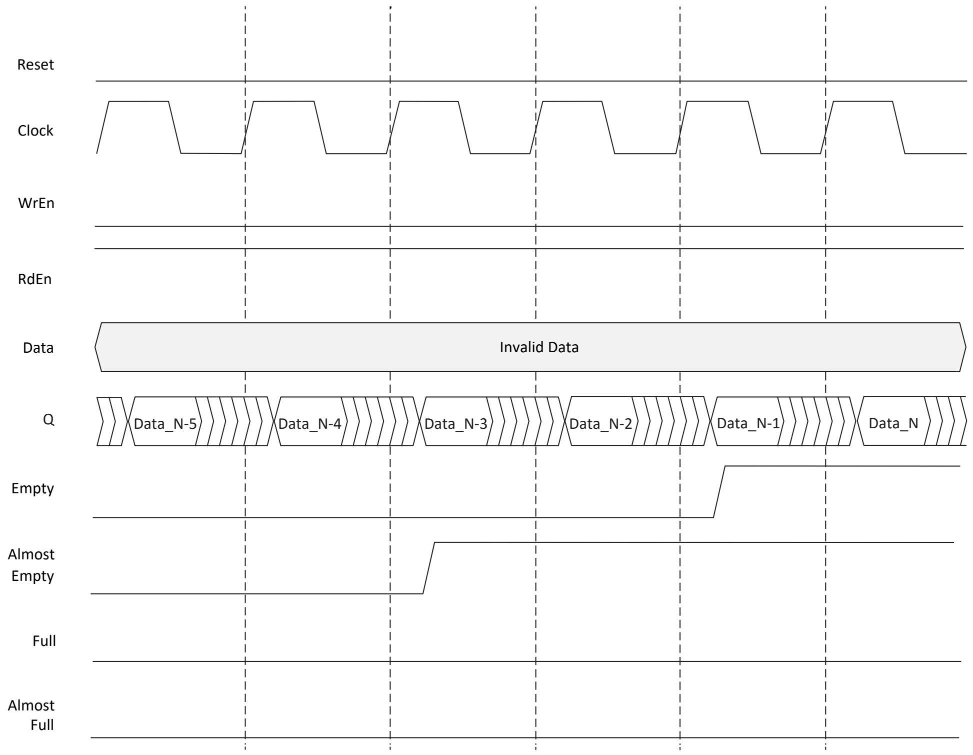


Figure 5.9. FIFO with Output Registers, End of Data Read Cycle

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

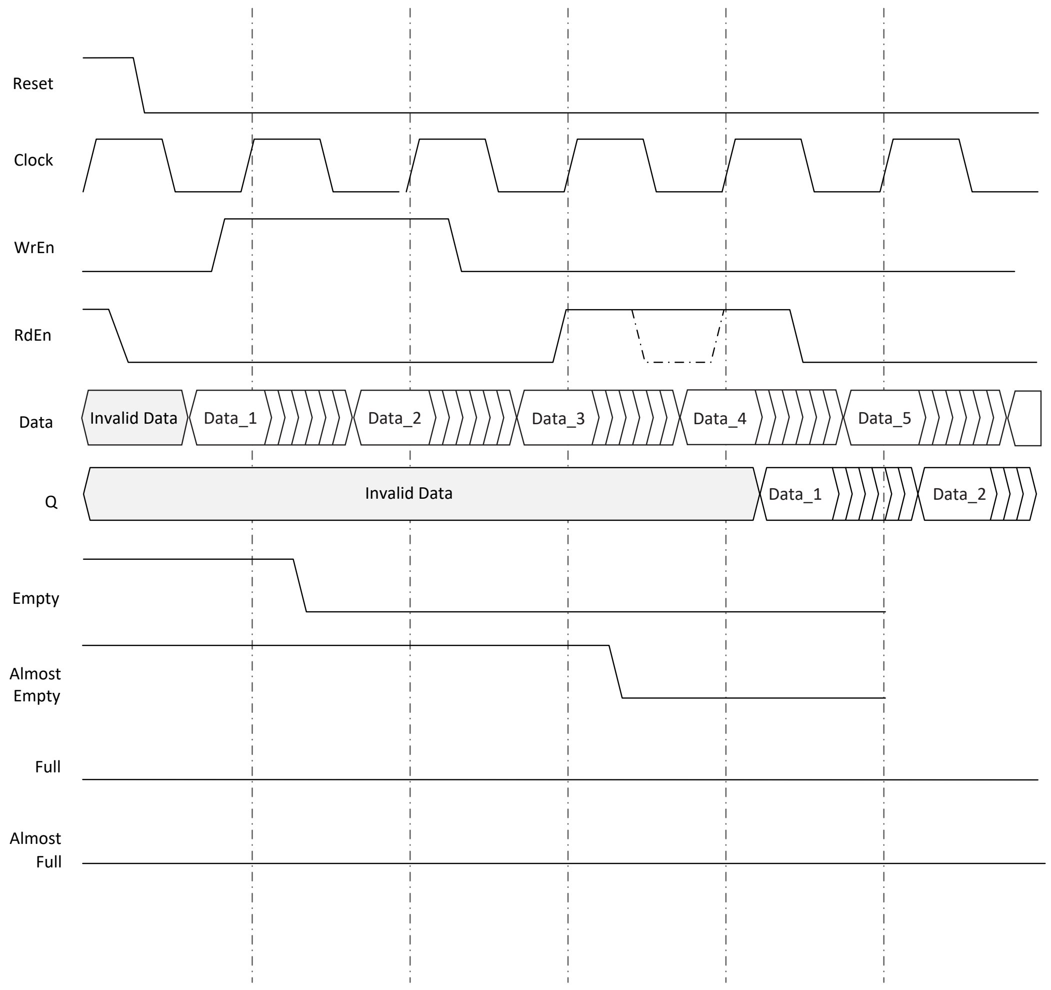


Figure 5.10. FIFO with Output Registers and RdEn on Output Registers

6. Dual Clock First-In-First-Out (FIFO_DC) – EBR or LUT Based

Following figure shows the module that gets generated by the Clarity Designer for FIFO.

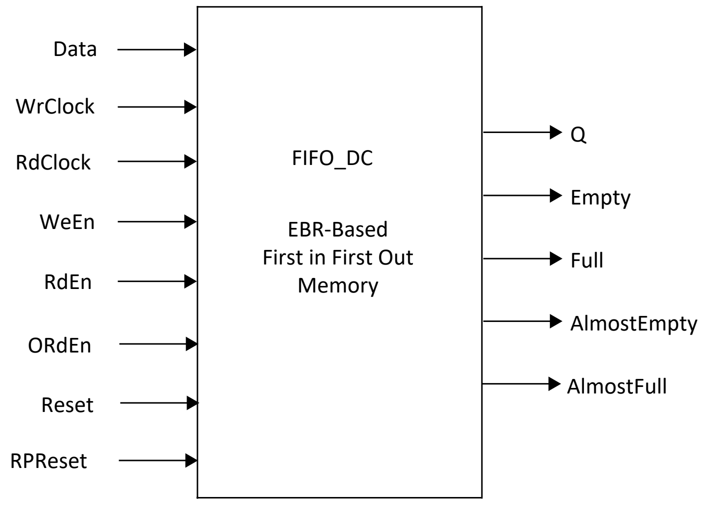


Figure 6.1. FIFO_DC Module generated by the Clarity Designer

The various ports and their definitions for the FIFO_DC are listed in Table 6.1.

Table 6.1. Port Names and Definitions for FIFO

Port Name in the Generated Module	Description	Active State
Data	Input Data	—
WrClock	Write Port Clock	Rising Clock Edge
RdClock	Read Port Clock	Rising Clock Edge
WrEn	Write Enable	Active High
RdEn	Read Enable	Active High
*ORdEn[2]	Output Read Enable	Active High
Reset[3]	Reset	Active High
RPReset[4]	Read Pointer Reset	Active High
Q	Data Output	—
Empty	Empty Flag Active High	
Full	Full Flag Active High	
AlmostEmpty	Almost Empty Flag	Active High
AlmostFull	Almost Full Flag	Active High
ERROR[1]	Error Check Code	Active High

Note:
1. Denotes optional port.
2. ORdEn 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. Reset 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. RPReset resets only the read pointer. See additional discussion below.

6.1. FIFO_DC Flags

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

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

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 the de-assertion of the Empty and Almost Empty flags.

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

With this in mind, let us look at the waveforms for FIFO_DC without output registers. Figure 6.2 shows the operation of the FIFO_DC when it is empty and the data begins to be written into it.

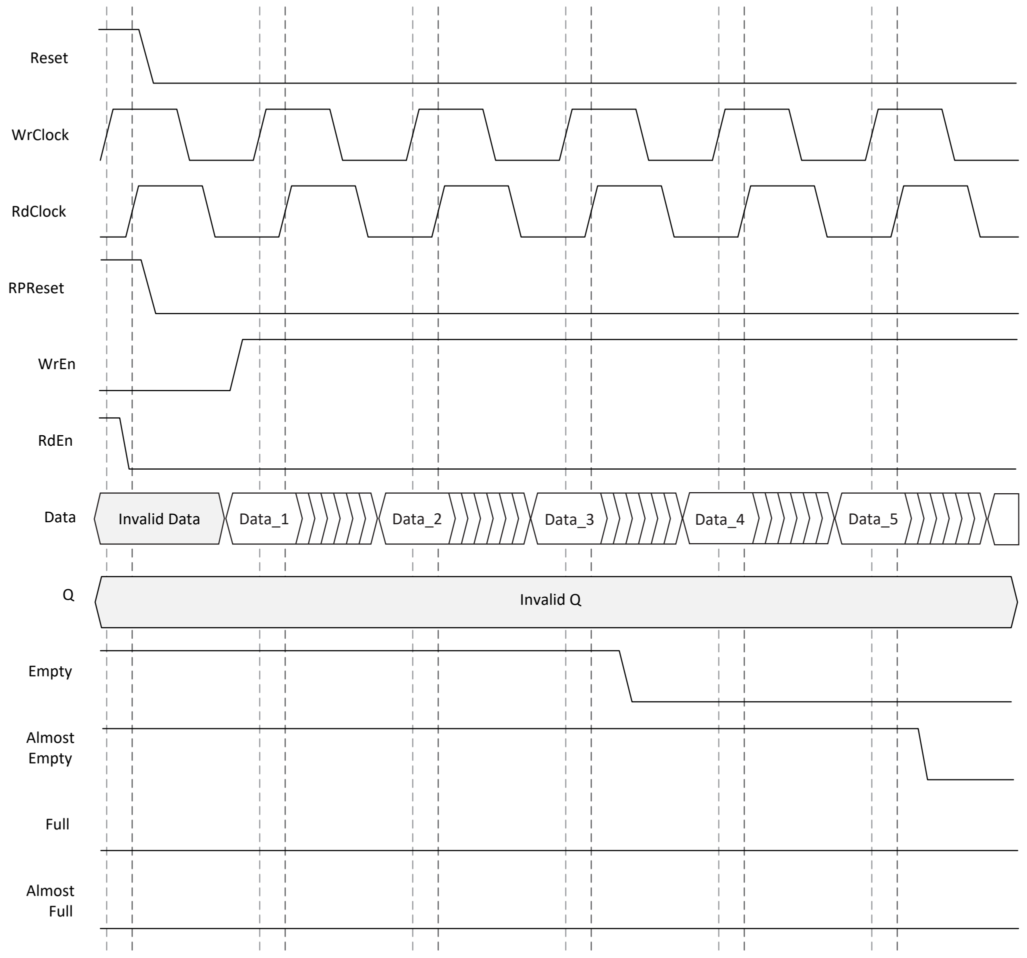


Figure 6.2. FIFO_DC Without Output Registers, Start of Data Write Cycle

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

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

Now 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 6.3 shows the behavior of these flags. In this figure we assume that FIFO_DC depth is 'N'.

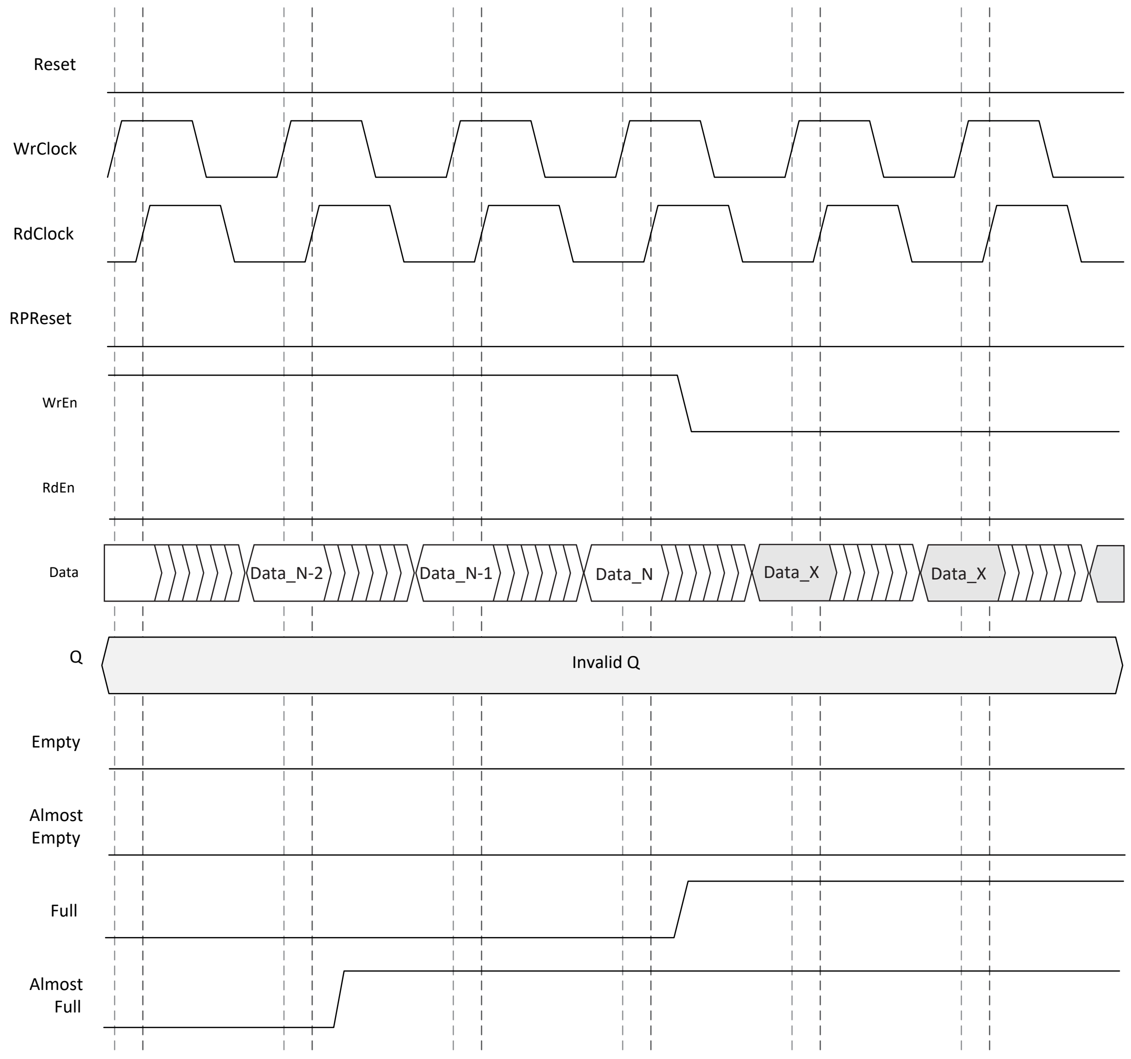


Figure 6.3. FIFO_DC Without Output Registers, End of Data Write Cycle

In Figure 6.3, the Almost Full flag is two locations before the FIFO_DC is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO_DC. Data_X data inputs are not written since the FIFO_DC is full (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 6.4 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted as shown. Note that the de-assertion is delayed by two clock cycles.

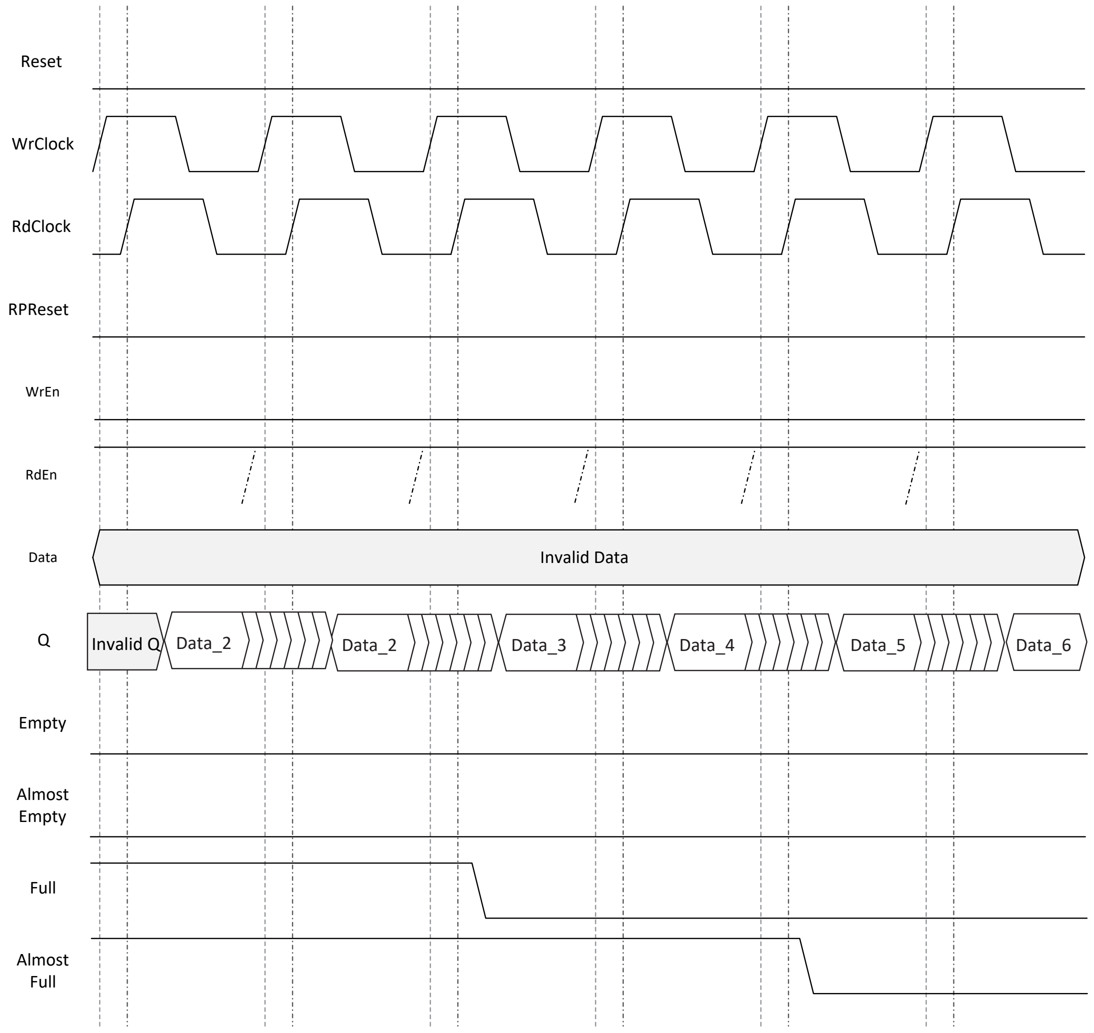


Figure 6.4. FIFO_DC Without Output Registers, Start of Data Read Cycle

Similarly, as the data is read out and FIFO_DC is emptied, the Almost Empty and Empty flags are asserted (see Figure 6.5).

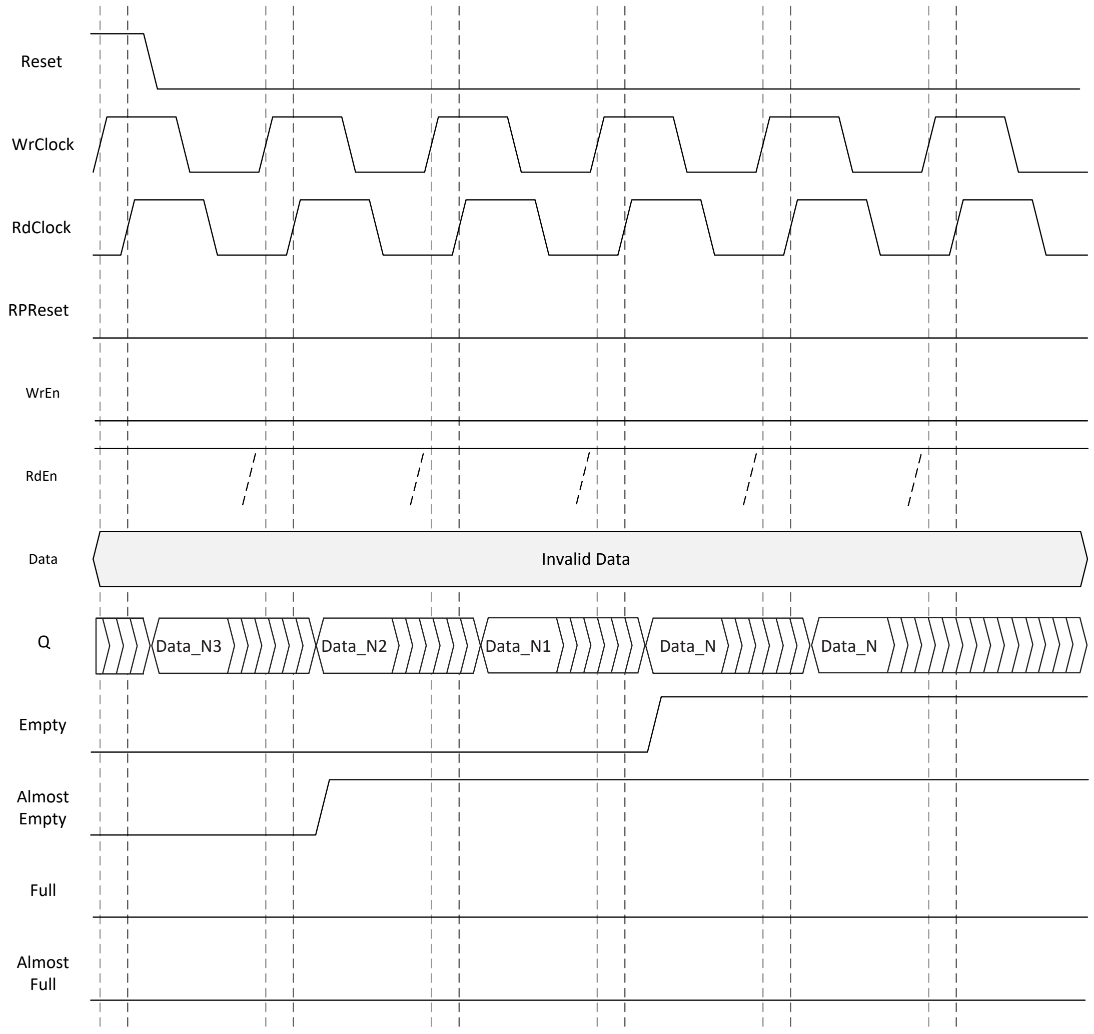


Figure 6.5. FIFO_DC Without Output Registers, Start of Data Read Cycle

Figure 6.2 to Figure 6.5 show the behavior of the non-pipelined FIFO_DC or FIFO_DC without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is an extra option for output registers to be enabled by the RdEn signal.

Figure 6.6 to Figure 6.8 show similar waveforms for the FIFO_DC with output register and without output register enable with RdEn. Note that flags are asserted and de-asserted with timing similar to the FIFO_DC without output registers. However, only the data out 'Q' is delayed by one clock cycle.

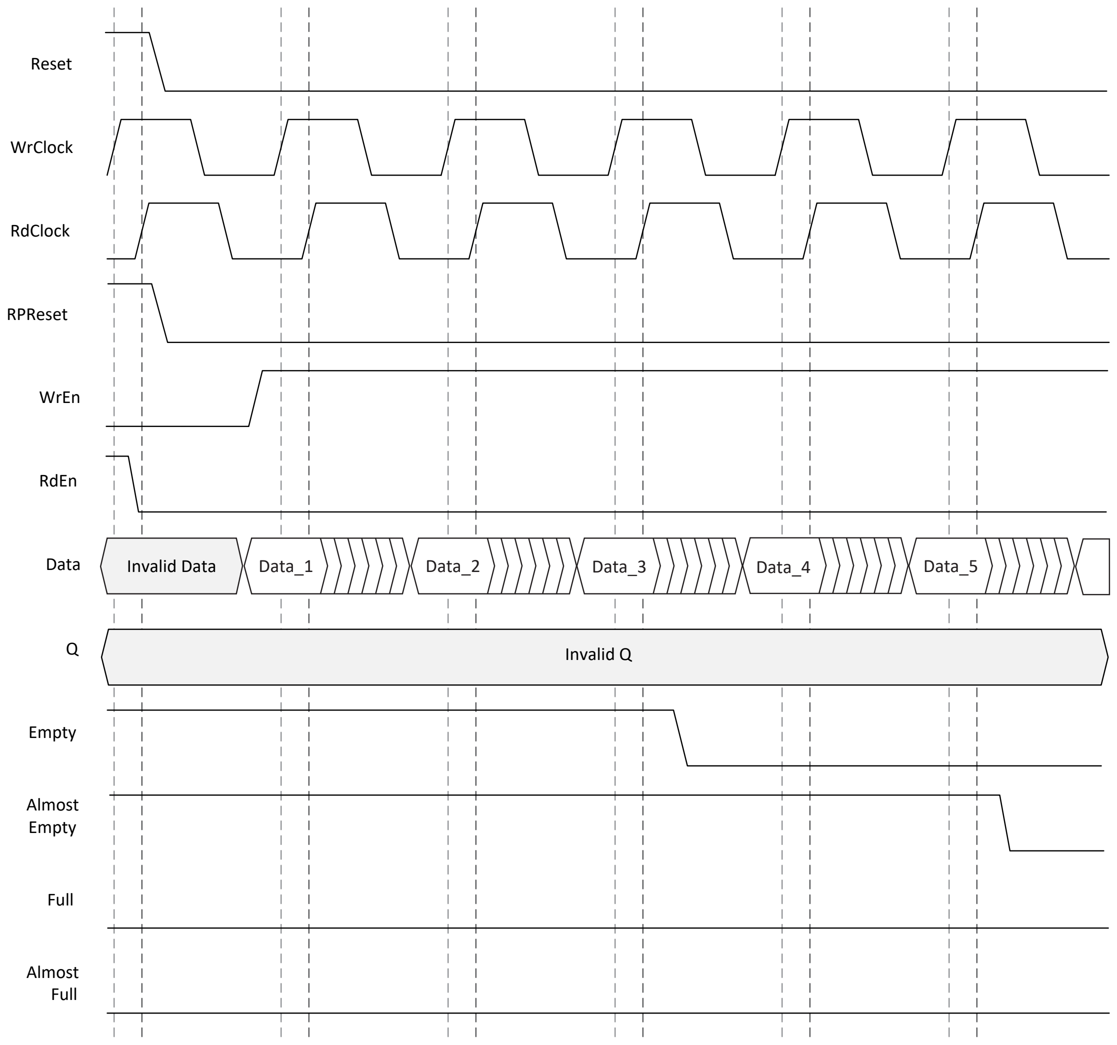


Figure 6.6. FIFO_DC with Output Registers, Start of Data Write Cycle

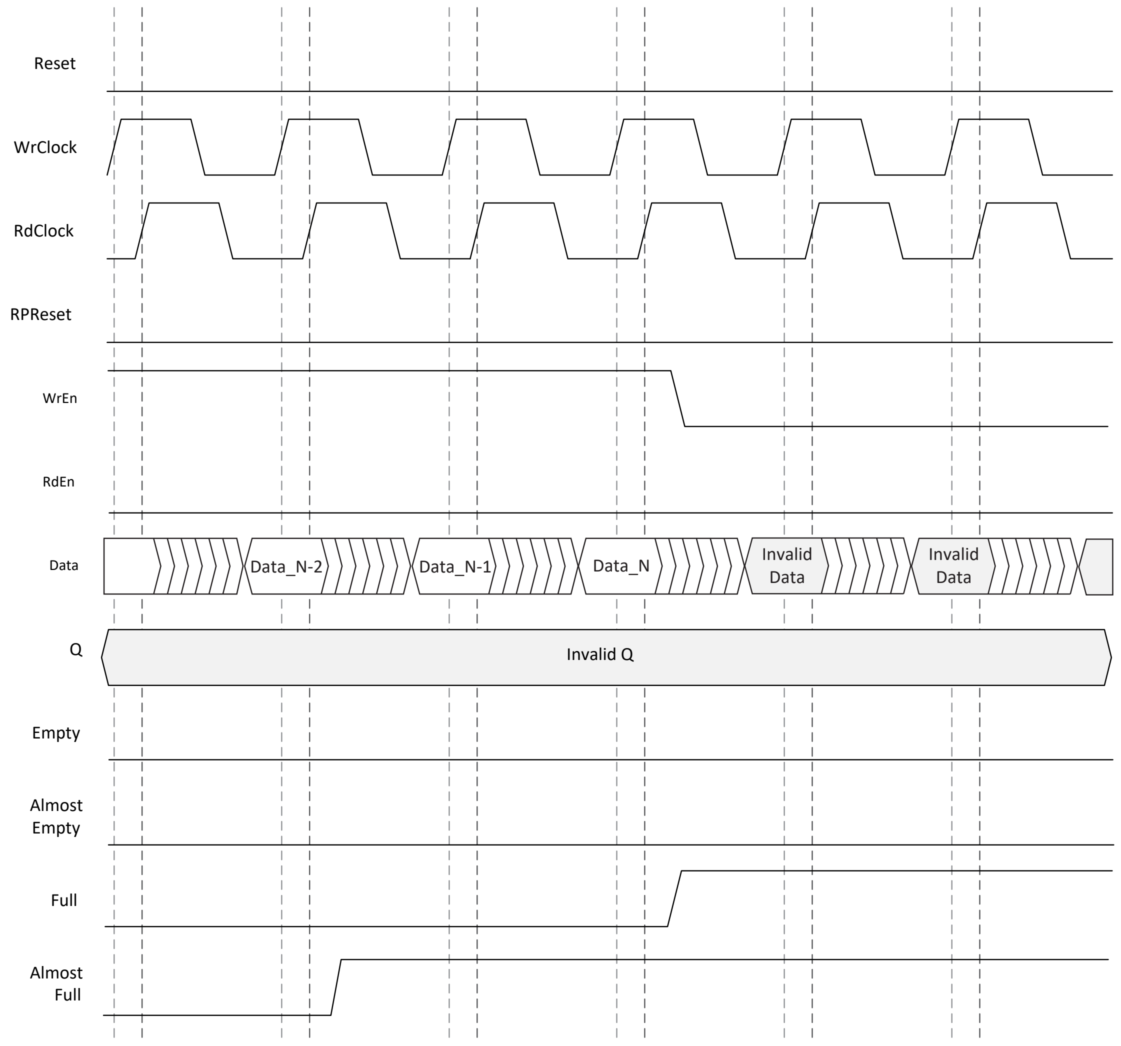


Figure 6.7. FIFO_DC with Output Registers, End of Data Write Cycle

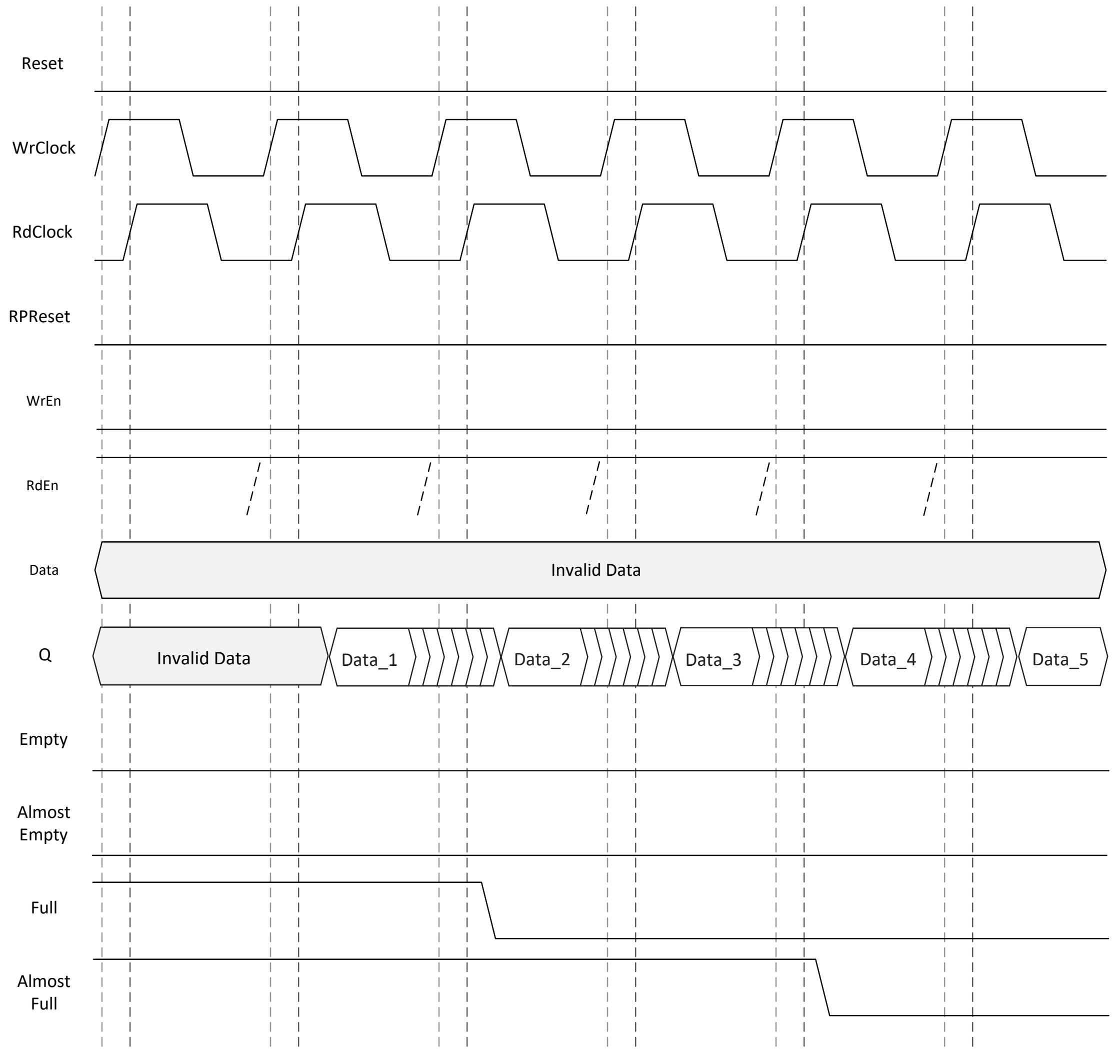


Figure 6.8. FIFO_DC with Output Registers, Start of Data Read Cycle

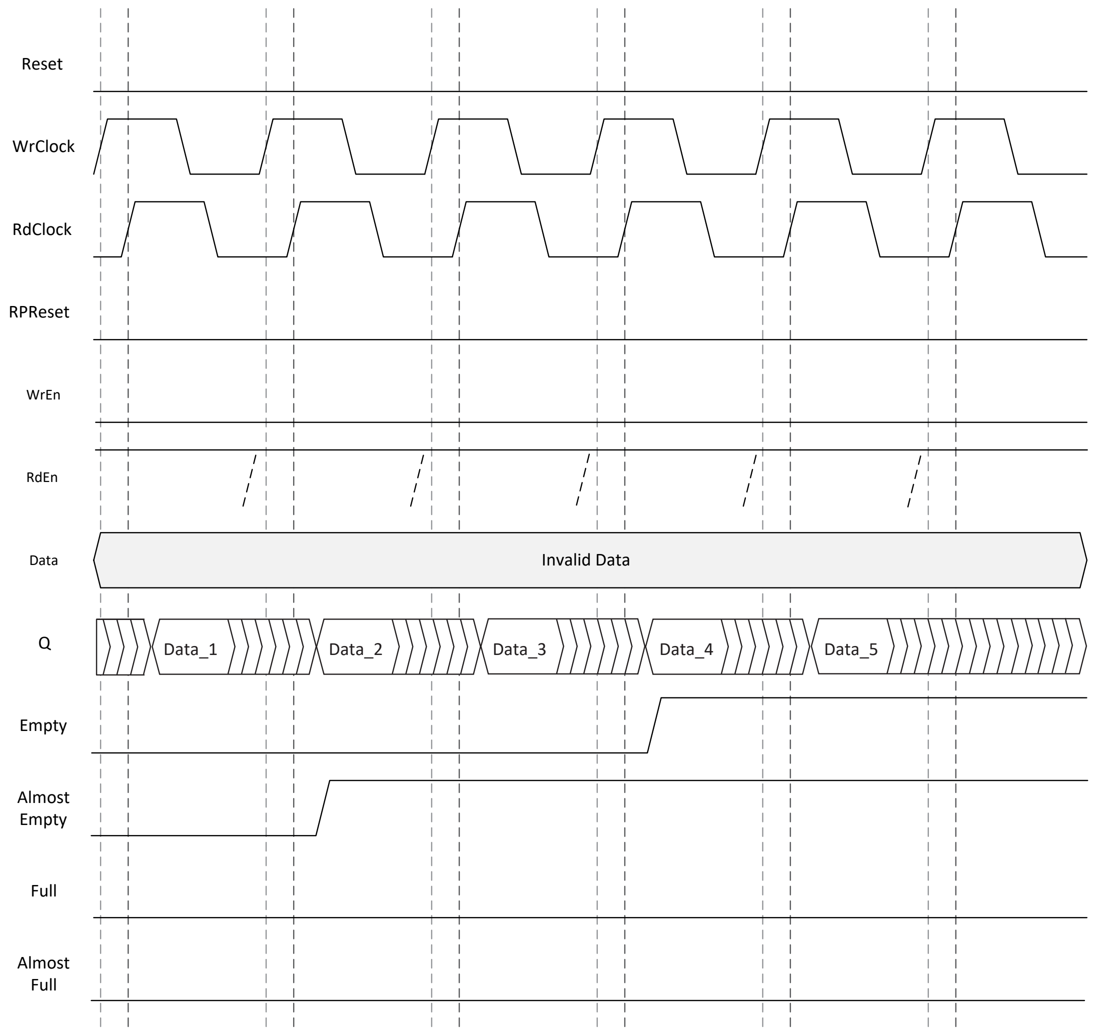


Figure 6.9. FIFO_DC with Output Registers, End of Data Read Cycle

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

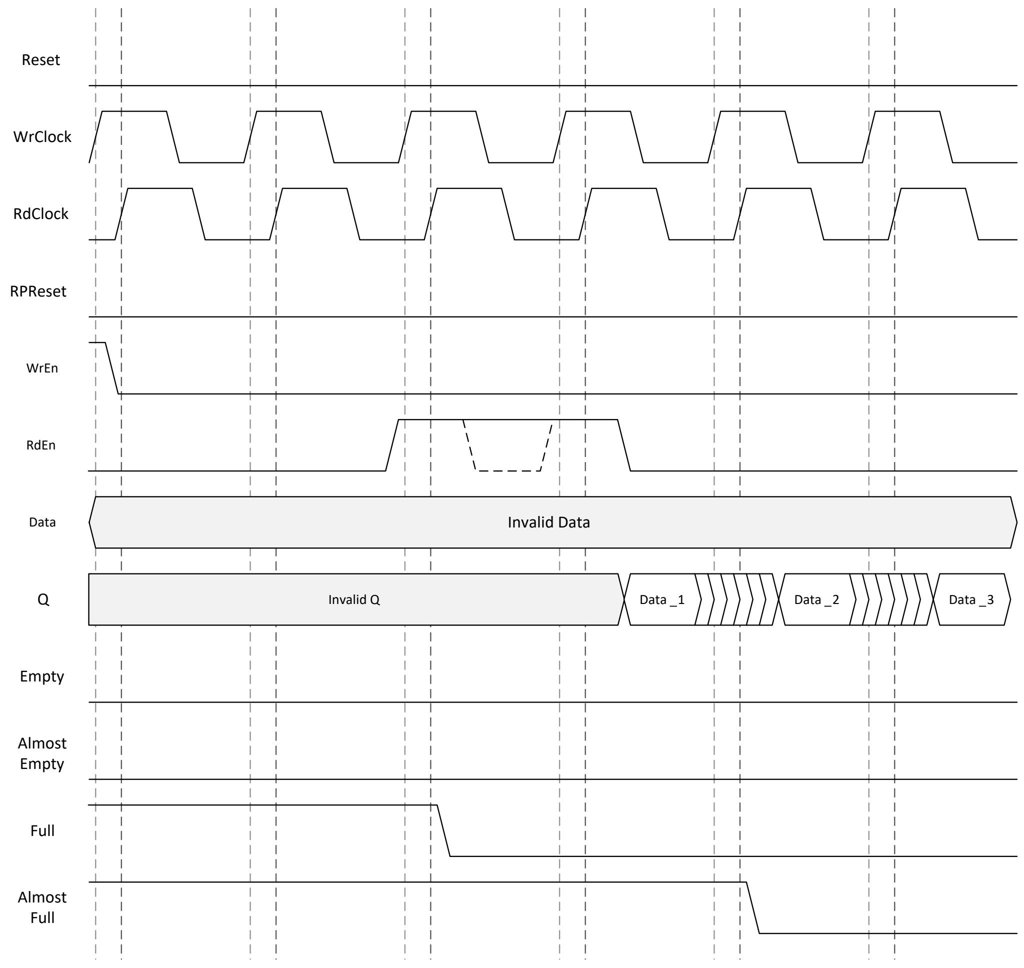


Figure 6.10. FIFO_DC with Output Registers and RdEn on Output Registers

When using FIFO_DC with different data widths on read and write ports, care should be taken to make sure that wider data width should be the multiple of smaller one. In addition to that, the words written or read out should follow the same relationship. For example, let us assume that the DataIn (write port) width is 8-bits and DataOut (read port) is 16-bits. In this case, there is a factor of 2 between the two. Now, for every 2 words written in the FIFO_DC, one word is read out. If we write an odd number of words, such as 7 for example, then the read port will read 3 complete words, and one half word. The other half of the incomplete word will either be the all zeroes (0s) or prior data written at the 8th location.

If we reverse the number of bits on DataIn and DataOut, then for every written word, 2 words are read out. Once again, in order to completely read the FIFO_DC, we need twice the number of clock cycles on the write port.

FIFO_DC does not include any arbitration logic, it has to be implemented outside of the FIFO_DC. Read and Write Count pointers can be used to aid in counting the number of written or read words.

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

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

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

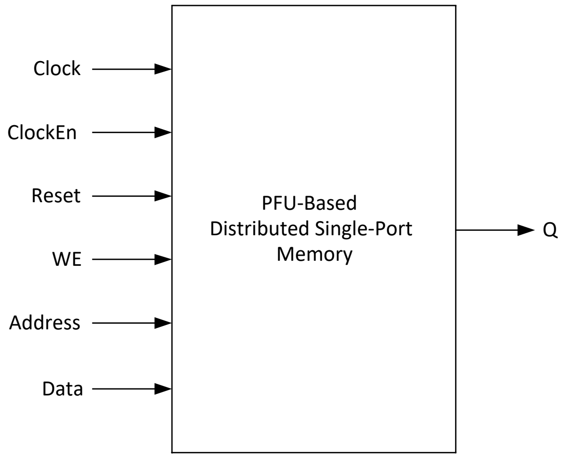


Figure 7.1. Distributed Single-Port RAM Module Generated by Clarity Designer

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

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

Figure 7.2 provides the primitive that can be instantiated for the Single Port Distributed RAM. Primitive name is SPR16X4C and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 64 bits of memory; if the memory required is larger than 64 bits, then cascading can be used. Further, the ports can be registered by using external PFU registers.

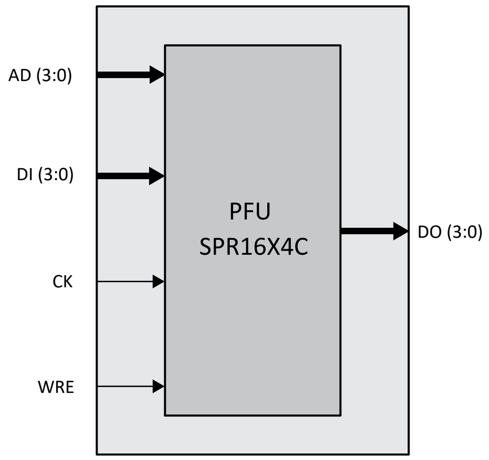


Figure 7.2. Single Port Distributed RAM Primitive for ECP5 and ECP5-5G Devices

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

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

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

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

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

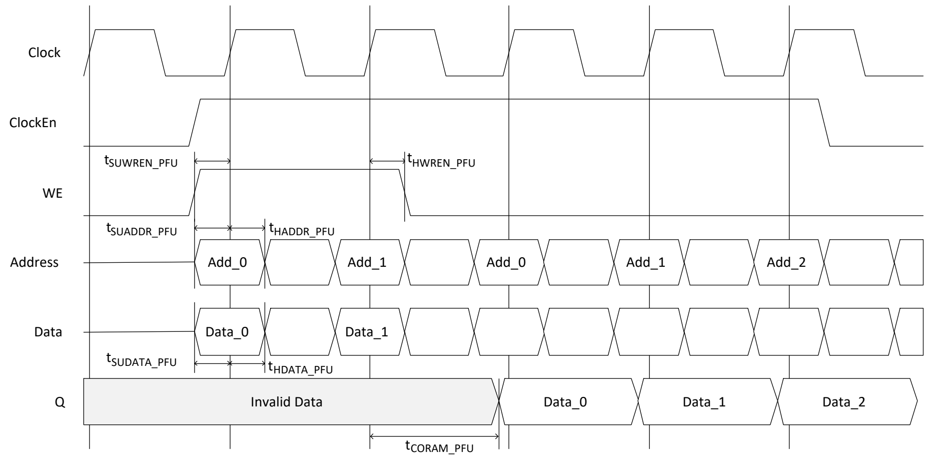


Figure 7.3. PFU Based Distributed Single Port RAM Timing Waveform – without Output Registers

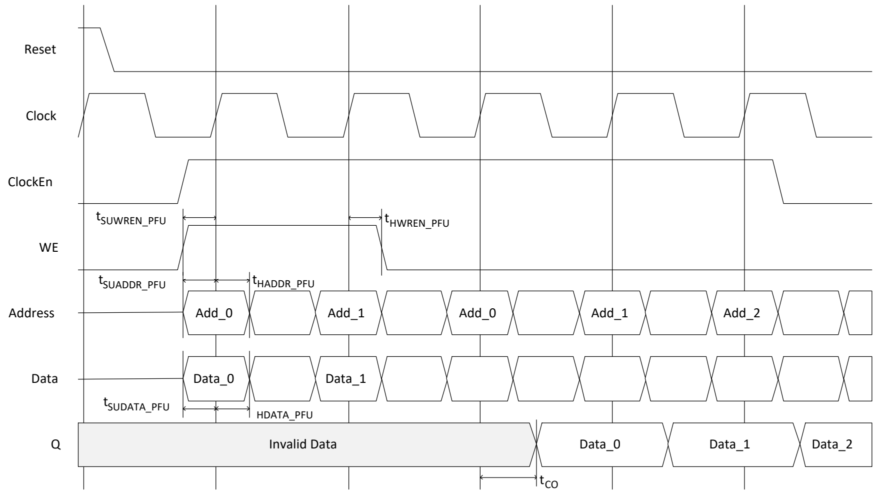


Figure 7.4. PFU Based Distributed Single Port RAM Timing Waveform – with Output Registers

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

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

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

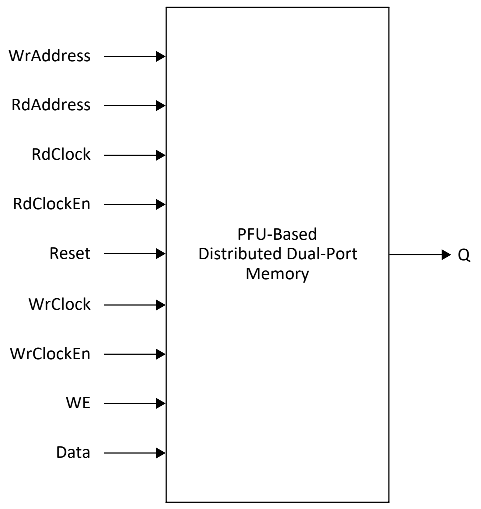


Figure 8.1. Distributed Dual-Port RAM Module Generated by Clarity Designer

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

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

Figure 8.2 provides the primitive that can be instantiated for the Dual Port Distributed RAM. Primitive name is DPR16X4C and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 64 bits of memory; if the memory required is larger than 64 bits, then cascading can be used. Further, the ports can be registered by using external PFU registers.

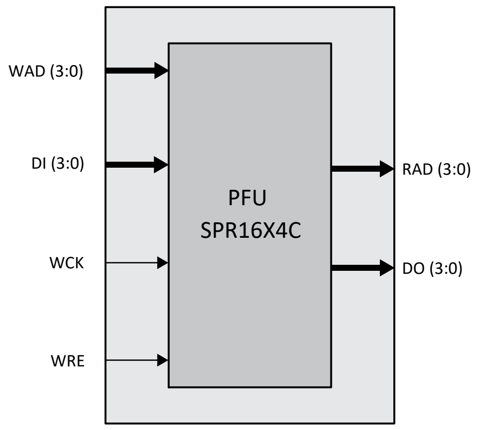


Figure 8.2. Dual Port Distributed RAM Primitive for ECP5 and ECP5-5G Devices

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

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

Port Name in the Generated Module	Port Name in the EBR block Primitive	Description
WrAddress	WAD[23:0]	Write Address
RdAddress	RAD[3:0]	Read Address
RdClock	—	Read Clock
RdClockEn	—	Read Clock Enable
WrClock	WCK	Write Clock
WrClockEn	—	Write Clock Enable
WE	WRE	Write Enable
Data	DI[1:0]	Data Input
Q	RDO[1:0]	Data Out

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

Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed_DPRAM). Figure 8.3 and Figure 8.4 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed_DPRAM) with these options.

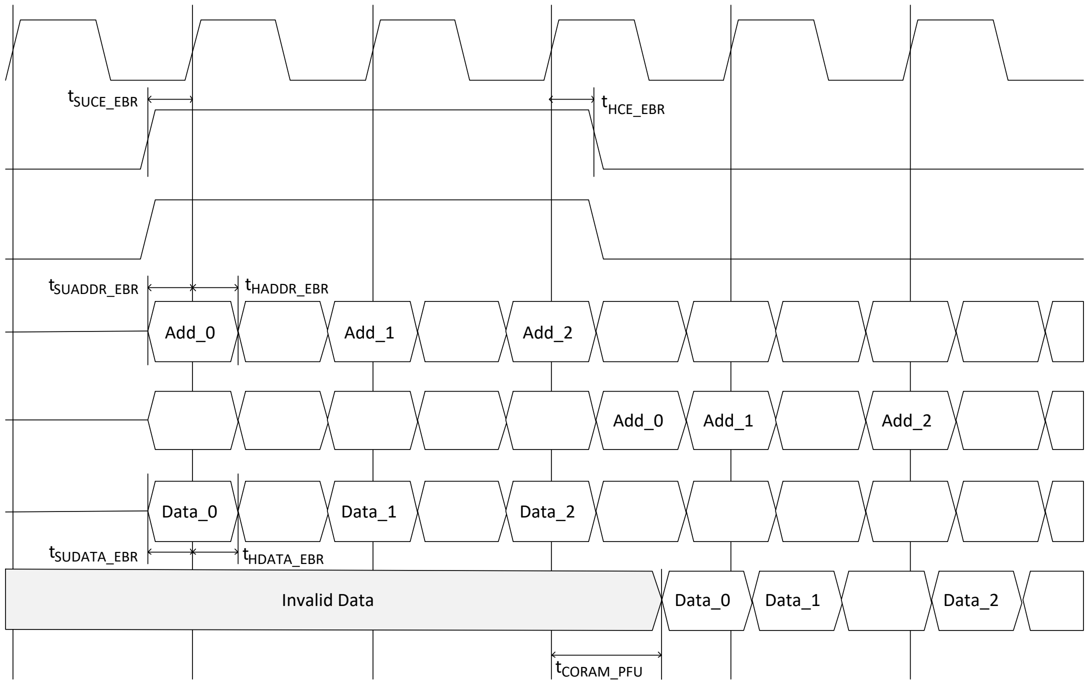


Figure 8.3. PFU Based Distributed Dual Port RAM Timing Waveform – without Output Registers

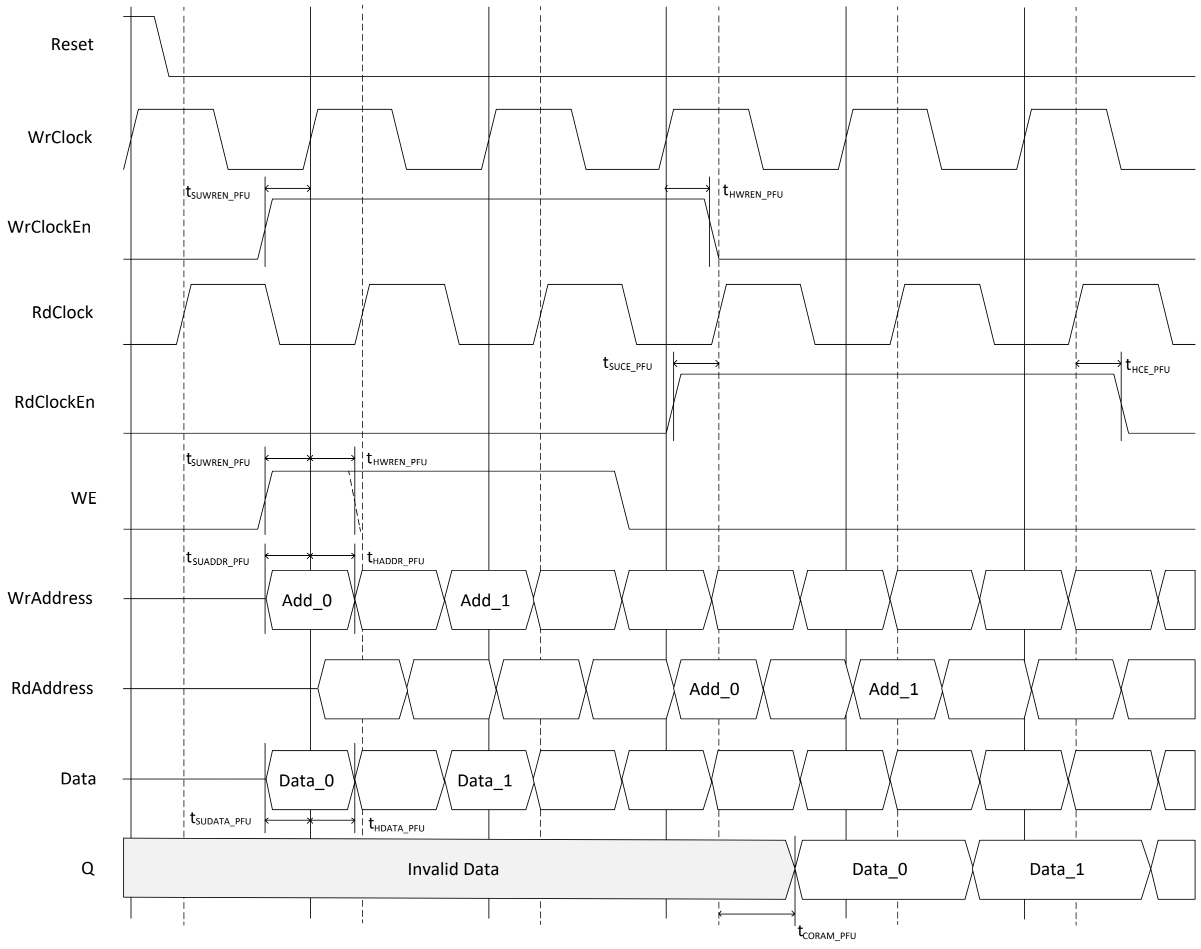


Figure 8.4. PFU Based Distributed Dual Port RAM Timing Waveform – with Output Registers

9. Distributed ROM (Distributed_ROM) – PFU-Based

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

Figure 9.1 shows the Distributed ROM module generated by Clarity Designer.

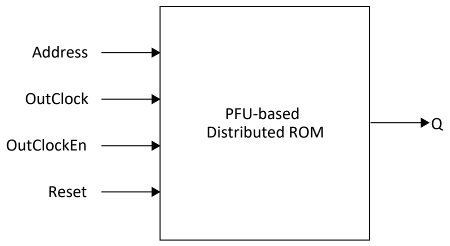


Figure 9.1. Distributed ROM Generated by Clarity Designer

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

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

Figure 9.2 provides the primitive that can be instantiated for the Distributed ROM. Primitive name is DPRnX1a and it can be directly instantiated in the code. 'n' can be 3, 4, 5, 6 or 7 depending upon the size of the ROM. Check the details on the port and port names under the primitives available under cae_library/synthesis folder in Lattice Diamond software installation folder.

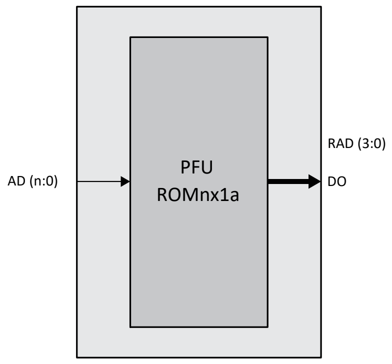


Figure 9.2. Distributed ROM Primitive for ECP5 and ECP5-5G Devices

If the memory required is larger than what can fit in the primitive bits, then cascading can be used. Further, the ports can be registered by using external PFU registers.

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

Table 9.1. PFU-Based Distributed ROM Port Definitions

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

Users have the option to enable the output registers for Distributed ROM (Distributed_ROM). Figure 9.3 and Figure 9.4 show the internal timing waveforms for the Distributed ROM with these options.

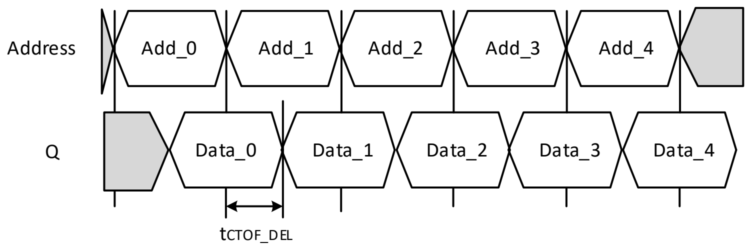


Figure 9.3. PFU Based Distributed ROM Timing Waveform - without Output Registers

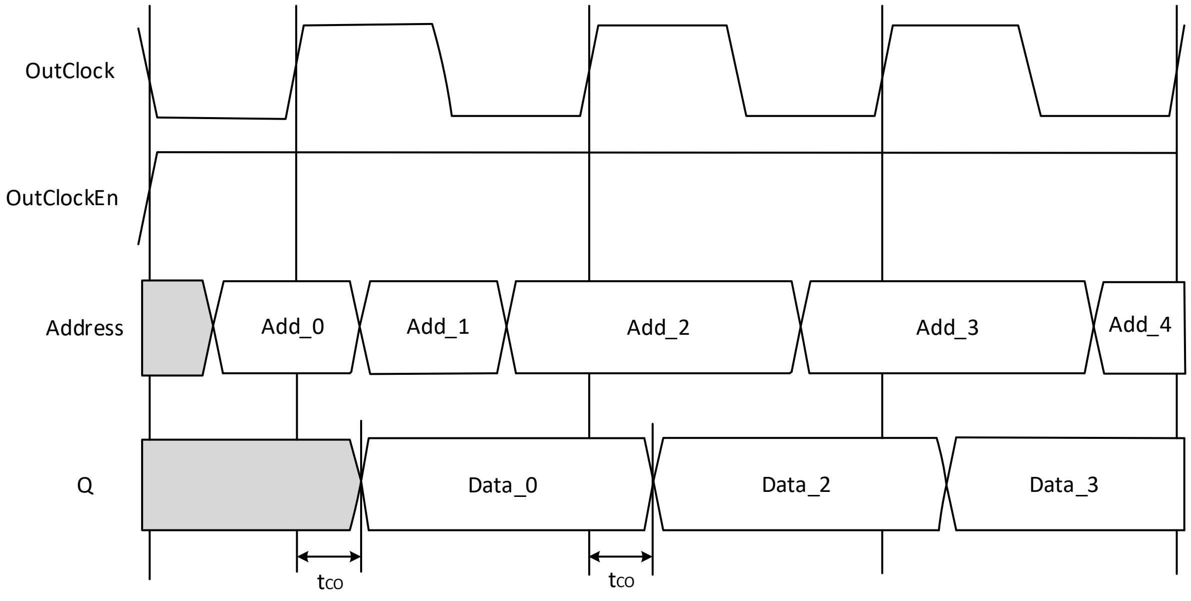


Figure 9.4. PFU PFU Based Distributed ROM Timing Waveform - with Output Registers

10. Initializing Memory

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

10.1. Initialization File Formats

The initialization file is an ASCII file, which the designer can create or edit using any ASCII editor. Clarity Designer supports three memory file formats:

1. Binary file
2. Hex File
3. Addressed Hex

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

The memory initialization can be static or dynamic. In case of static initialization, the memory values are stored in the bitstream. Dynamic initialization of memories, involve memory values stored in the external flash and can be updated by user logic knowing the EBR address locations. The size of the bitstream (bit or rbt file) will be larger due to static values stored in them.

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

10.1.1. Binary File

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

Memory Size 20×32

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

10.1.2. Hex File

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

Memory Size 8×16

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

10.1.3. Addressed Hex

Addressed hex consists of lines of addresses and data. Each line starts with an address, followed by a colon, and any number of data. The format of the file is “address: data data data data” where the address and data are hexadecimal numbers.

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

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

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

Appendix A. Attribute Definitions

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 uses 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. CSDECODE

CSDECODE or the Chip Select Decode attributes are associated with block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily.

CSDECODE takes the following parameters: “000”, “001”, “010”, “011”, “100”, “101”, “110”, and “111”. CSDECODE values determine the decoding value of CS[2:0]. CSDECODE_W is chip select decode for write and CSDECODE_R is chip select decode for read for Pseudo Dual Port RAM. CSDECODE_A and CSDECODE_B are used for True Dual-Port RAM elements and refer to the A and B ports.

A.4. WRITEMODE

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

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

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

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

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write address, during a write cycle. This mode is supported for x9, x18 and x36 data widths.

For all modes of the True Dual-Port module, simultaneous read access from one port and write access from the other port to the same memory address is not recommended. The read data may be unknown in this situation.

Also, simultaneous write access to the same address from both ports is not recommended. When this occurs, the data stored in the address becomes undetermined when one port tries to write a ‘H’ and the other tries to write a ‘L’. It is recommended that control logic be implemented to identify this situation if it occurs and do one of the following:

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

Technical Support Assistance

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

Revision History

Revision 1.4, June 2021

Section	Change Summary
Distributed ROM (Distributed_ROM) – PFU-Based	Replaced Figure 9.3. PFU Based Distributed ROM Timing Waveform - without Output Registers and Figure 9.4. PFU PFU Based Distributed ROM Timing Waveform - with Output Registers

Revision 1.3, January 2020

Section	Change Summary
All	• Changed document number from TN1264 to FPGA-TN-02204. • Updated document template.
Disclaimers	Added this section.
Revision History	Updated format.

Revision 1.2, November 2015

Section	Change Summary
All	• Added support for ECP5-5G. • Changed document title to ECP5 and ECP5-5G Memory Usage Guide.
Clarity Designer Flow	Updated Clarity Designer Flow section. Replaced Figure 2.3. Example: Generating Pseudo Dual Port RAM (RAM_DP) Using Clarity Designer.
Technical Support Assistance	Updated Technical Support Assistance section.

Revision 1.1, May 2014

Section	Change Summary
FIFO_DC Flags	Updated FIFO_DC Flags section. Added information on using FIFO_DC with different data widths on read and write ports.

Revision 1.0, March 2014

Section	Change Summary
All	Initial release.