# Memory Usage Guide for Nexus Platform

## Technical Note

FPGA-TN-02094-1.1

June 2020

**Disclaimers**

## Contents

## Figures

## Tables

## Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---|---|
| AW | Address Width |
| DW | Data Width |
| ECC | Error-Correcting Code |
| EBR | Embedded Block RAM |
| FIFO | First In First Out |
| LRAM | Large Random-Access Memory |
| LUT | Look Up Table |
| PFU | Programmable Function Unit |
| PMI | Parameterizable Module Instantiation |
| RAM | Random Access Memory |
| ROM | Read Only Memory |
| SECDED | Single Error Correction-Double Error Detection |
| SRAM | Static Random Access Memory |

# 1. Introduction

This technical note discusses memory usage for the Lattice Semiconductor devices built on the Lattice Nexus™ platform, which include CrossLink™-NX and Certus™-NX. It is intended to be used by design engineers as a guide when integrating the EBR (Embedded Block RAM)-based, PFU (Programmable Function Unit)-based, and peripheral SRAM (Static Random Access Memory) memories for these device families in Lattice Radiant® software.

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

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

- Through the **IP Catalog** – The IP Catalog interface allows you to specify the memory type and size required. The IP Catalog takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.
- Through **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 Radiant project navigator. The parameters and the control signals needed either in Verilog or VHDL can be set. The top-level design has the parameters defined and signals declared so the interface can automatically generate the black box during synthesis.
- Through the **Instantiation of Memory Primitives** – Memory primitives are called directly by the top-level module and instantiated in your design. This is an advanced method and requires a thorough understanding of memory hook-ups and design interfaces.

The remainder of this document discusses these methods.

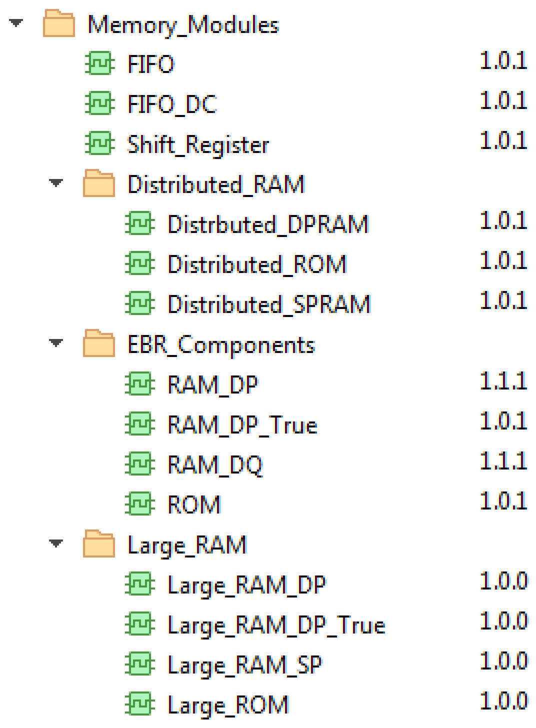
# 2. Memory Generation

You can utilize the 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 modules in the IP Catalog 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 (EBR or LUT)
  - FIFO Single Clock (FIFO)
  - FIFO Dual Clock (FIFO_DC)
- Large RAM
  - Single Port Large RAM (Large_RAM_SP)
  - True Dual Port Large RAM (Large_RAM_DP_True)
  - Pseudo Dual Port Large RAM (Large_RAM_DP)
  - Read Only Memory Large RAM (Large_ROM)

Figure 2.1 shows the memory modules under IP Catalog in Lattice Radiant software.



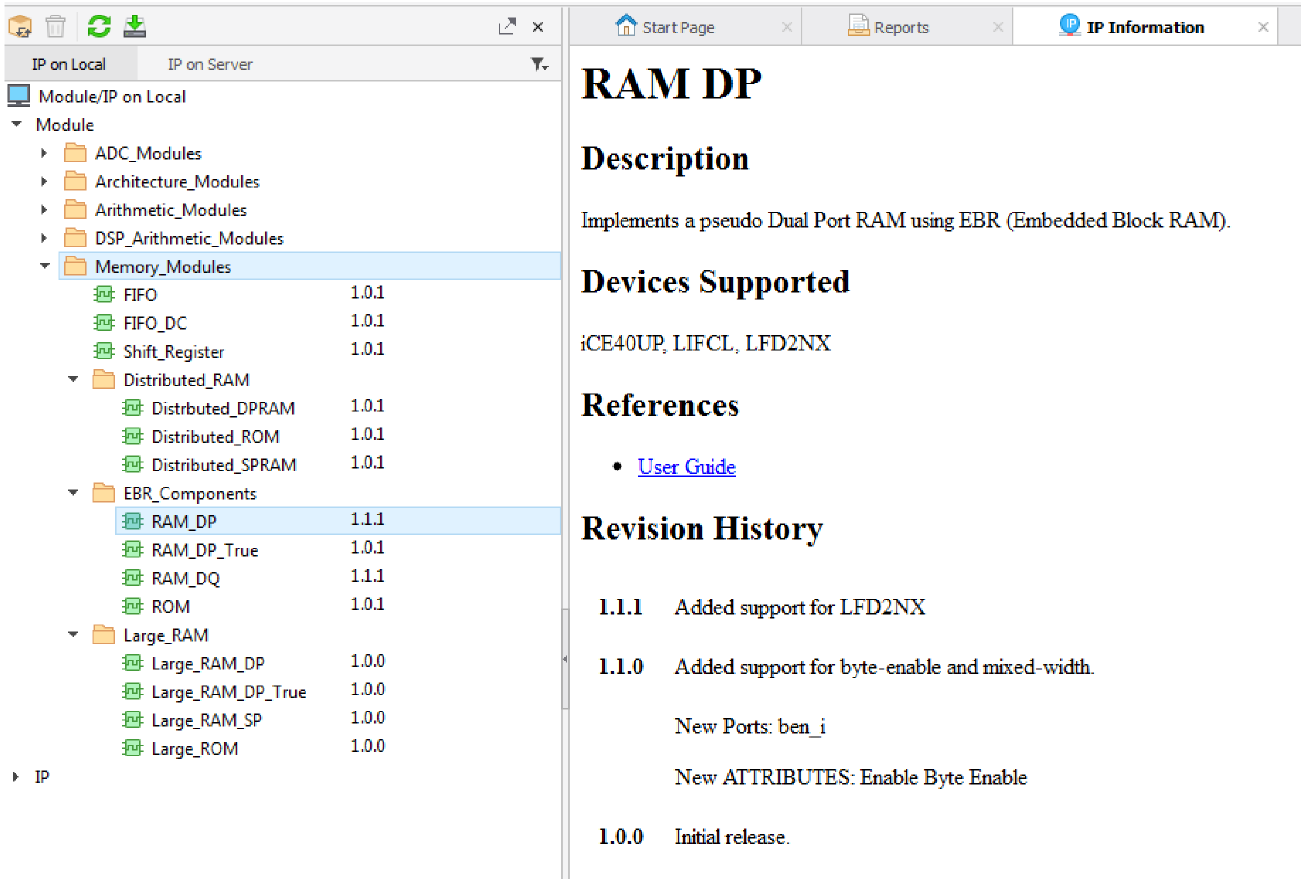
```
Memory_Modules
    FIFO                     1.0.1
    FIFO_DC                  1.0.1
    Shift_Register           1.0.1
    Distributed_RAM
        Distrbuted_DPRAM     1.0.1
        Distributed_ROM      1.0.1
        Distributed_SPRAM    1.0.1
    EBR_Components
        RAM_DP               1.1.1
        RAM_DP_True          1.0.1
        RAM_DQ               1.1.1
        ROM                  1.0.1
    Large_RAM
        Large_RAM_DP         1.0.0
        Large_RAM_DP_True    1.0.0
        Large_RAM_SP         1.0.0
        Large_ROM            1.0.0
```

**Figure 2.1. Memory Modules Available in IP Catalog**

## 2.1. IP Catalog Flow

The IP Catalog allows you to generate, create (or open) any of the available modules for Nexus platform devices.

In the Lattice Radiant software, choose View > Show Views > IP Catalog, or click the IP Catalog icon in the toolbar. This opens the IP Catalog window as shown in Figure 2.2.



RAM DP

Description

Implements a pseudo Dual Port RAM using EBR (Embedded Block RAM).

Devices Supported

iCE40UP, LIFCL, LFD2NX

References

- User Guide

Revision History

| | |
|---|---|
| 1.1.1 | Added support for LFD2NX |
| 1.1.0 | Added support for byte-enable and mixed-width. |
| | New Ports: ben_i |
| | New ATTRIBUTES: Enable Byte Enable |
| 1.0.0 | Initial release. |

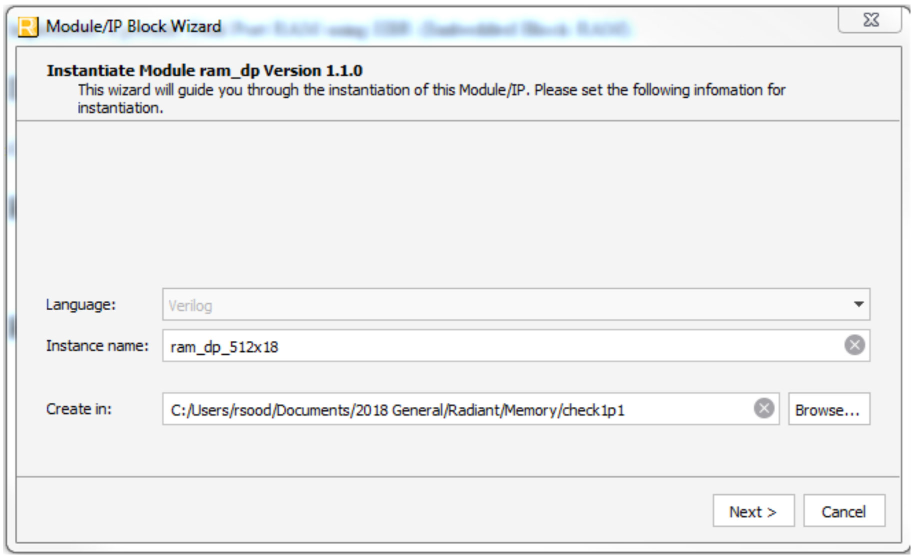
**Figure 2.2. IP Catalog in Lattice Radiant Software**

The left section of the IP Catalog window includes the Module Tree. The Memory Modules are categorized as Distributed RAM, EBR Components FIFOs, Large RAM, and Shift Register. The right section of the window shows the description of the module selected and links to the documentation for additional information.

This section provides an example of generating an EBR-based Pseudo Dual Port RAM of size 512 x 18.

To generate an EBR-based Pseudo Dual Port RAM:

1. Double-click **ram_dp** under the **EBR_Components**.
2. Fill out the information of the module to generate ash shown in Figure 2.3.
3. Click the **Next** button.

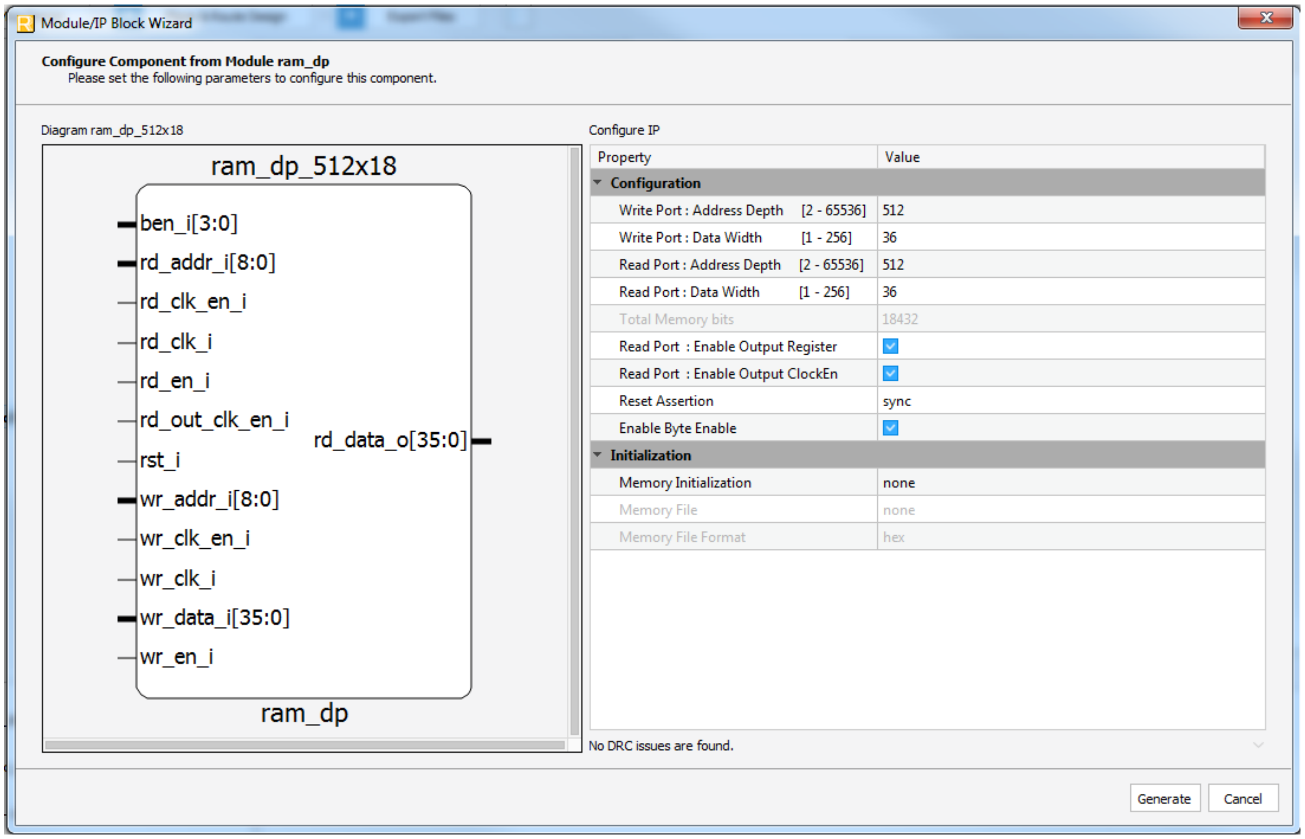

Module/IP Block Wizard

**Instantiate Module ram_dp Version 1.1.0**
This wizard will guide you through the instantiation of this Module/IP. Please set the following infomation for instantiation.

| | |
|---|---|
| Language: | Verilog |
| Instance name: | ram_dp_512x18 |
| Create in: | C:/Users/rsood/Documents/2018 General/Radiant/Memory/check1p1 |

**Figure 2.3. Example: Generating Pseudo Dual Port RAM (RAM_DP) Using IP Catalog**

4. Customize the EBR-based DPRAM in the **Module/IP Block Wizard** window as shown in Figure 2.4.


Module/IP Block Wizard

**Configure Component from Module ram_dp**
Please set the following parameters to configure this component.

Configure IP

| Property | | Value |
|---|---|---|
| **Configuration** | | |
| Write Port : Address Depth | [2 - 65536] | 512 |
| Write Port : Data Width | [1 - 256] | 36 |
| Read Port : Address Depth | [2 - 65536] | 512 |
| Read Port : Data Width | [1 - 256] | 36 |
| Total Memory bits | | 18432 |
| Read Port : Enable Output Register | | ✓ |
| Read Port : Enable Output ClockEn | | ✓ |
| Reset Assertion | | sync |
| Enable Byte Enable | | ✓ |
| **Initialization** | | |
| Memory Initialization | | none |
| Memory File | | none |
| Memory File Format | | hex |

No DRC issues are found.

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

5. When all the options of the module being generated are filled out, click **Generate**.

This module, once in the Lattice Radiant 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 Lattice Radiant 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, create a Verilog or VHDL behavior code for the memory and the synthesis tool automatically identifies it as memory and synthesizes it as a distributed or EBR memory. Memory sizes smaller than 2K bits are automatically mapped to Distributed mode and those larger than 2K bits are implemented using EBRs. This default option can be over-ridden using the RAM_STYLE attribute in 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 Nexus Plaform devices. When instantiating the primitives, you have to work at the EBR block level. In case there is a need to have a memory that spans multiple modules, you are required to create the cascading memory on their own.

Lattice provides library files containing all of the primitives in a VHDL/Verilog file under the cae_library/synthesis folder in the Lattice Radiant 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.

## 3.1. 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 are introduced, either during the process of transmission, or on storage.

EBR-based Memory modules IP Catalog allows you 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 only supported in 512 x 32 EBR configuration mode.

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.

The error flags are aligned to output data and be available in the same cycle as their respective data.

## 3.2. Byte Enable

Byte Enable is a feature available in the selected RAM modules where you can mask the bytes written in the RAM. Each Byte Enable bit controls the enable to 9 bits; the selection can be made in IP Catalog while generating the module.

Each bit of the BE signal corresponds to the corresponding 9-bit selection, starting from LSB side. For example, if you add Byte Enable to an 18-bit wide RAM, then Table 3.1 explains how the written data (Data In) is masked for a 9-bit Byte Size. Bits 8, 17, 26, and 35 are parity bits, which you ignore in x8, x16, and x32 modes.

**Table 3.1. Masked Data in Bits for a 9-Bit Byte Size**

| Byte Enable Bit | Data In Bits that Get Masked (with 9-bit Byte Size) |
|---|---|
| ByteEn(0) | Data(8:0) |
| ByteEn(1) | Data(17:9) |
| ByteEn(2) | Data(26:18) |
| ByteEn(3) | Data(35:27) |

Note 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 you specify the width and depth of the memory in the IP Catalog, the tool generates the memory by depth cascading and/or width cascading or EBR blocks or Distributed RAM primitives. IP Catalog automatically allows you to create memories larger than the width and depth supported for each primitive.

## 4.1. Memory Cascading

For memory sizes that are smaller than what can fit in a single EBR block or the Distributed primitive, the module utilizes the complete block or primitive.

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

### 4.1.1. Input and Output Register

The architecture of the EBR blocks in Nexus platform 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. You can choose this option by selecting the Enable Output Register check box in IP Catalog 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

To correctly write into a memory cell in the EBR block, the correct address should 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 the Lattice Radiant 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

FPGAs built on the Nexus platform support all the features of Single Port Memory Module or RAM_DQ. IP Catalog allows you to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement.

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

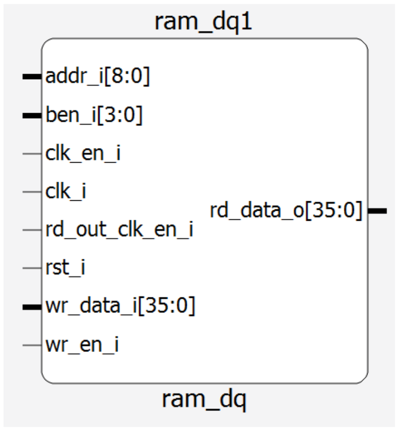


**Figure 4.1. Single-Port Memory Module Generated by IP Catalog**

Figure 4.2 provides the primitive that can be instantiated for the Single Port RAM. The 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 Radiant 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).

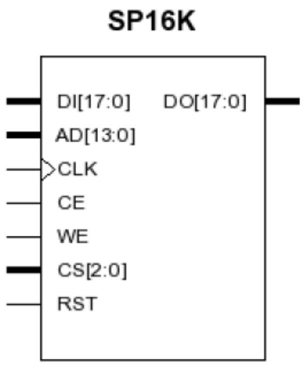


**Figure 4.2. Single Port RAM Primitive for Nexus Platform Devices**

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 IP Catalog.

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

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Clock |
| rst_i | Input | 1 | Reset |
| clk_en_i | Input | 1 | (Input) Clock Enable |
| rd_out_clk_en_i | Input | 1 | Read Output Register Enable (Present is Enable Output Register == TRUE) |
| wr_en_i | Input | 1 | Write Enable |
| wr_data_i | Input | Data Width | Data Input |
| addr_i | Input | Address Width | Address Bus |
| rd_data_o | Output | Data Width | Data Output |
| ben_i | Input | 4 | Byte Enable |

***Note:** Address width is calculated from address depth

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 Nexus Platform 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). You can select some of these attributes through the IP Catalog interface.

The ones without selectable options in IP Catalog are handled by the engine. However, you can access these options if you are working with the direct primitive instantiation.

**Table 4.3. Single-Port Memory Attributes in Nexus Platform 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 — 512 | 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 you to select Byte Enable options. | TRUE, FALSE | FALSE |
| Reset Assertion | Selection for the Reset to be Synchronous or Asynchronous to the Clock. | ASYNC, SYNC | SYNC |
| Initialization | Allows you to initialize their memories to all 1s, 0s or providing a custom initialization by providing a memory file. | 0s, 1s, File | 0s |
| 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 you to select if the memory file is formatted as Binary, Hex or address Hex. | Binary, Hex, Addressed Hex | Binary |

You have the option 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.

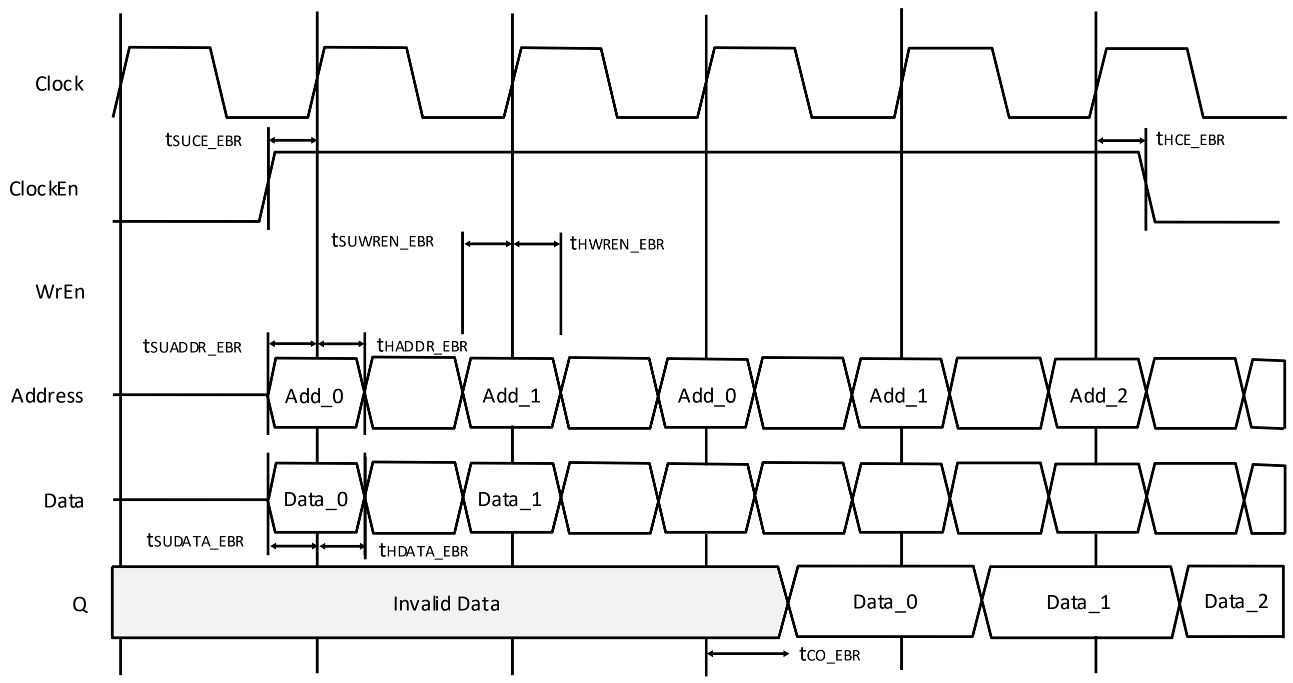


**Figure 4.3. Single Port RAM Timing Waveform, without Output Registers**

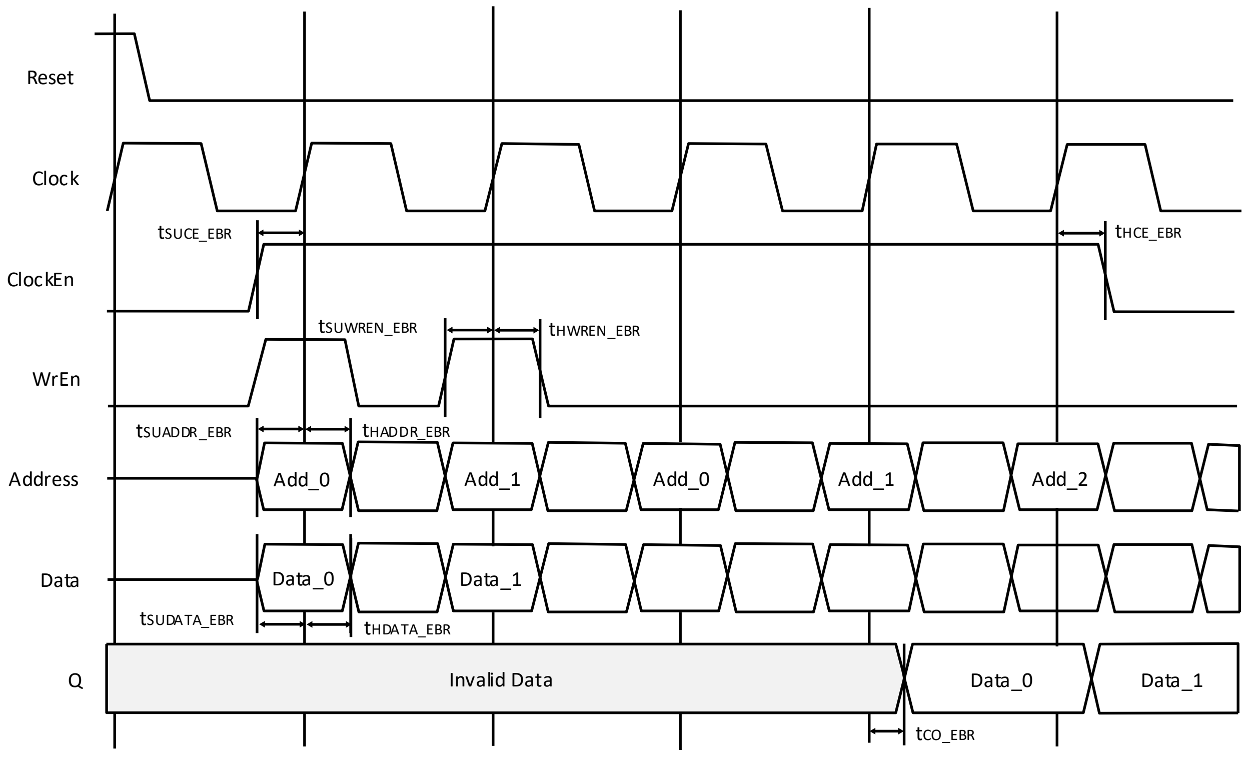


**Figure 4.4. Single Port RAM Timing Waveform, with Output Registers**

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

The EBR blocks in the Nexus platform devices can be configured as True-Dual Port RAM or RAM_DP_TRUE. IP Catalog allows you to generate the Verilog-HDL, VHDL or EDIF netlists for the memory size as per design requirements.

IP Catalog generates the memory module, as shown in Figure 4.5.

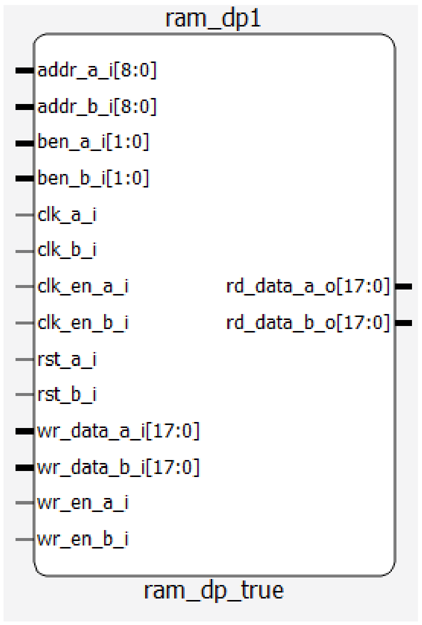


**Figure 4.5. True Dual-Port Memory Module Generated by IP Catalog**

Figure 4.6 provides the primitive that can be instantiated for the True Dual Port RAM. The primitive name is DP16K 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 Radiant software installation folder.

Note 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).

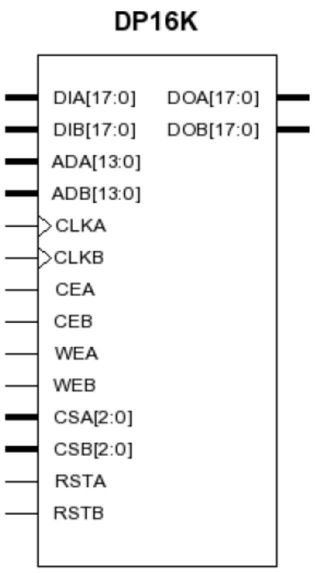


**Figure 4.6. True Dual Port RAM Primitive for Nexus Platform Devices**

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 IP Catalog.

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

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_a_i | Input | 1 | Clock for Port A |
| rst_a_i | Input | 1 | Reset for Port A |
| clk_en_a_i | Input | 1 | Clock Enable for Port A |
| wr_en_a_i | Input | 1 | Write Enable for Port A |
| wr_data_a_i | Input | Data Width | Data Input for Port A |
| addr_a_i | Input | Address Width | Address Bus for Port A |
| rd_data_a_o | Output | Data Width | Data Output for Port A |
| ben_a_i | Input | 2 | Byte Enable for Port A |
| clk_b_i | Input | 1 | Clock for Port B |
| rst_b_i | Input | 1 | Reset for Port B |
| clk_en_b_i | Input | 1 | Clock Enable for Port B |
| wr_en_b_i | Input | 1 | Write Enable for Port B |
| wr_data_b_i | Input | Data Width | Data Input for Port B |
| addr_b_i | Input | Address Width | Address Bus for Port B |
| rd_data_b_o | Output | Data Width | Data Output for Port B |
| ben_b_i | Input | 2 | Byte Enable for Port B |

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 Nexus Platform Devices**

| 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). You can select some of these attributes through the IP Catalog interface.

**Table 4.6. True Dual-Port RAM Attributes for Nexus Platform 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 | 18 |
| 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 | 18 |
| Port A Enable Output Register | Port A Data Out port (QA) can be registered or not using this selection. | TRUE, FALSE | TRUE |
| Port B Enable Output Register | Port B Data Out port (QB) can be registered or not using this selection. | TRUE, FALSE | TRUE |
| Byte Enable A | Allows you to select Byte Enable options | TRUE, FALSE | FALSE |
| Byte Enable B | Allows you to select Byte Enable options | TRUE, FALSE | FALSE |
| Reset Assertion A | Selection for the Reset to be Synchronous or Asynchronous to the Clock | ASYNC, SYNC | SYNC |
| Reset Assertion B | Selection for the Reset to be Synchronous or Asynchronous to the Clock | ASYNC, SYNC | SYNC |
| Initialization | Allows you to initialize their memories to all 1s, 0s or providing a custom initialization by providing a memory file. | 0s, 1s, File | 0s |
| 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 you to select if the memory file is formatted as Binary, Hex or address Hex. | Binary, Hex, Addressed Hex | Binary |

You have the option 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.

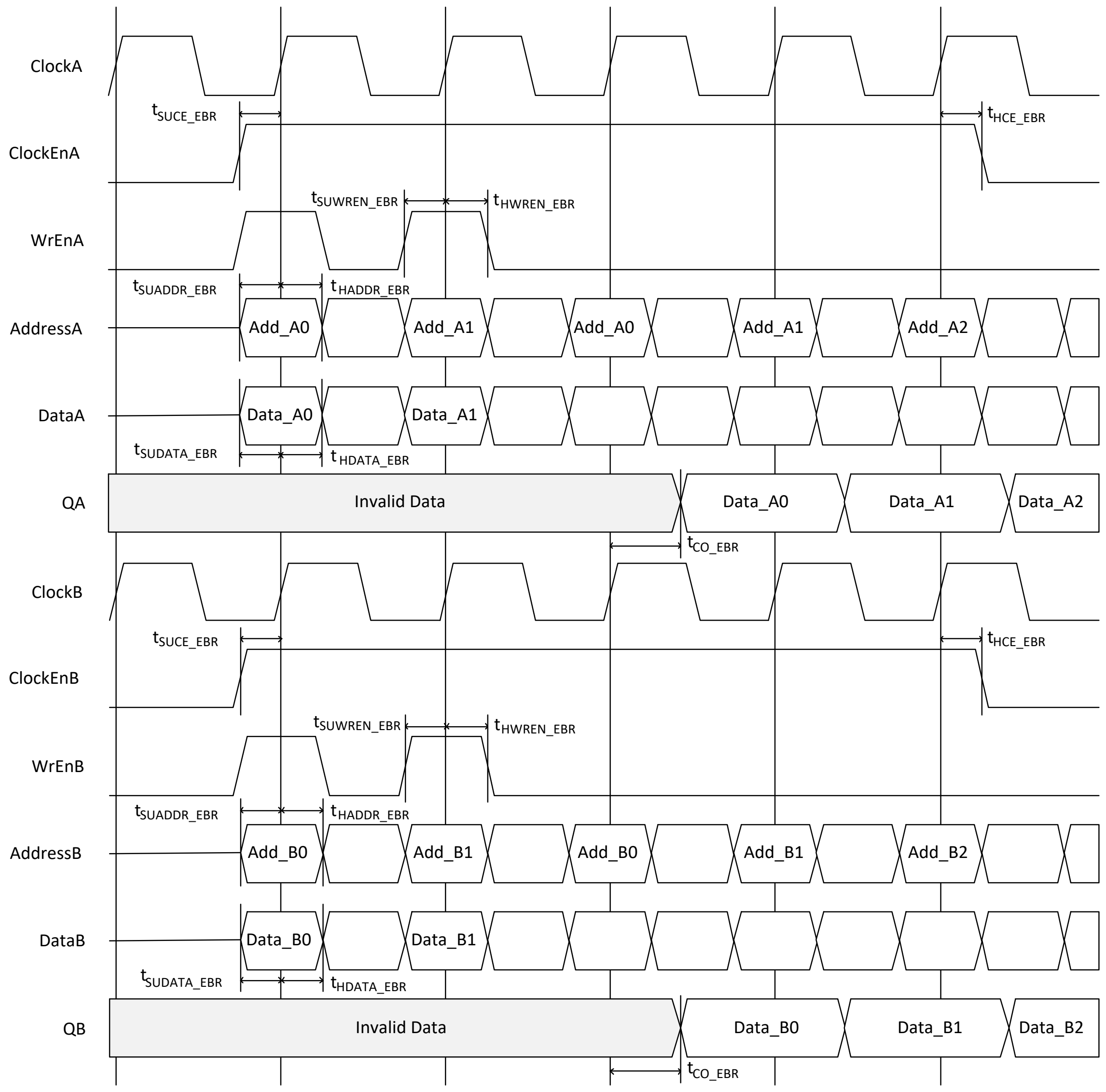


**Figure 4.7. True Dual Port RAM Timing Waveform, without Output Registers**

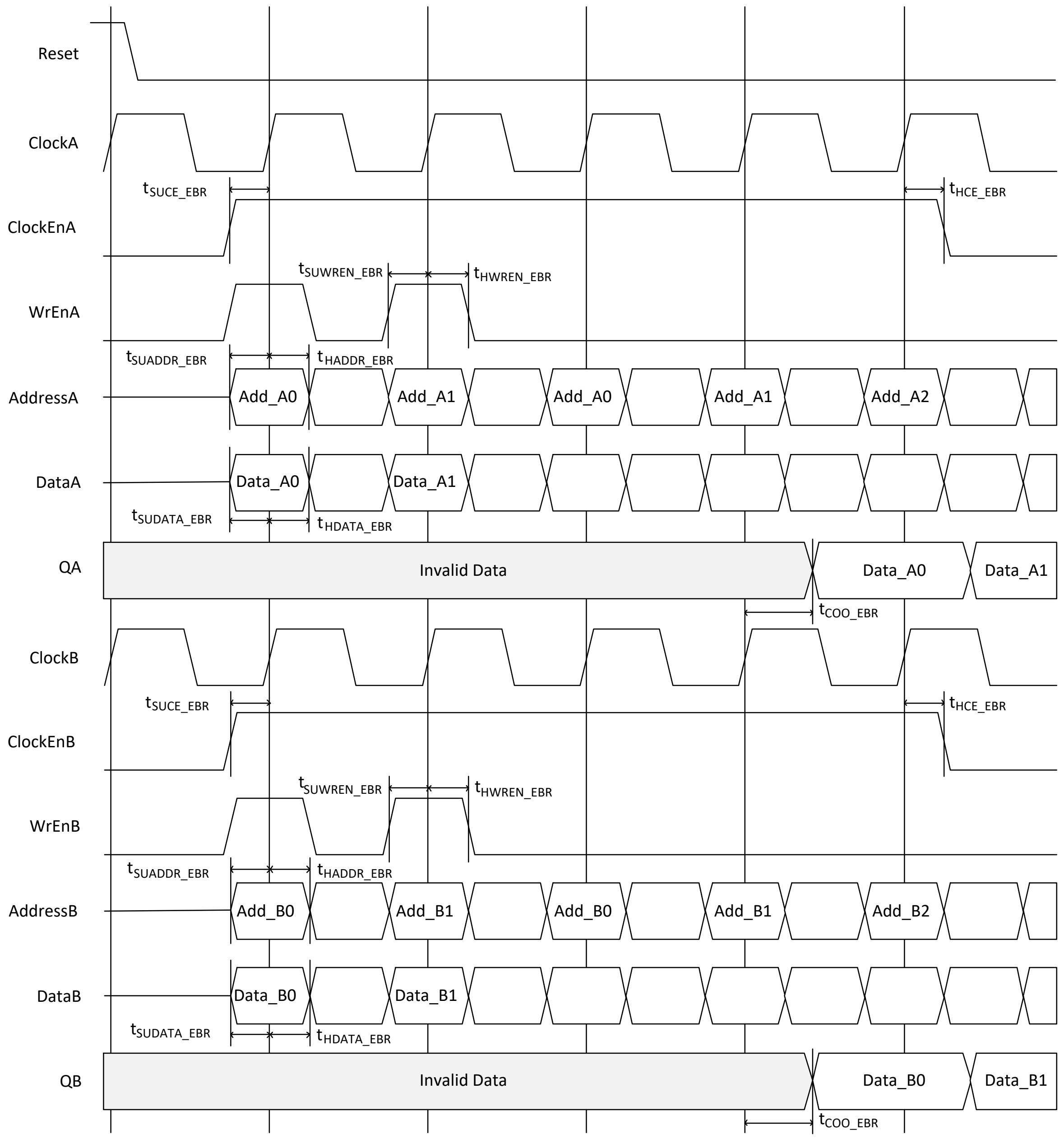


**Figure 4.8. True Dual Port RAM Timing Waveform, with Output Registers**

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

FPGAs built on the Nexus platform support all the features of Pseudo-Dual Port Memory Module or RAM_DP. IP Catalog allows you to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement.

IP Catalog generates the memory module shown in Figure 4.9.

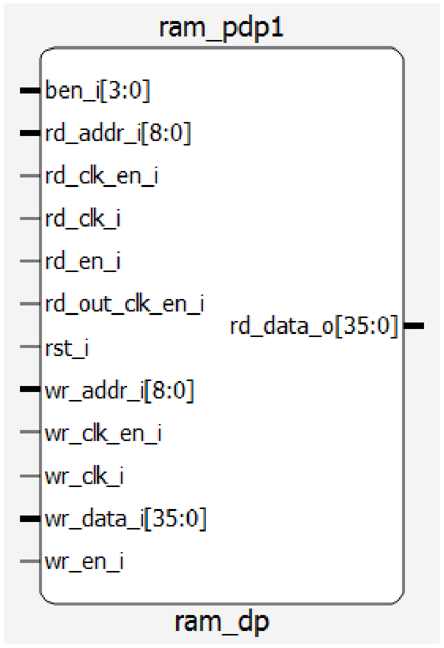


**Figure 4.9. Pseudo Dual-Port Memory Module Generated by IP Catalog**

Figure 4.10 provides the primitive that can be instantiated for the Pseudo-Dual Port RAM. The primitive name is PDPW16K 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 Radiant software installation folder.

Note 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).

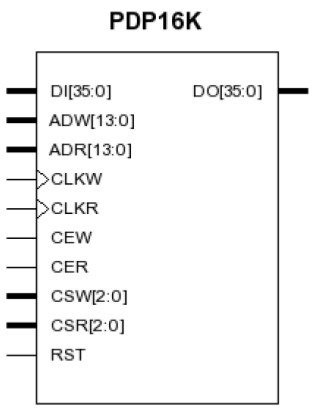


**Figure 4.10. Pseudo-Dual Port RAM Primitive for Nexus Platform Devices**

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 IP Catalog.

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

| Port Name | Direction | Width | Description |
|---|---|---|---|
| wr_clk_i | Input | 1 | Write Clock |
| rd_clk_i | Input | 1 | Read Clock |
| rst_i | Input | 1 | Reset |
| wr_clk_en_i | Input | 1 | Write Clock Enable |
| rd_en_i | Input | 1 | Read Enable |
| rd_clk_en_i | Input | 1 | Read Clock Enable |
| rd_out_clk_en_i | Input | 1 | Read Output Register Clock Enable |
| wr_en_i | Input | 1 | Write Enable |
| ben_i | Input | 4 | Byte Enable |
| wr_data_i | Input | Write Port Data Width | Write Data |
| wr_addr_i | Input | Write Port Address Width | Write Address |
| rd_addr_i | Input | Read Address Width | Read Address |
| rd_data_o | Output | Read Port Data Width | Read Data |

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 Nexus Platform 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). You can select some of these attributes through the IP Catalog interface.

**Table 4.9. Pseudo Dual-Port RAM Attributes for Nexus Platform 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 | 36 |
| 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 |
| Enable Byte Enable | Allows you to select Byte Enable options. | TRUE, FALSE | FALSE |
| Reset Assertion | Selection for the Reset to be Synchronous or Asynchronous to the Clock | ASYNC, SYNC | SYNC |
| Memory Initialization | Allows you to initialize their memories to all 1s, 0s or providing a custom initialization by providing a memory file. | 0s, 1s, File | 0s |
| 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 you to select if the memory file is formatted as Binary, Hex or address Hex. | Binary, Hex, Addressed Hex | Binary |

You 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.

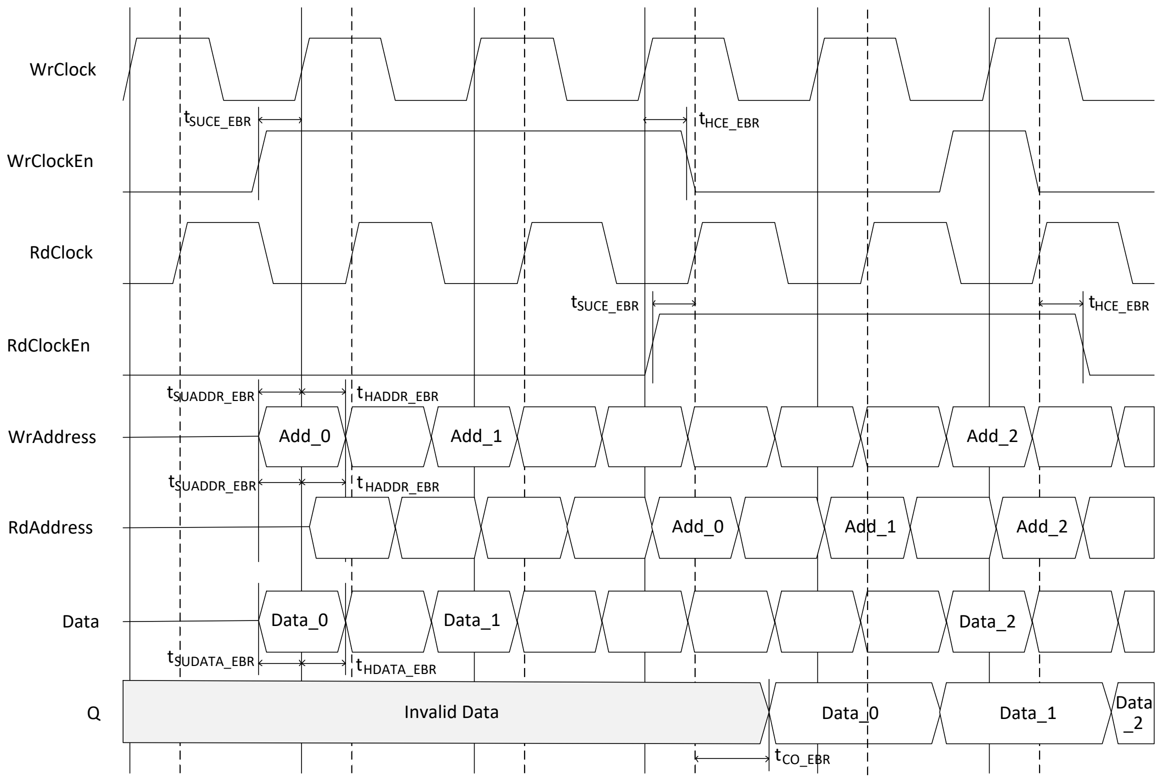


**Figure 4.11. PSEUDO DUAL PORT RAM Timing Diagram - without Output Registers**

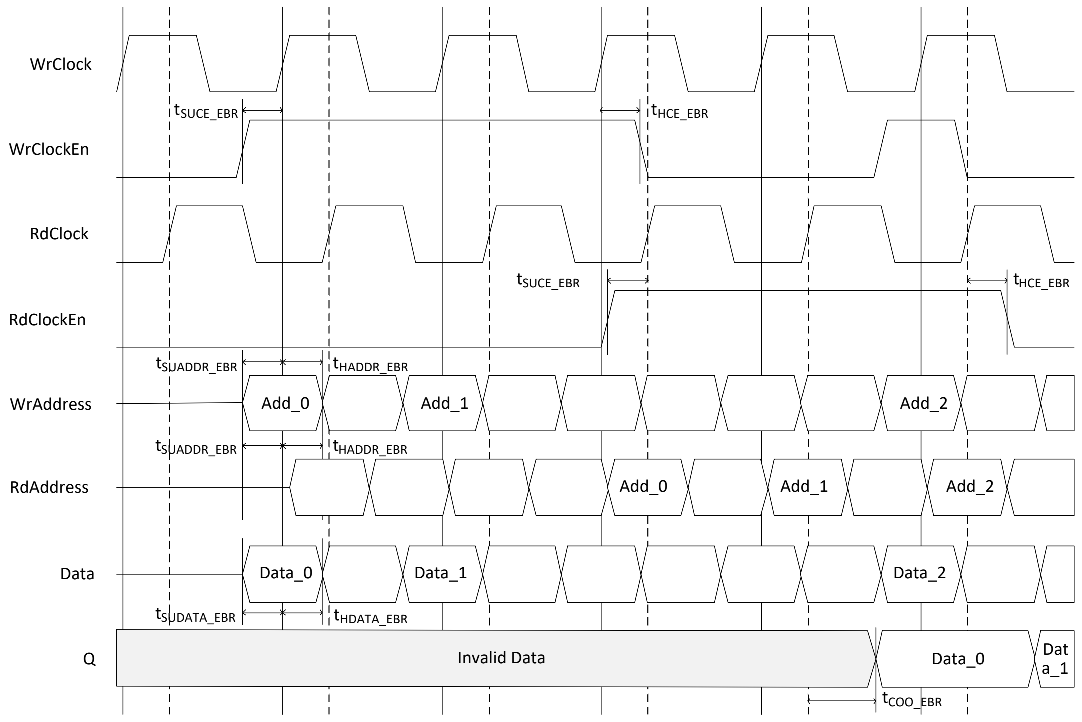


**Figure 4.12. PSEUDO DUAL PORT RAM Timing Diagram - with Output Registers**

## 4.5. Read Only Memory (ROM) – EBR-Based

FPGAs built on the Nexus platform support all the features of ROM Memory Module or ROM. IP Catalog allows you to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement. You are required to provide the ROM memory content in a form of an initialization file.

IP Catalog generates the memory module shown in Figure 4.13.

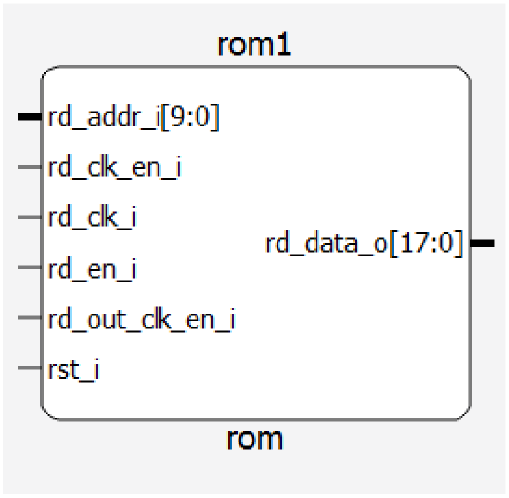


**Figure 4.13. ROM – Read Only Memory Module Generated by IP Catalog**

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

**Table 4.10. EBR-Based ROM Port Definitions**

| Port Name | Direction | Width | Description |
|---|---|---|---|
| rd_clk_i | Input | 1 | Clock |
| rst_i | Input | 1 | Reset |
| rd_en_i | Input | 1 | Read Enable |
| rd_clk_en_i | Input | 1 | Input Clock Enable |
| rd_out_clk_en_i | Input | 1 | Output Clock Enable |
| rd_addr_i | Input | Address Width | Address Bus |
| rd_data_o | Input | Data Width | Data Output |

When generating ROM using IP Catalog, 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 are included in Table 4.11.

**Table 4.11. ROM Memory Sizes for 16K Memory for Nexus Platform 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(8:0) |

Table 4.12 shows the various attributes available for the Read Only Memory (ROM). You can select some of these attributes through the IP Catalog interface.

**Table 4.12. ROM Attributes for Nexus Platform Devices**

| Configuration Tab Attributes | Description | Values | Default Value |
|---|---|---|---|
| Address Depth | Address depth of the read and write port | 2 — 65536 | 1024 |
| Data Width | Data word width of the Read and write port | 1 — 256 | 18 |
| Enable Output Register | Data Out port (Q) can be registered or not using this selection. | TRUE, FALSE | TRUE |
| Enable Output Clock | Enables read output clock | TRUE, FALSE | FALSE |
| Reset Assertion Mode | Selection for the Reset to be Synchronous or Asynchronous to the Clock | ASYNC, SYNC | SYNC |
| Initialization | Allows you to initialize their memories to all 1s, 0s or providing a custom initialization by providing a memory file. | 0s, 1s, File | 0s |
| 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 you to select if the memory file is formatted as Binary, Hex or address Hex. | Binary, Hex, Addressed Hex | Binary |
| Enable ECC | Option allows you to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits. | TRUE, FALSE | FALSE |
| Address Depth | Address depth of the read and write port | 2 — 65536 | 1024 |

You have the option to enable the output registers for Read Only Memory (ROM). Figure 4.14 and Figure 4.15 show the internal timing waveforms for ROM with these options.

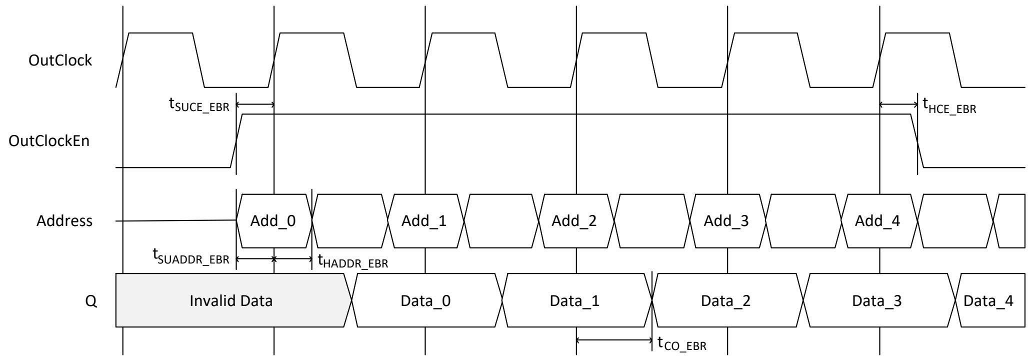


**Figure 4.14. ROM Timing Waveform - without Output Registers**

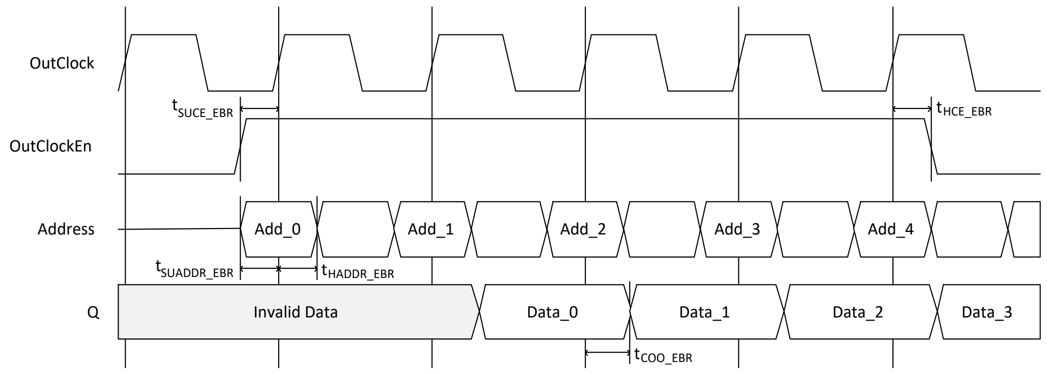


**Figure 4.15. ROM Timing Waveform - with Output Registers**

# 5. First In First Out (FIFO) Memory

Nexus platform devices support two different types of FIFOs:

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

The EBR blocks in Nexus platform 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). IP Catalog allows you to generate the Verilog-HDL or VHDL netlist for various memory sizes depending on design requirements.

IP Catalog generated FIFO modules and their operation are discussed in detail in the following pages.

## 5.1. Single Clock FIFO (FIFO) – EBR and LUT

Figure 5.1 shows the module that is generated by the IP Catalog for FIFO.

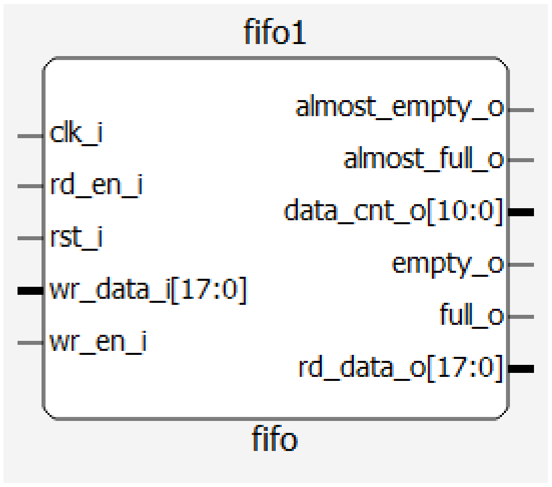


**Figure 5.1. FIFO Module Generated by IP Catalog**

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 | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Write Clock |
| rst_i | Input | 1 | Reset |
| wr_en_i | Input | 1 | Write Enable |
| rd_en_i | Input | 1 | Read Enable |
| wr_data_i | Input | Data Width | Write Data |
| rd_data_o | Output | Data Width | Read Data |
| full_o | Output | 1 | Full Flag |
| empty_o | Output | 1 | Empty Flag |
| almost_full_o | Output | 1 | Almost Full Flag |
| almost_empty_o | Output | 1 | Almost Empty Flag |
| data_cnt_o | Output | Address Width | Data Counter Width based on MEM Address Width |

**Table 5.2. FIFO Attributes for Nexus Platform Devices**

| Configuration Tab Attributes | Description | Values | Default Value |
|---|---|---|---|
| Implementation Type | EBR-Based or LUT-Based | EBR, LUT | EBR |
| Address Depth | Address depth of the read and write port (values are powers of 2) | 2 – <Max that can fit in the device> | 1024 |
| Data Width | Data word width of the Read and write port | 1 – 512 | 18 |
| Enable Output Register | Data Out port (Q) can be registered or not using this selection. | TRUE, FALSE | TRUE |
| Reset Assertion | Selection for the Reset to be Synchronous or Asynchronous to the Clock | ASYNC, SYNC | SYNC |
| Almost Empty Flag | Enables the generation of Almost Empty Flag | TRUE, FALSE | TRUE |
| Almost Empty Assertion Type | This option allows you to select the type of threshold to be used for Almost Empty flag. Static threshold is set by constant parameter while Dynamic threshold is set though the input ports.<br>Single threshold provides only the assertion level while Dual threshold provides both assertion and de-assertion level. | Static-Single,<br>Static-Dual,<br>Dynamic-Single,<br>Dynamic-Dual | Static-Dual |
| (Almost Empty Threshold) Assert | This option allows you to set the assertion level of Almost Empty Flag. This is applicable for Static-(Single/Dual) threshold mode. | 1 – Address Depth | 1 |
| (Almost Empty Threshold) Deassert | This option allows you to set the de-assertion level of Almost Empty Flag after it goes high. This is applicable only for Static-Dual threshold mode. | 1 – Address Depth | 4 |
| Almost Full Flag | Enables the generation of Almost Full Flag | TRUE, FALSE | TRUE |
| Almost Full Assertion Type | This option allows you to select the type of threshold to be used for Almost Full flag. Static threshold is set by constant parameter while Dynamic threshold is set though the input ports.<br>Single threshold provides only the assertion level while Dual threshold provides both assertion and de-assertion level. | Static-Single,<br>Static-Dual,<br>Dynamic-Single,<br>Dynamic-Dual | Static-Dual |
| (Almost Full Threshold) Assert | This option allows you to set the assertion level of Almost Full Flag. Applicable for Static-(Single/Dual) threshold mode. | 1 – Address Depth | 1023 |
| (Almost Full Threshold) Deassert | This option allows you to set the de-assertion level of Almost Full Flag after it goes high. This option is applicable only for Static-Dual threshold mode. | 1 – Address Depth | 1020 |
| Data Count | This option allows you to enable generation of write data count. | TRUE, FALSE | FALSE |

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.

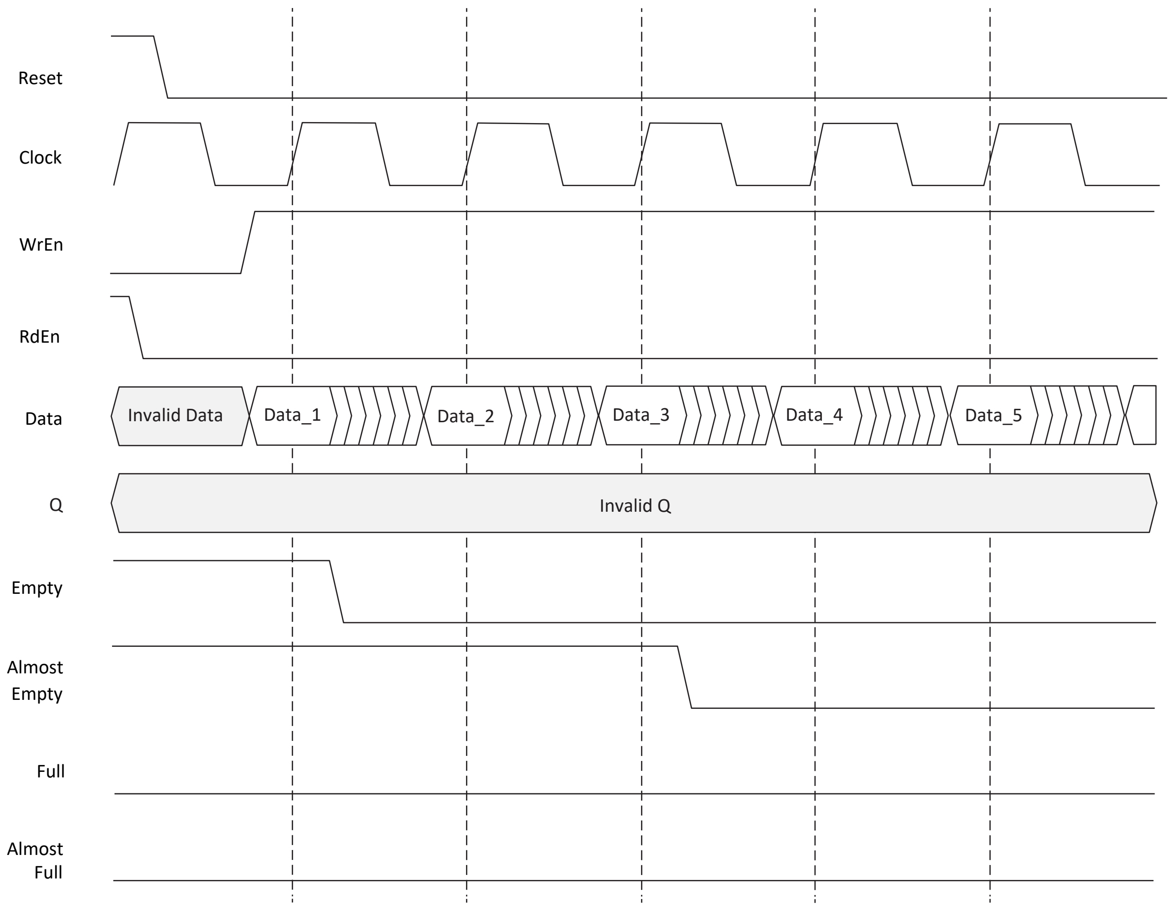


**Figure 5.2. FIFO Without Output Registers, Start of Data Write Cycle**

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

Assume that you 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 it is assumed that the FIFO depth is *N*.

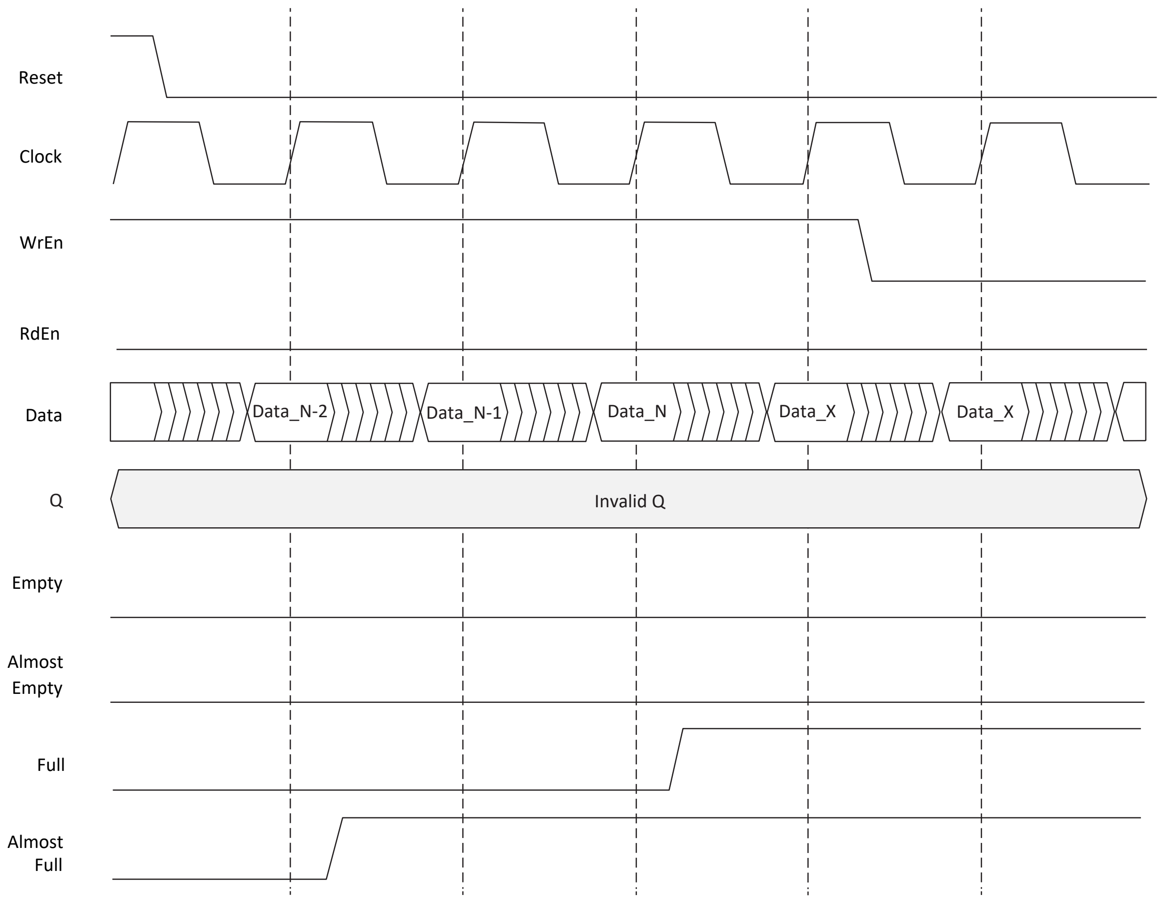


**Figure 5.3. FIFO Without Output Registers, End of Data Write Cycle**

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).

Examine 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.

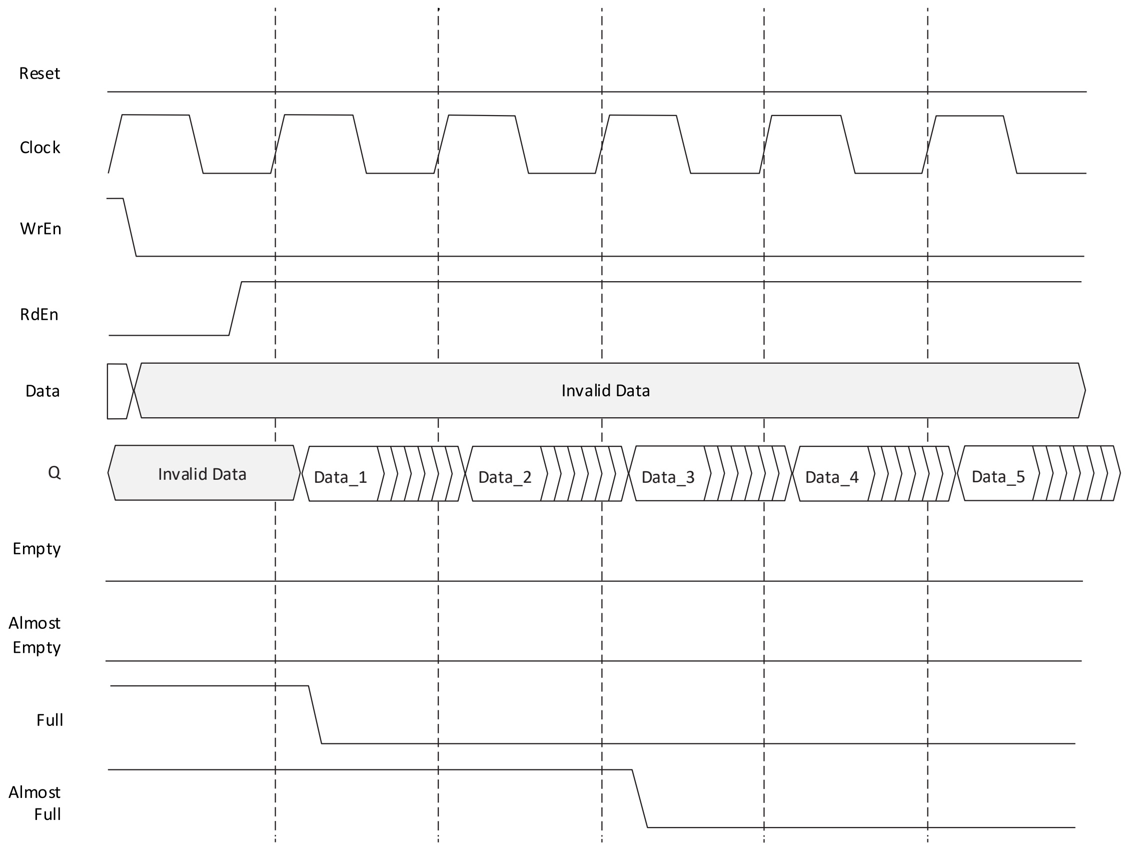


**Figure 5.4. FIFO Without Output Registers, Start of Data Read Cycle**

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

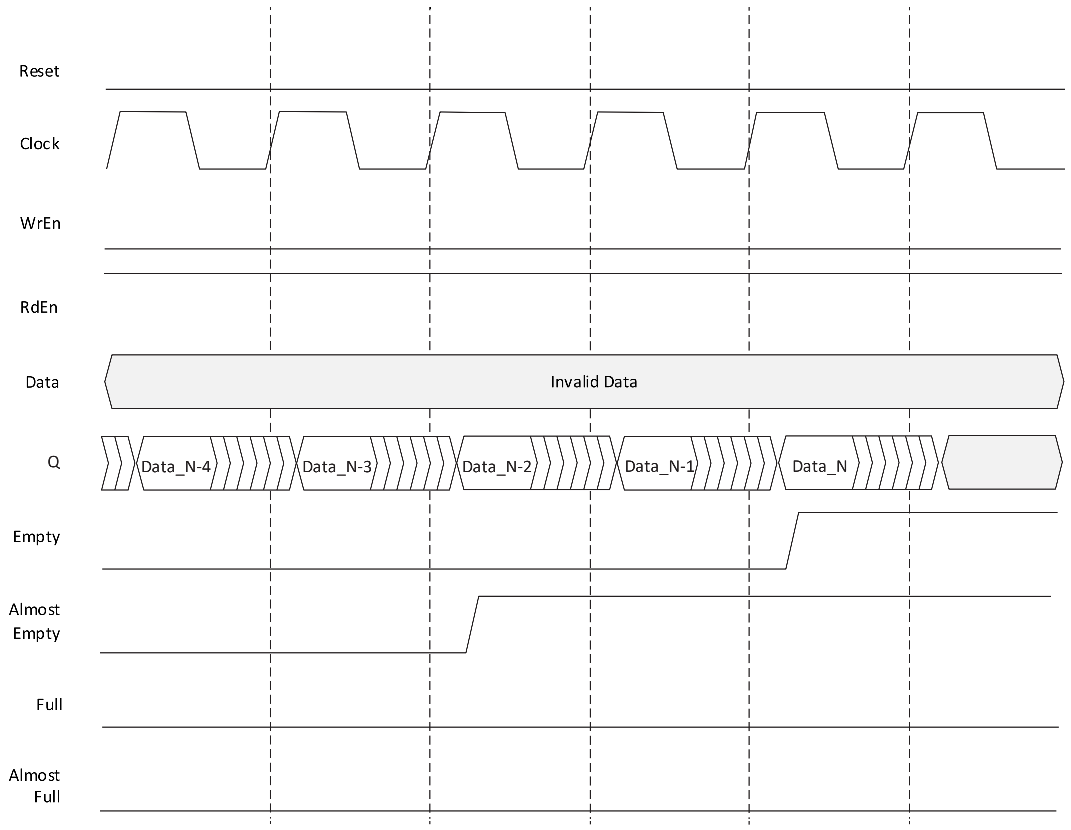


**Figure 5.5. FIFO Without Output Registers, End of Data Read Cycle**

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* is delayed by one clock cycle.

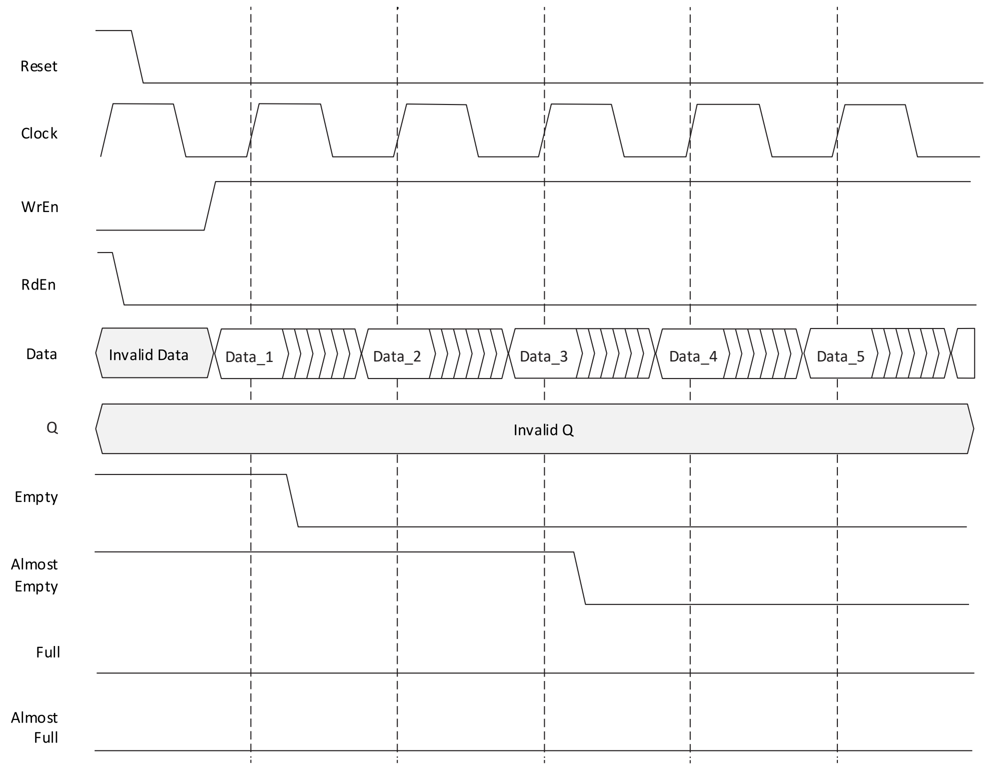


**Figure 5.6. FIFO with Output Registers, Start of Data Write Cycle**

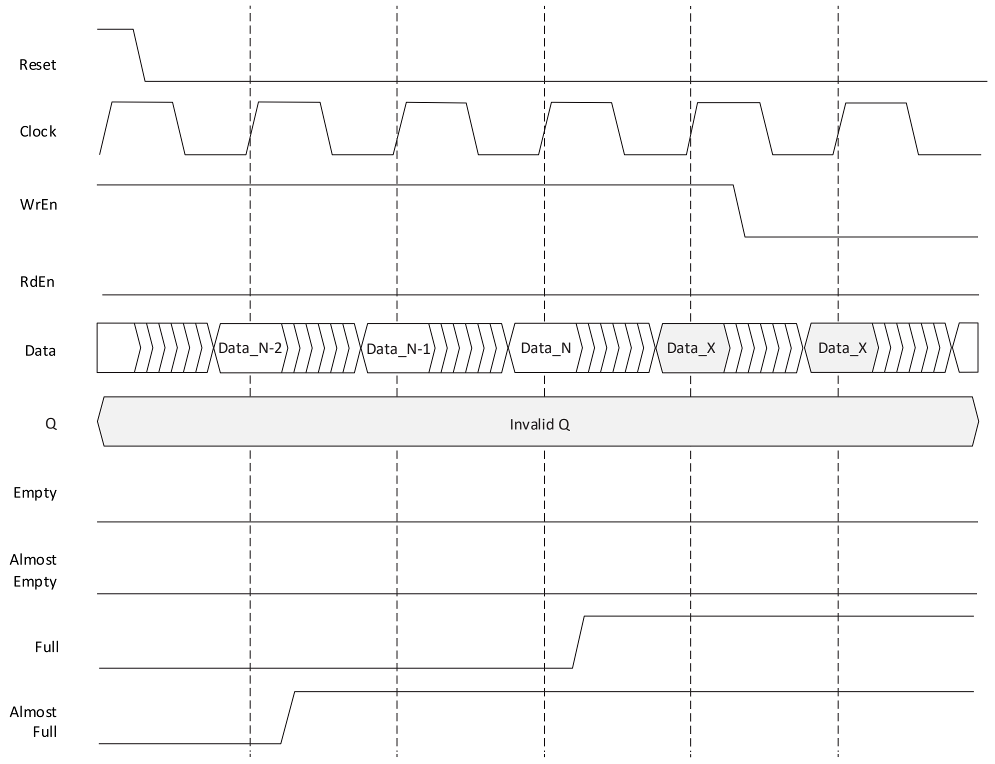


**Figure 5.7. FIFO with Output Registers, End of Data Write Cycle**

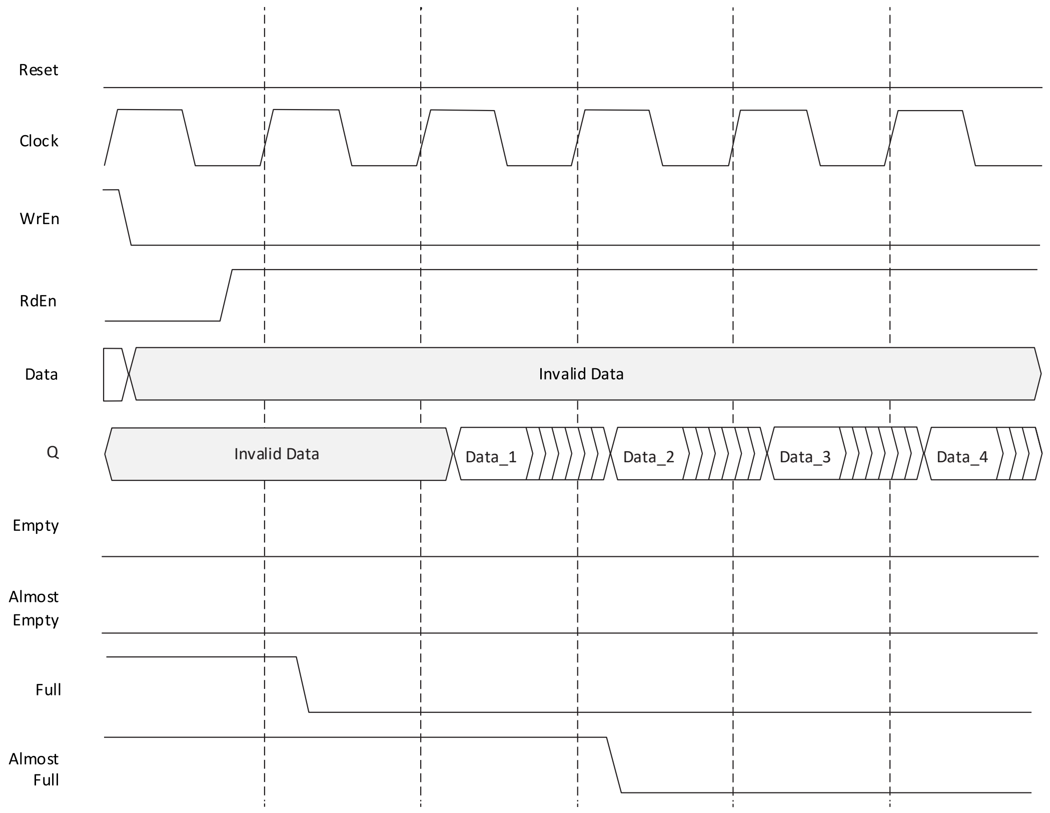


**Figure 5.8. FIFO with Output Registers, Start of Data Read Cycle**

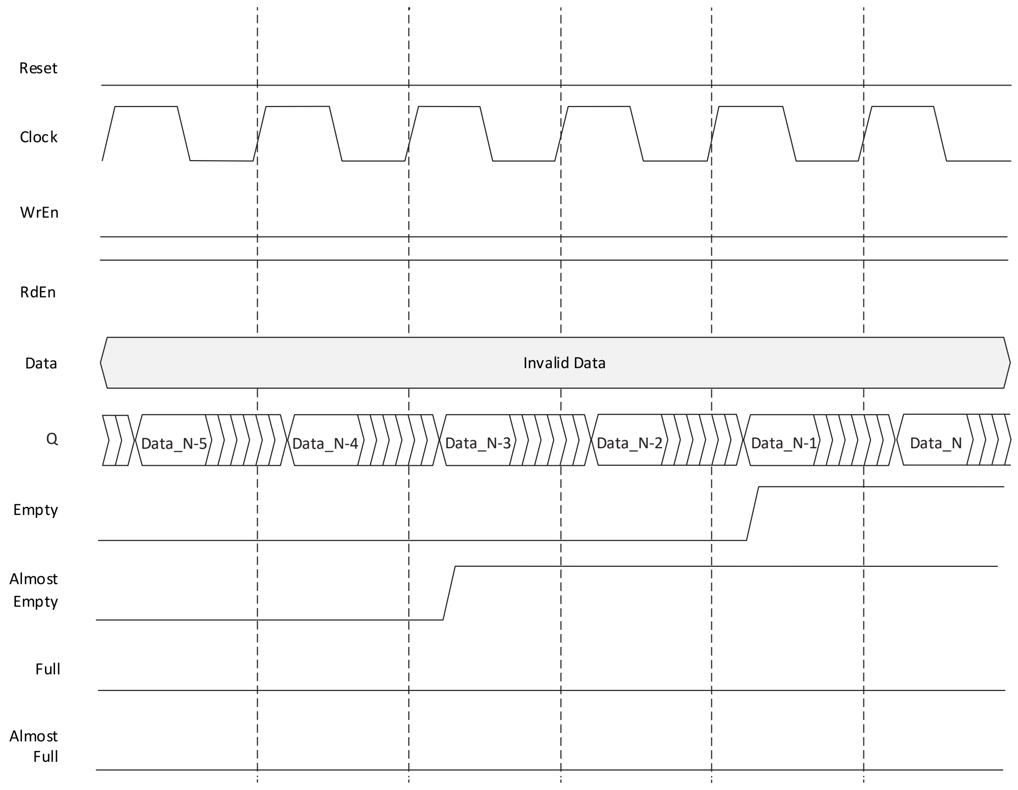


**Figure 5.9. FIFO with Output Registers, End of Data Read Cycle**

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.

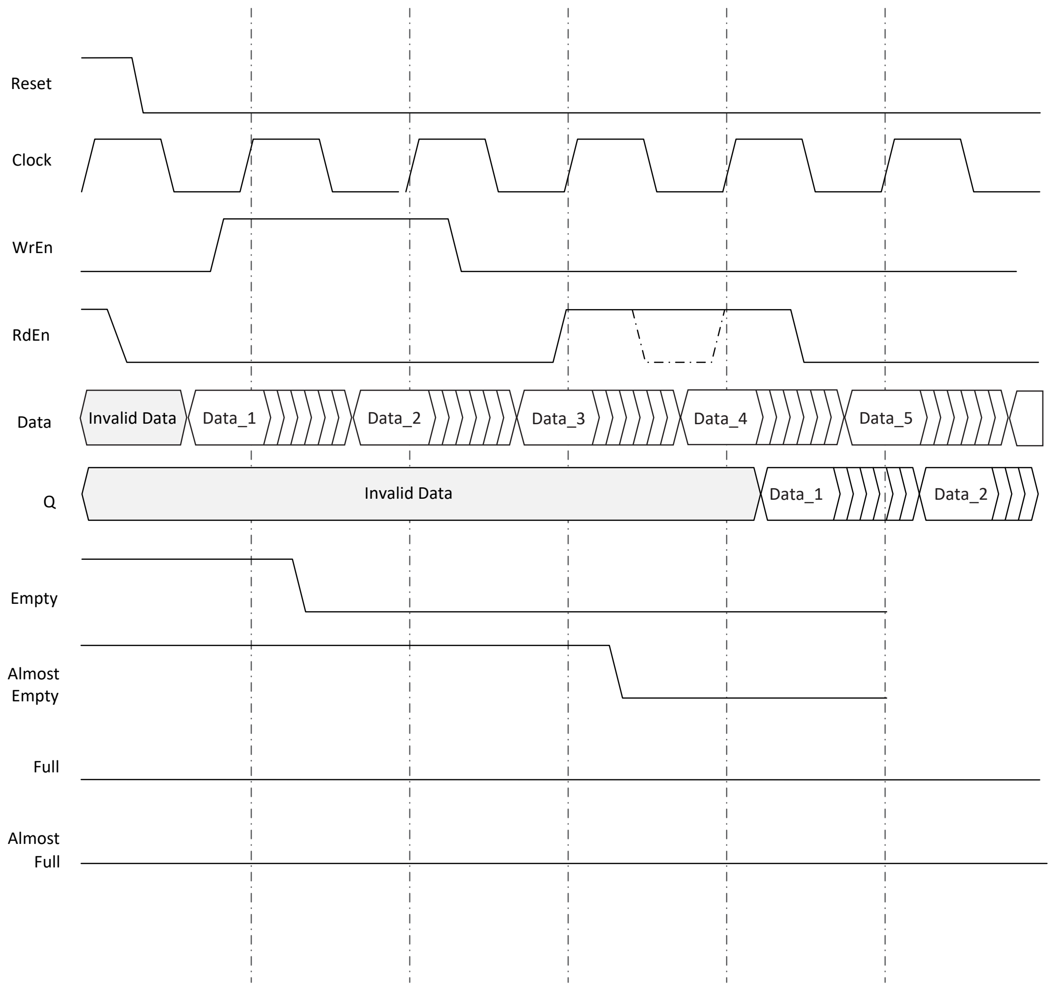


**Figure 5.10. FIFO with Output Registers and RdEn on Output Registers**

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

Figure 6.1 shows the module that is generated by the IP Catalog for FIFO.

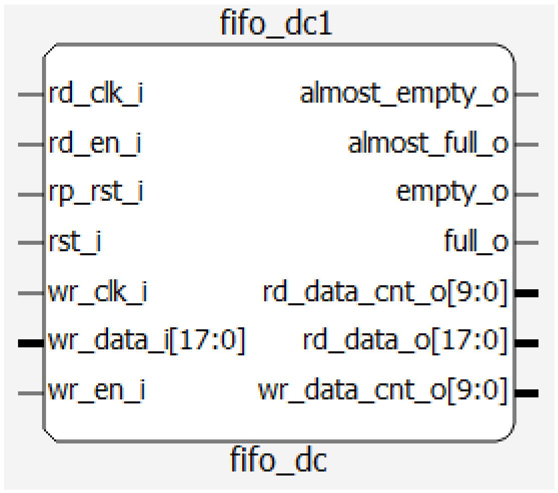


**Figure 6.1. FIFO_DC Module Generated by IP Catalog**

The various ports and their definitions for the FIFO_DC are listed in Table 6.1. The software attributes are listed in Table 6.2.

**Table 6.1. Port Names and Definitions for FIFO_DC**

| Port Name | Direction | Width | Description |
|---|---|---|---|
| wr_clk_i | Input | 1 | Write Clock |
| rd_clk_i | Input | 1 | Read Clock |
| rst_i | Input | 1 | Reset |
| rp_rst_i | Input | 1 | Read Pointer Reset |
| wr_en_i | Input | 1 | Write Enable |
| rd_en_i | Input | 1 | Read Enable |
| wr_data_i | Input | Data Width | Write Data |
| rd_data_o | Output | Data Width | Read Data |
| full_o | Output | 1 | Full Flag |
| empty_o | Output | 1 | Empty Flag |
| almost_full_o | Output | 1 | Almost Full Flag |
| almost_empty_o | Output | 1 | Almost Empty Flag |
| wr_data_cnt_o | Output | Address Width | Write Data Counter |
| rd_data_cnt_o | Output | Address Width | Read Data Counter |

**Table 6.2. FIFO_DC Attributes for Nexus Platform Devices**

| Configuration Tab Attributes | Description | Values | Default Value |
|---|---|---|---|
| Implementation Type | EBR-Based or LUT-Based | EBR, LUT | EBR |
| Write Address Depth | Address depth of the write port (values are powers of 2) | 2 – <Max that can fit in the device> | 512 |
| Write Data Width | Data word width write port | 1 – 512 | 18 |
| Read Address Depth | Address depth of the read port. Note: If not equal to Write Address Depth, the valid values are powers of two such that the ratio between Write and Read data widths are also powers of 2. | 2 – <Max that can fit in the device> | 512 |
| Enable Output Register | Data Out port (rd_data_o) can be registered or not using this selection. | True, False | False |
| Reset Mode | Selection for the Reset to be Synchronous or Asynchronous to the Clock | async, sync | sync |
| Almost Empty Flag | Enables the generation of Almost Empty Flag | TRUE, FALSE | TRUE |
| Almost Empty Threshold Mode | This option allows you to select the type of threshold to be used for Almost Empty flag. Static threshold is set by constant parameter while Dynamic threshold is set though the input ports. Single threshold provides only the assertion level while Dual threshold provides both assertion and de-assertion level. | Static-Single, Static-Dual, Dynamic-Single, Dynamic-Dual | Static-Dual |
| (Almost Empty Threshold) Assert | This option allows you to set the assertion level of Almost Empty Flag. This is applicable for Static-(Single/Dual) threshold mode. | 1 – Address Depth | 1 |
| (Almost Empty Threshold) Deassert | This option allows you to set the de-assertion level of Almost Empty Flag after it goes high. This is applicable only for Static-Dual threshold mode. | 1 – Address Depth | 4 |
| Almost Full Flag | Enables the generation of Almost Full Flag | True, False | False |
| Almost Full Threshold Mode | This option allows you to select the type of threshold to be used for Almost Full flag. Static threshold is set by constant parameter while Dynamic threshold is set though the input ports. Single threshold provides only the assertion level while Dual threshold provides both assertion and de-assertion level. | Static-Single, Static-Dual, Dynamic-Single, Dynamic-Dual | Static-Dual |
| (Almost Full Threshold) Assert | This option allows you to set the assertion level of Almost Full Flag. This is applicable for Static-(Single/Dual) threshold mode. | 1 – Address Depth | 511 |
| (Almost Full Threshold) Deassert | This option allows you to set the de-assertion level of Almost Full Flag after it goes high. This is applicable only for Static-Dual threshold mode. | 1 – Address Depth | 508 |
| Data Count (Synchronized to Write clock) | This options allows you to enable generation of write data count. | True, False | False |
| Data Count (Synchronized to Read clock) | This options allows you to enable generation of read data count. | True, False | False |

## 6.1. FIFO_DC Flags

As a hardware FIFO, FIFO_DC avoids latency to the flags during assertion or de-assertion, which distinguishes it from devices with emulated FIFO.

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.

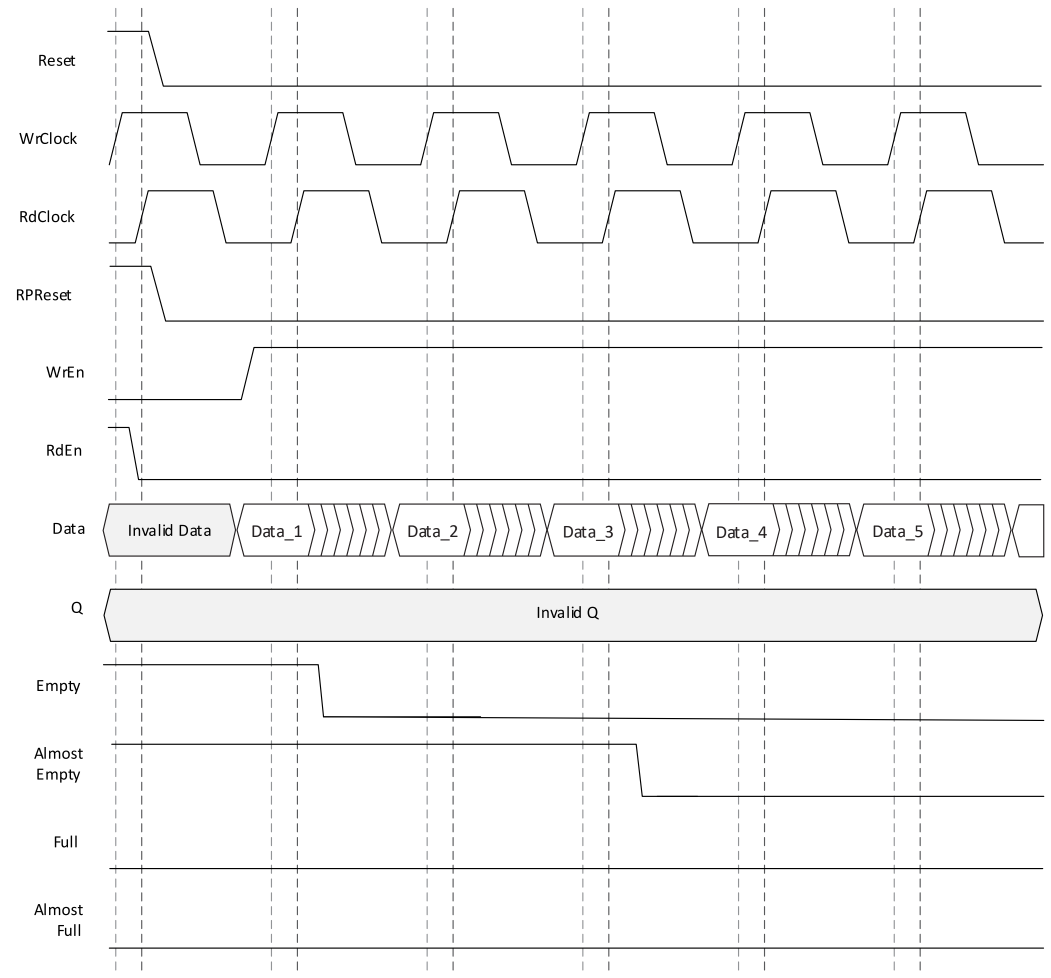


**Figure 6.2. FIFO_DC Without Output Registers, Start of Data Write Cycle**

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

Assume that you continue to write into the FIFO_DC to fill it. When the FIFO_DC is filled, the Almost Full and Full Flags are asserted. Figure 6.3 shows the behavior of these flags. In this figure, it is assumed that FIFO_DC depth is *N*.

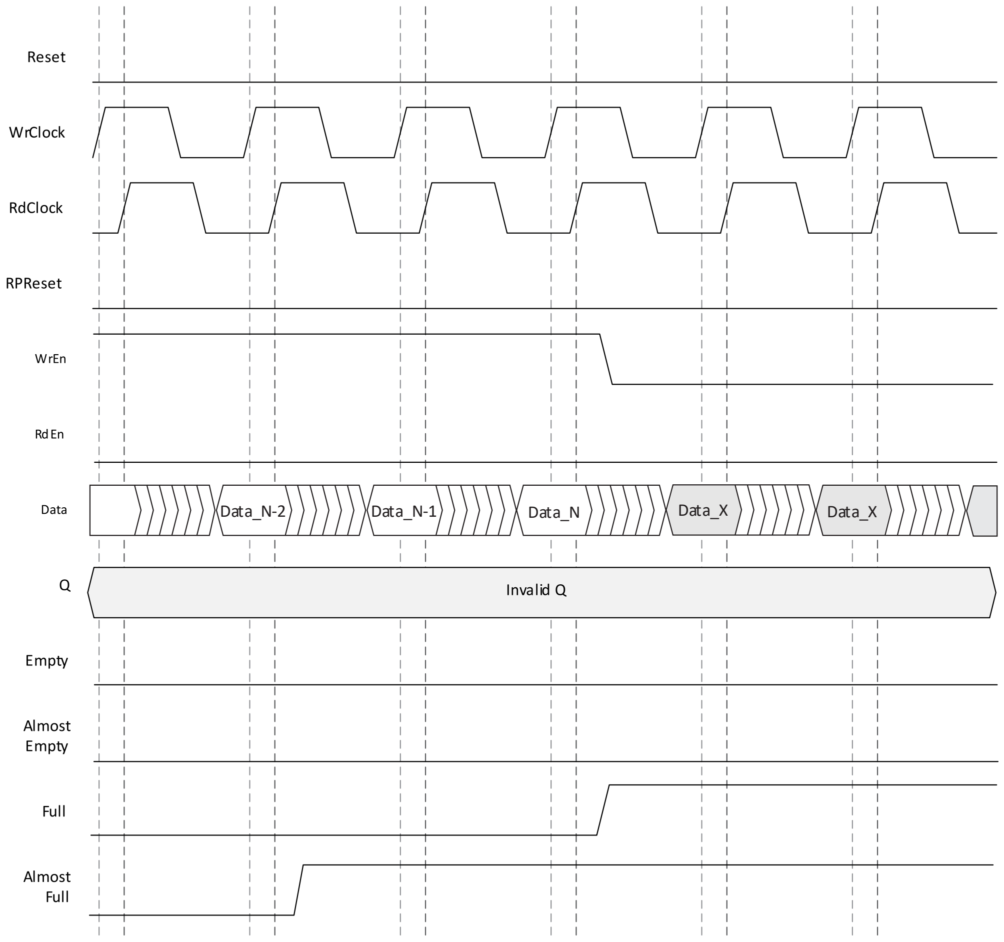


**Figure 6.3. FIFO_DC Without Output Registers, End of Data Write Cycle**

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.

Examine 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.

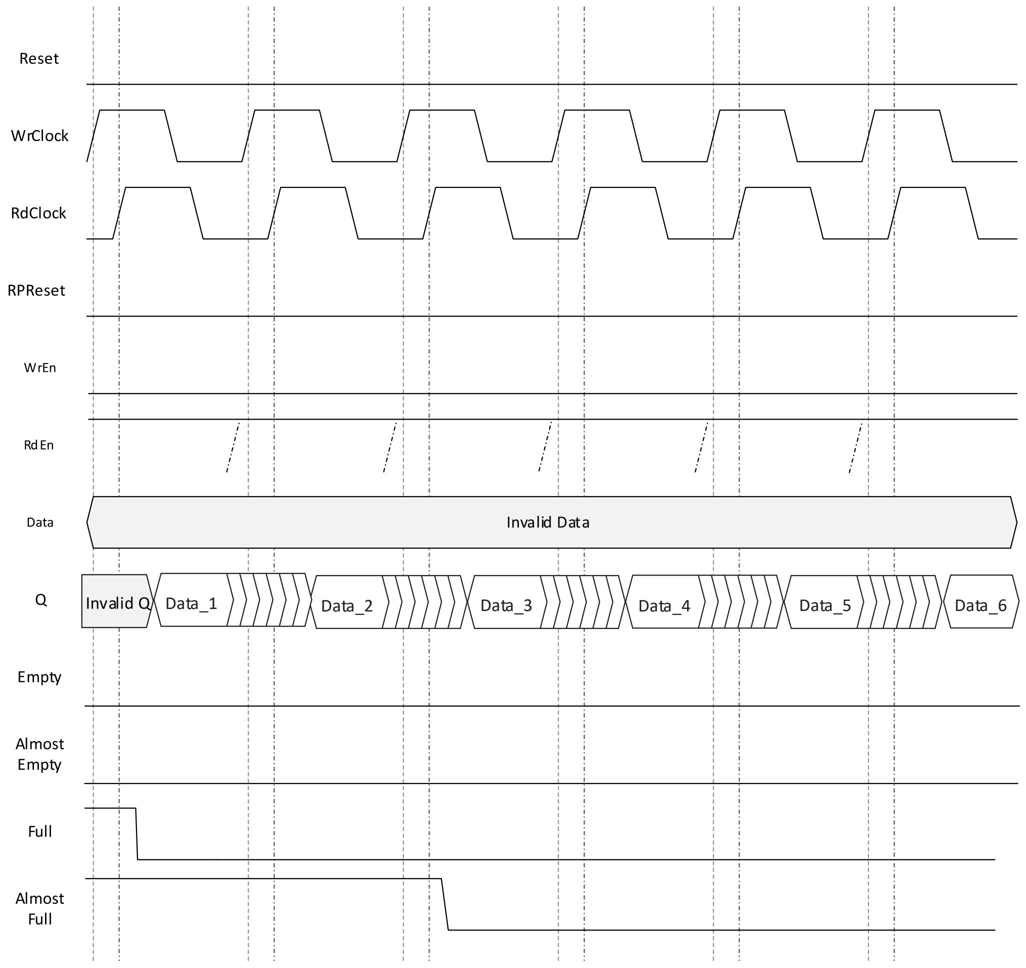


**Figure 6.4. FIFO_DC Without Output Registers, Start of Data Read Cycle**

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

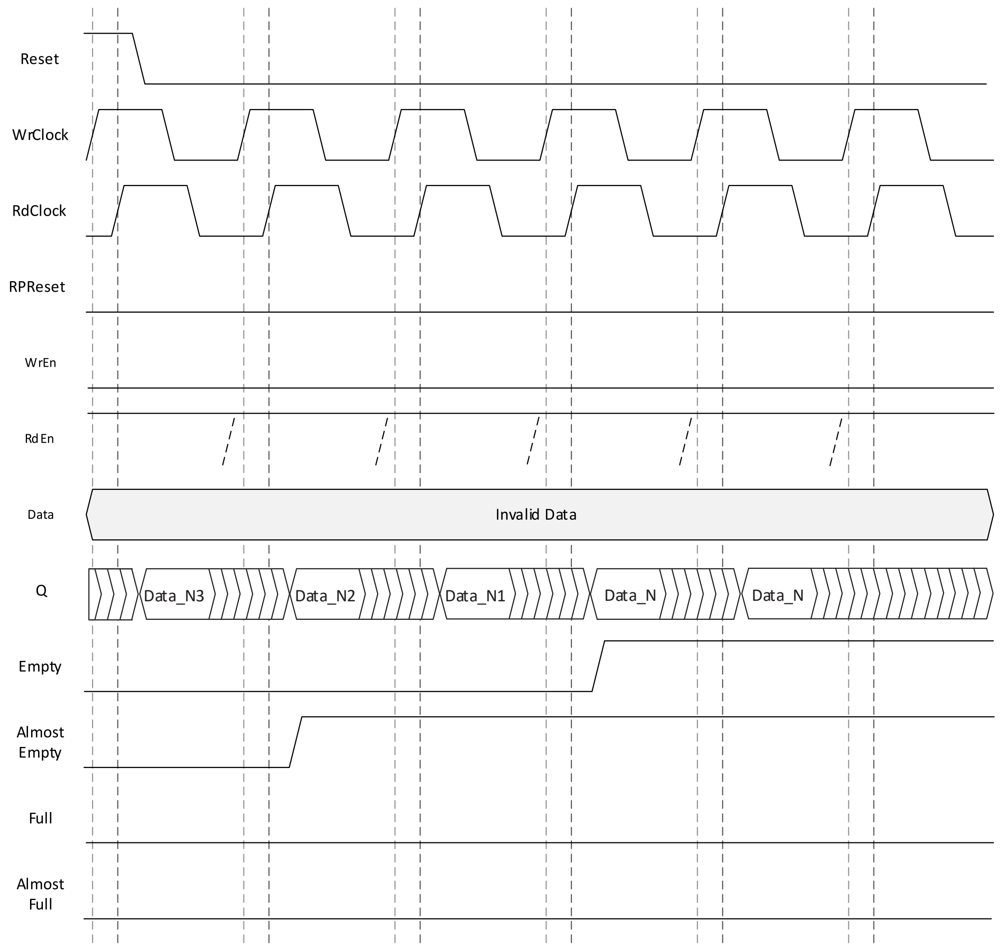


**Figure 6.5. FIFO_DC Without Output Registers, Start of Data Read Cycle**

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

Figure 6.6 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.

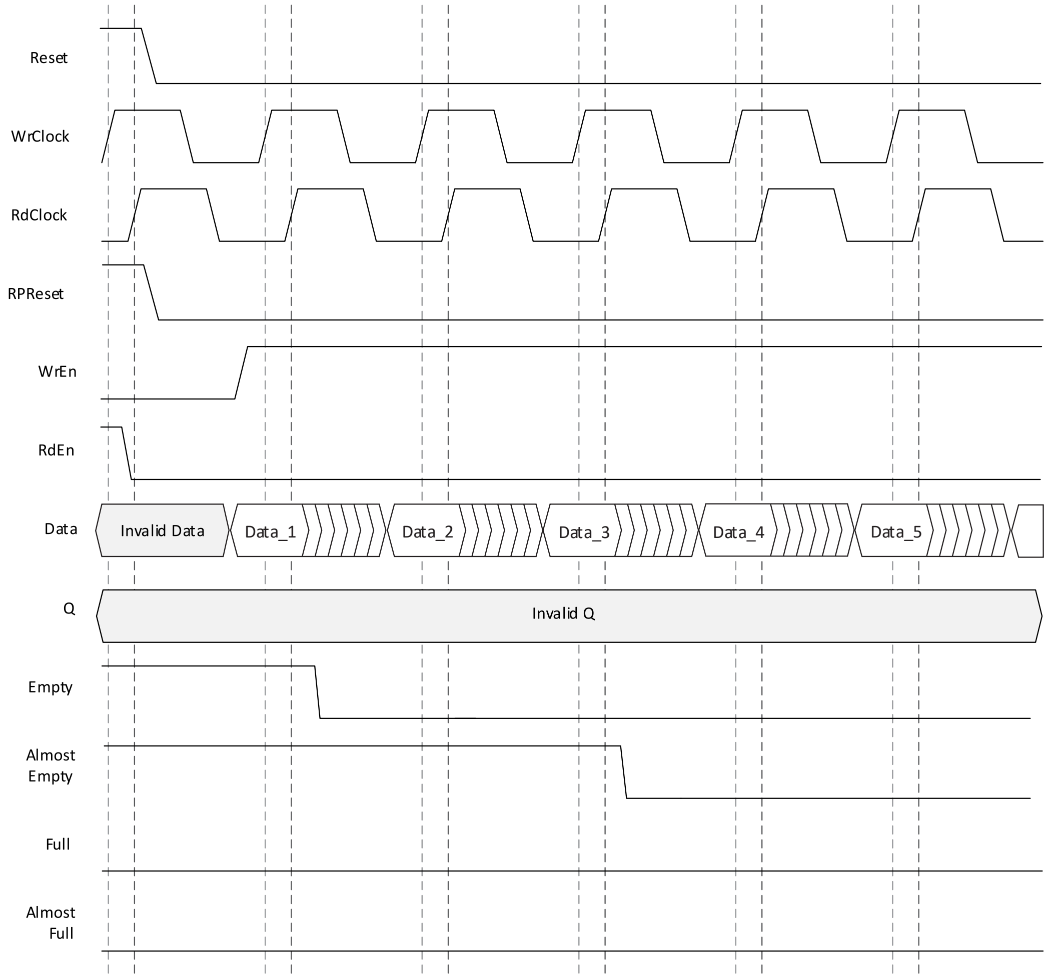


**Figure 6.6. FIFO_DC with Output Registers, Start of Data Write Cycle**

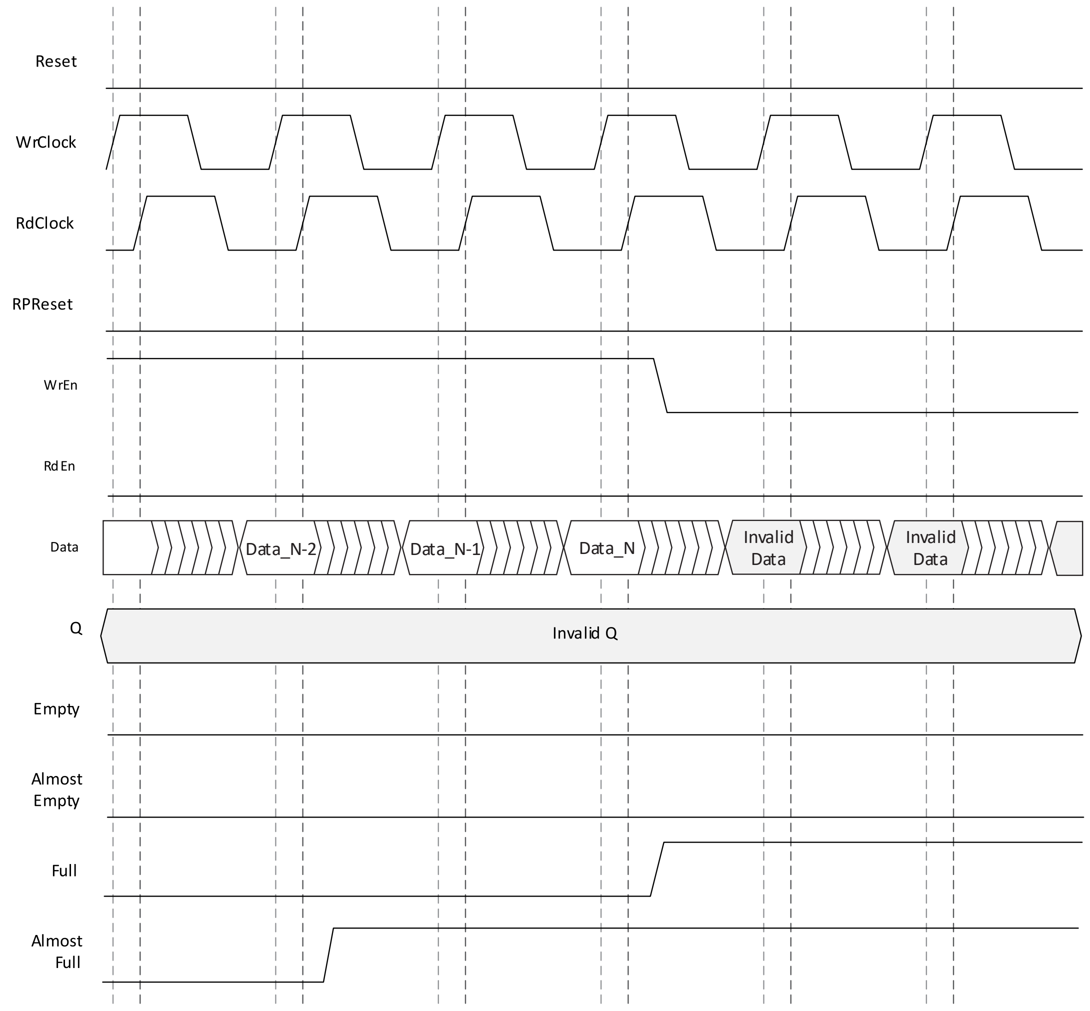


**Figure 6.7. FIFO_DC with Output Registers, End of Data Write Cycle**

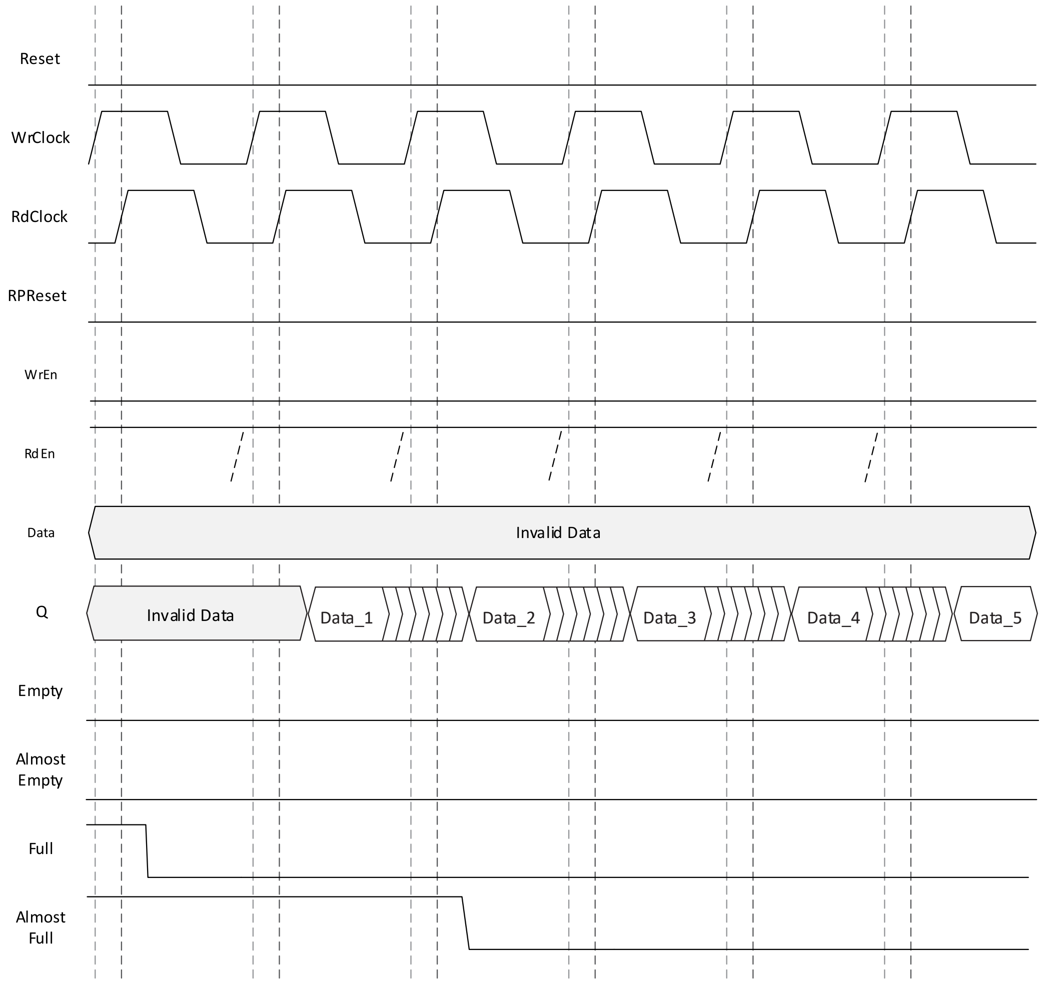


**Figure 6.8. FIFO_DC with Output Registers, Start of Data Read Cycle**

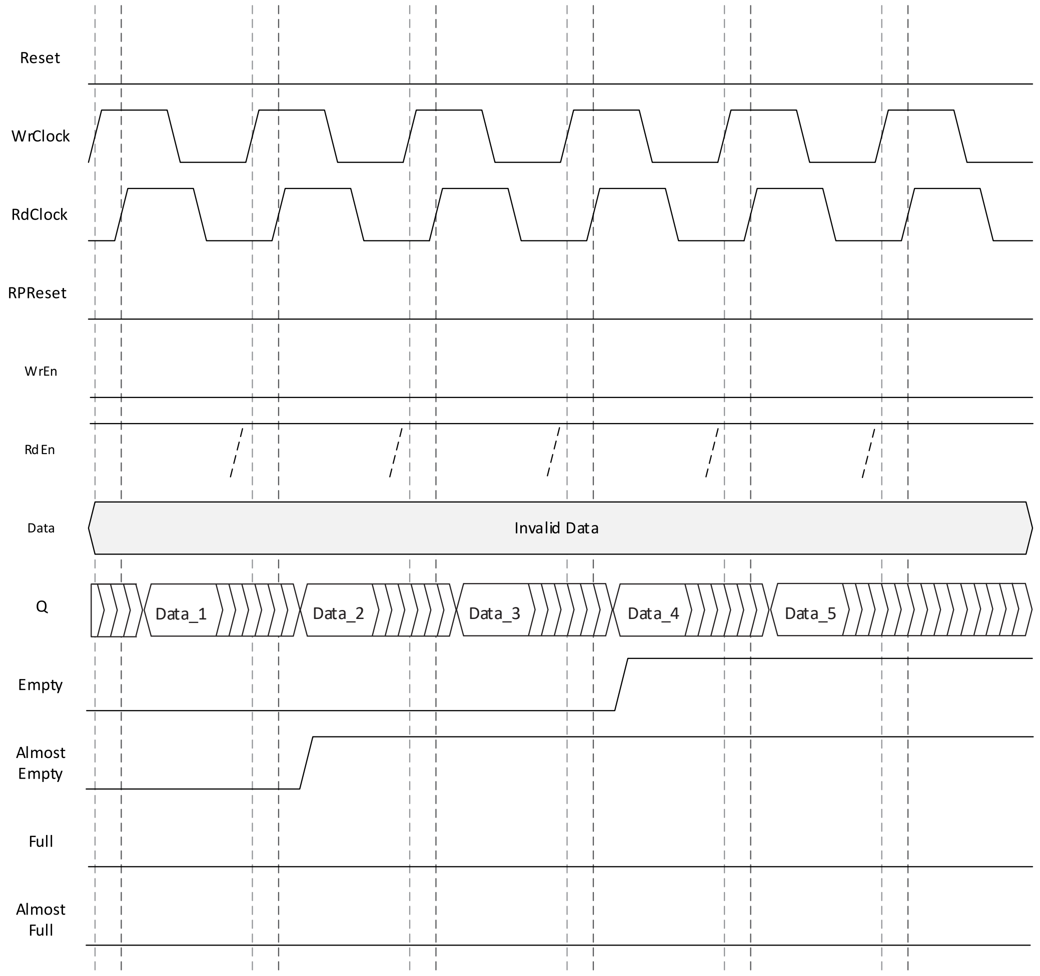


**Figure 6.9. FIFO_DC with Output Registers, End of Data Read Cycle**

If you select the option to enable output register with RdEn, data out is still delayed 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.

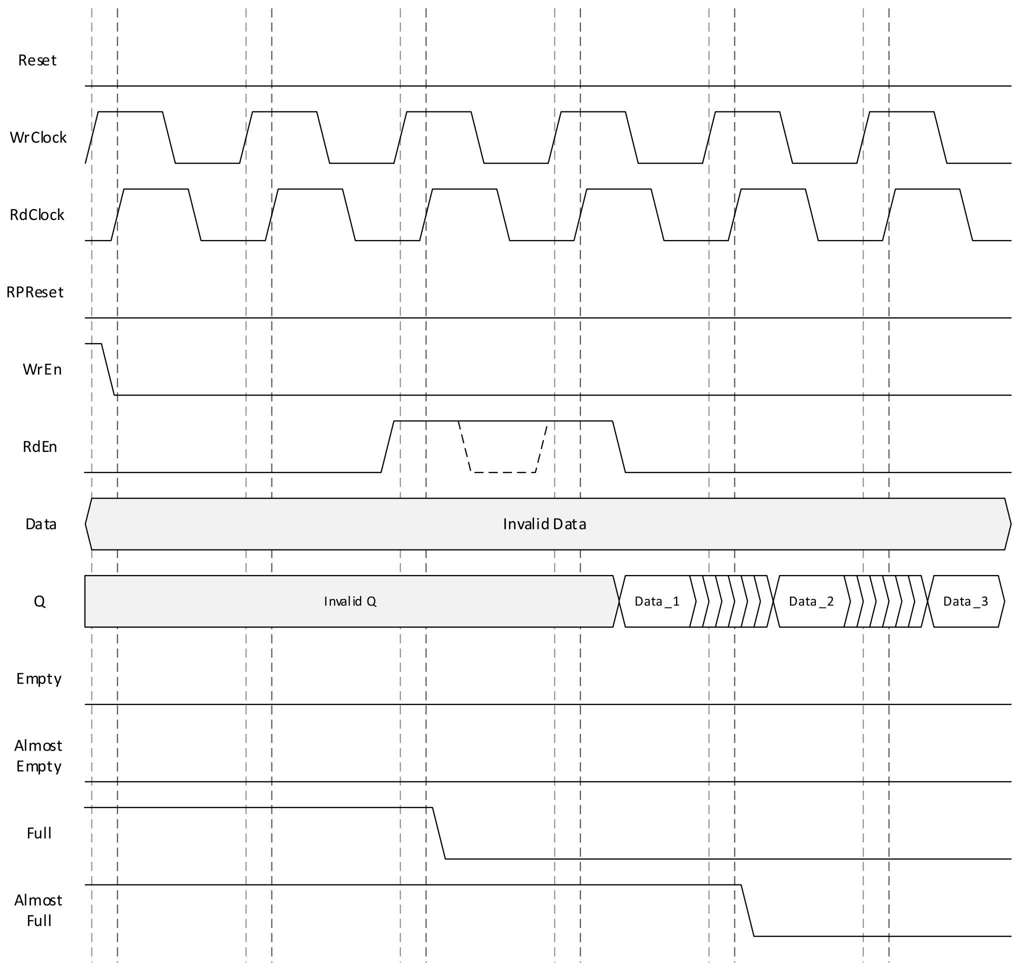


**Figure 6.10. FIFO_DC with Output Registers and RdEn on Output Registers**

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

If you reverse the number of bits on DataIn and DataOut, then for every written word, two words are read out. To completely read the FIFO_DC, you 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 IP Catalog.

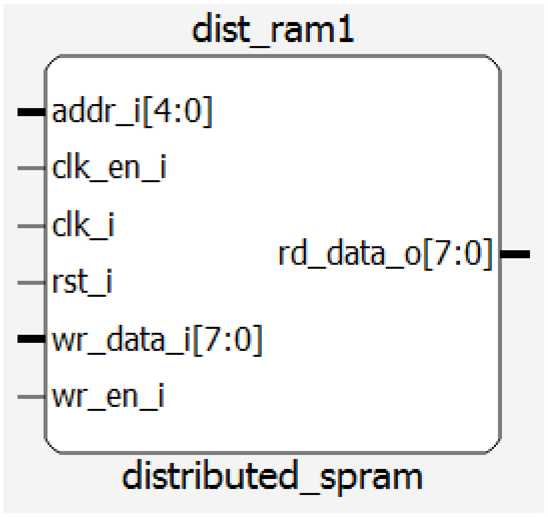


**Figure 7.1. Distributed Single-Port RAM Module Generated by IP Catalog**

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 IP Catalog when you wants to enable the output registers in the IP Catalog configuration.

Figure 7.2 provides the primitive that can be instantiated for the Single Port Distributed RAM. The 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 Radiant 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.

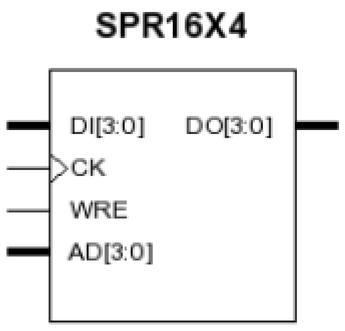


**Figure 7.2. Single Port Distributed RAM Primitive for Nexus Platform Devices**

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

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

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Clock |
| rst_i | Input | 1 | Reset |
| clk_en_i | Input | 1 | Clock Enable |
| we_i | Input | 1 | Read Enable |
| wr_data_i | Input | Data Width | Write Data |
| addr_i | Input | Address Width | Address |
| rd_data_o | Output | Data Width | Read Data |

The software attributes for the Distributed SPRAM are included in Table 7.2.

**Table 7.2. Distributed_SPRAM Attributes for Nexus Platform 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> | 32 |
| Data Width | Data word width of the read and write port | 1 – 256 | 8 |
| Enable Output Register | Data Out port (Q) can be registered or not using this selection. | TRUE, FALSE | TRUE |
| Memory Initialization | Allows you to initialize their memories to all 1s, 0s or providing a custom initialization by providing a memory file. | none, 0s, 1s, Memory file | none |
| Memory File | When Memory file is selected, you can browse to the mem file for custom initialization of RAM. | — | — |
| Memory File Format | This option allows you to select if the memory file is formatted as Binary, Hex. | binary, hex | binary |

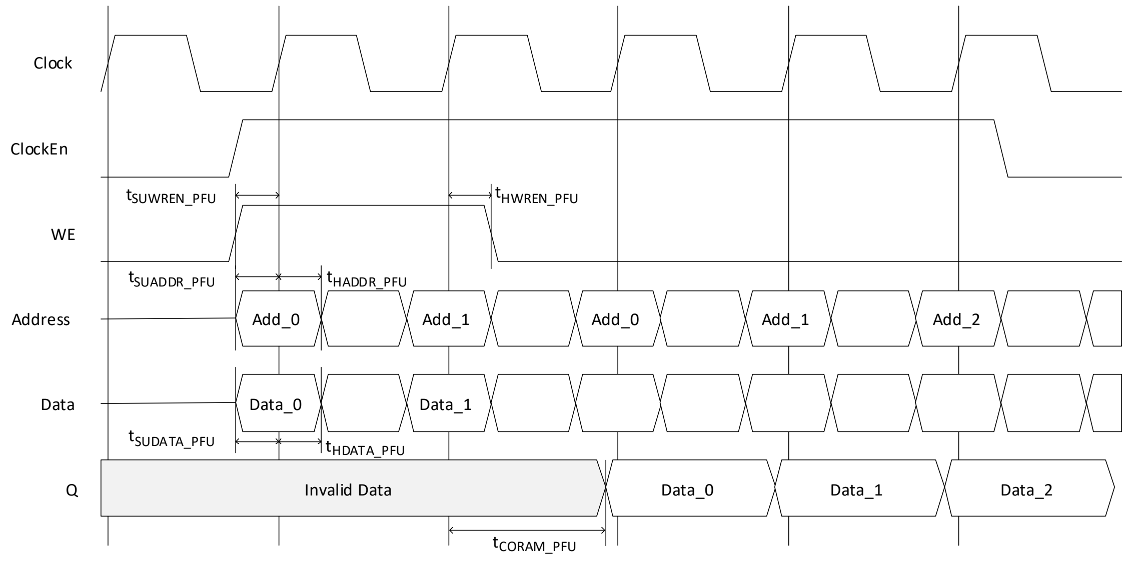


**Figure 7.3. PFU-Based Distributed Single Port RAM Timing Waveform – without Output Registers**

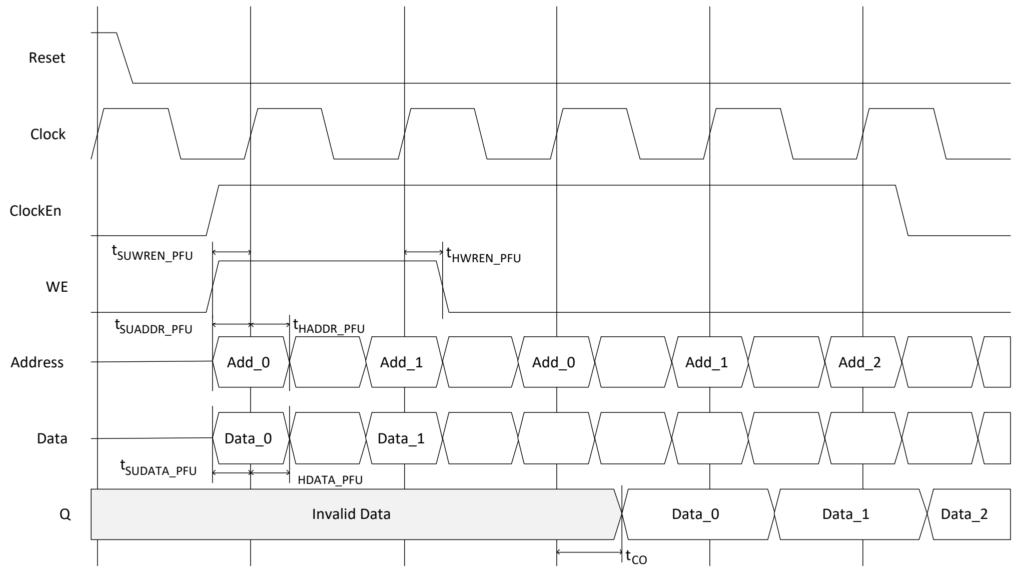


**Figure 7.4. PFU-Based Distributed Single Port RAM Timing Waveform – with Output Registers**

# 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 IP Catalog.

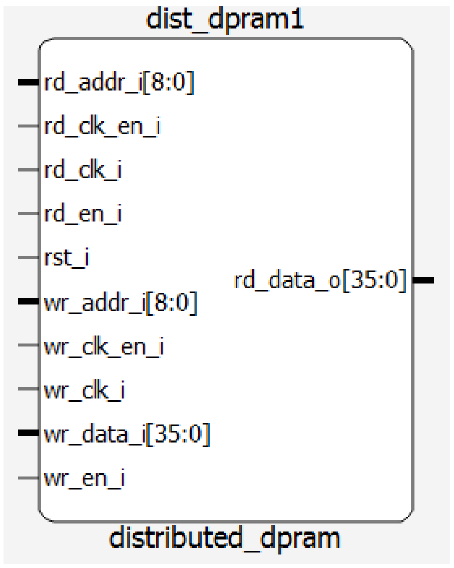


**Figure 8.1. Distributed Dual-Port RAM Module Generated by IP Catalog**

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 and Read Clock Enable are not available in the hardware primitive. These are generated by IP Catalog when you want to enable the output registers in the IP Catalog configuration.

Figure 8.2 provides the primitive that can be instantiated for the Dual Port Distributed RAM. The primitive name is DPR16X4 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 Radiant 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.

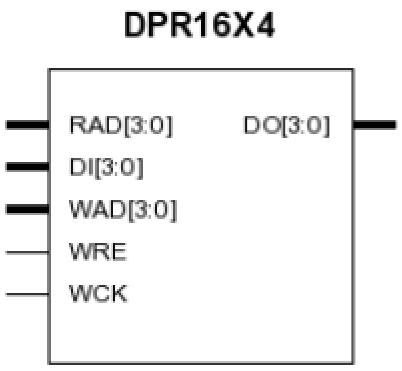


**Figure 8.2. Dual Port Distributed RAM Primitive for Nexus Platform Devices**

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

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

| Port Name | Direction | Width | Description |
|---|---|---|---|
| wr_clk_i | Input | 1 | Write Clock |
| rd_clk_i | Input | 1 | Read Clock |
| rst_i | Input | 1 | Reset |
| wr_clk_en_i | Input | 1 | Write Clock Enable |
| rd_clk_en_i | Input | 1 | Read Clock Enable |
| rd_en_i | Input | 1 | Read Enable |
| wr_en_i | Input | 1 | Write Enable |
| wr_data_i | Input | Data Width | Write Data |
| wr_addr_i | Input | Address Width | Read Address |
| rd_addr_i | Input | Address Width | Read Address |
| rd_data_o | Output | Data Width | Read Data |

The software attributes for the Distributed DPRAM are described in Table 8.2.

**Table 8.2. Distributed_DPRAM Attributes for Nexus Platform 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> | 32 |
| Data Width | Data word width of the Read and write port | 1 – 256 | 8 |
| Enable Output Register | Data Out port (Q) can be registered or not using this selection. | TRUE, FALSE | TRUE |
| Memory Initialization | This option allows you to initialize memories to all 1s, 0s, or provide a custom initialization by providing a memory file. | none, 0s, 1s, Memory file | none |
| Memory File | When Memory file is selected, you can browse to the mem file for custom initialization of RAM. | — | — |
| Memory File Format | This option allows you to select if the memory file is formatted as Binary, Hex. | binary, hex | binary |

You 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.

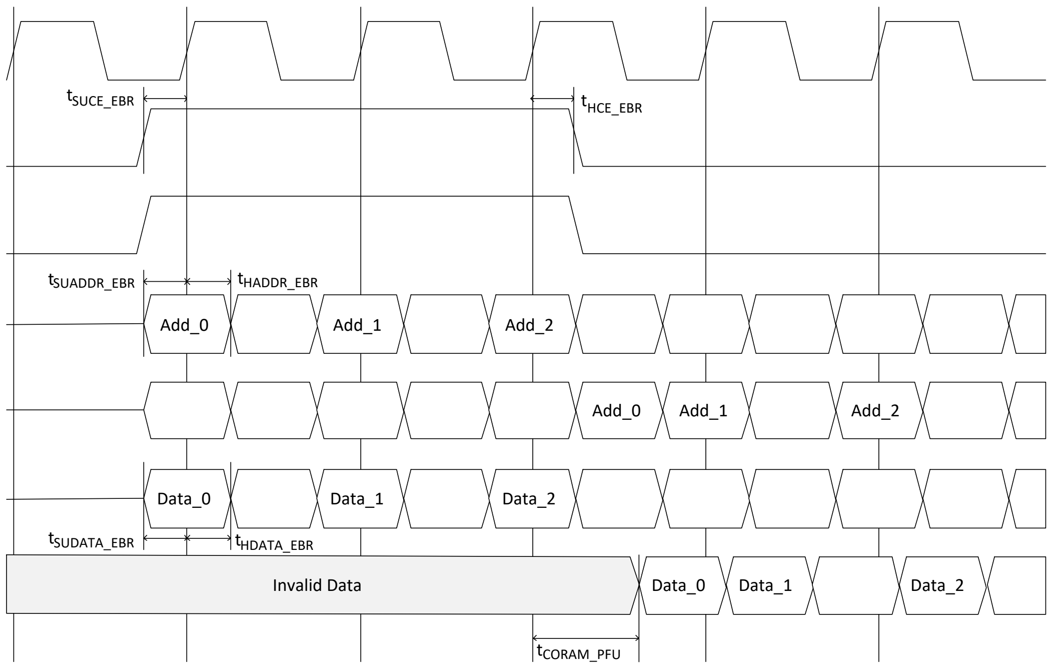


**Figure 8.3. PFU-Based Distributed Dual Port RAM Timing Waveform – without Output Registers**

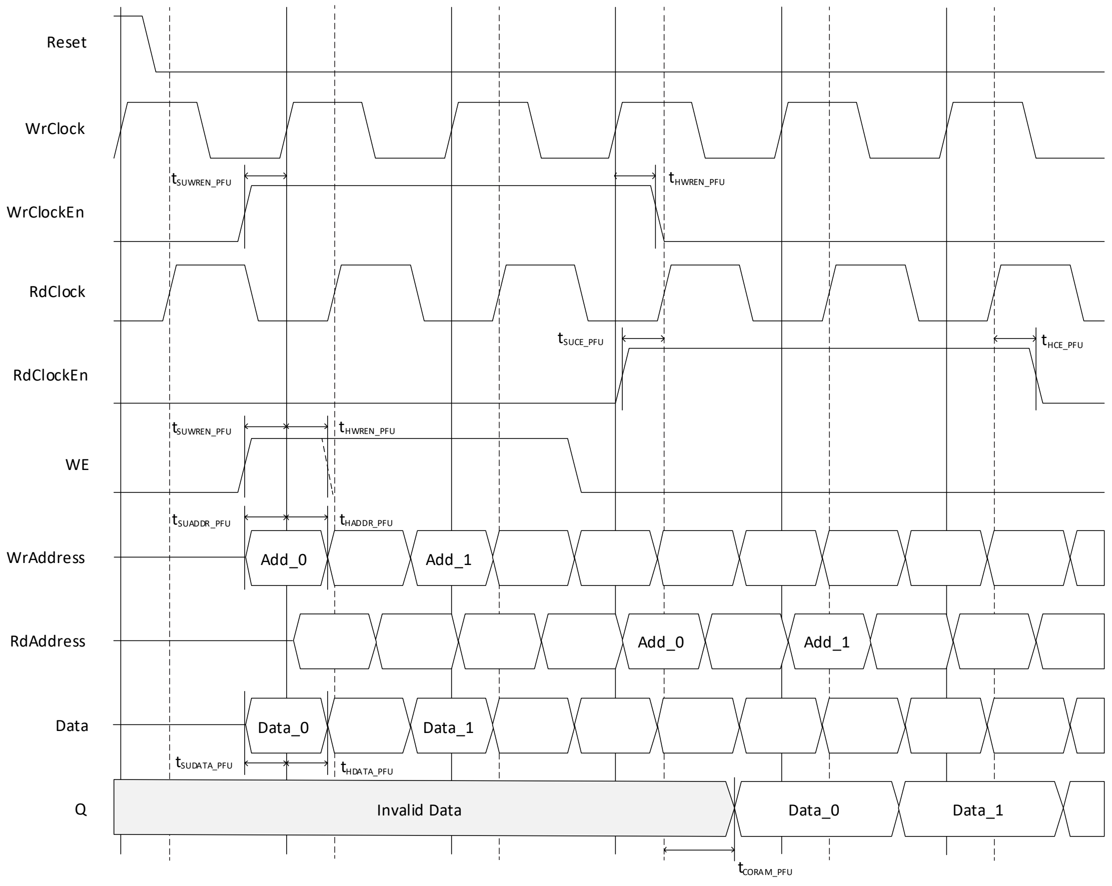


**Figure 8.4. PFU-Based Distributed Dual Port RAM Timing Waveform – without Output Registers**

# 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 IP Catalog.

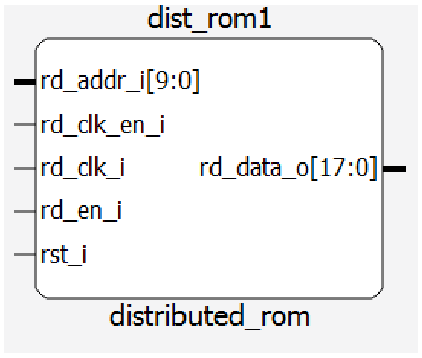


**Figure 9.1. Distributed ROM Generated by IP Catalog**

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 and Out Clock Enable are not available in the hardware primitive. These are generated by IP Catalog when you want to enable the output registers in the IP Catalog configuration.

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 IP Catalog and for the primitive.

**Table 9.1. PFU-Based Distributed ROM Port Definitions**

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Clock |
| rst_i | Input | 1 | Reset |
| clk_en_i | Input | 1 | Clock Enable |
| addr_i | Input | Address Width | Address |
| rd_data_o | Output | Data Width | Read Data |

The software attributes for the Distributed ROM are included in Table 9.2.

**Table 9.2. Distributed_ROM Attributes for Nexus Platform 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> | 32 |
| Data Width | Data word width of the Read and write port | 1 – 256 | 8 |
| Enable Output Register | Data Out port (Q) can be registered or not using this selection. | True, False | True |
| Memory File | When Memory file is selected, you can browse to the mem file for custom initialization of RAM. | — | — |
| Memory File Format | This option allows you to select if the memory file is formatted as Binary, Hex. | binary, hex | binary |

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

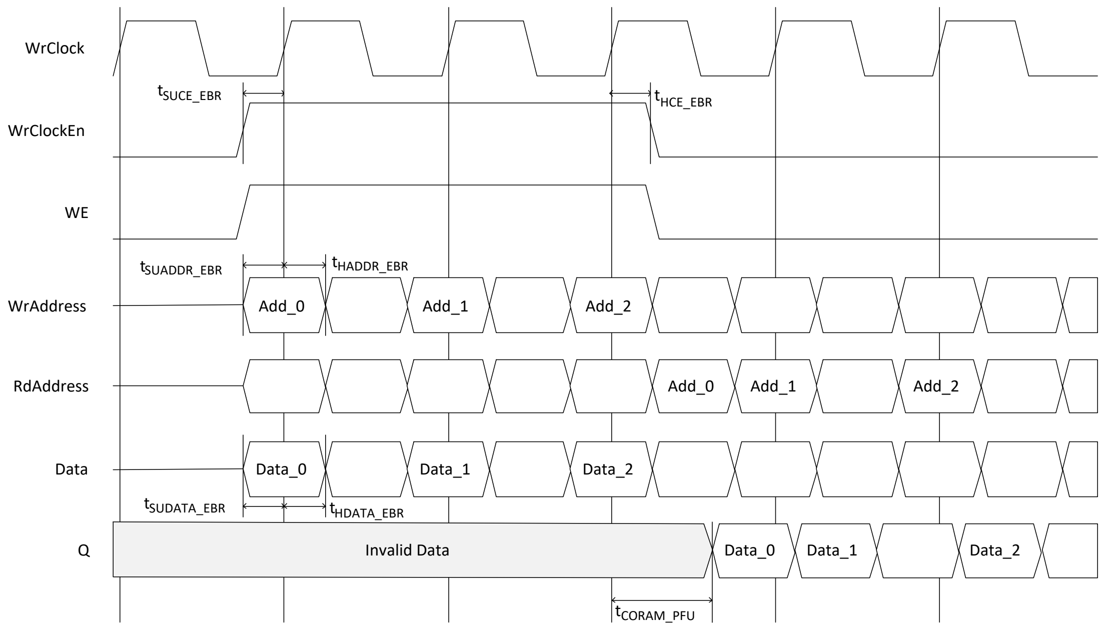


**Figure 9.2. PFU-Based Distributed Dual Port RAM Timing Waveform – without Output Registers**

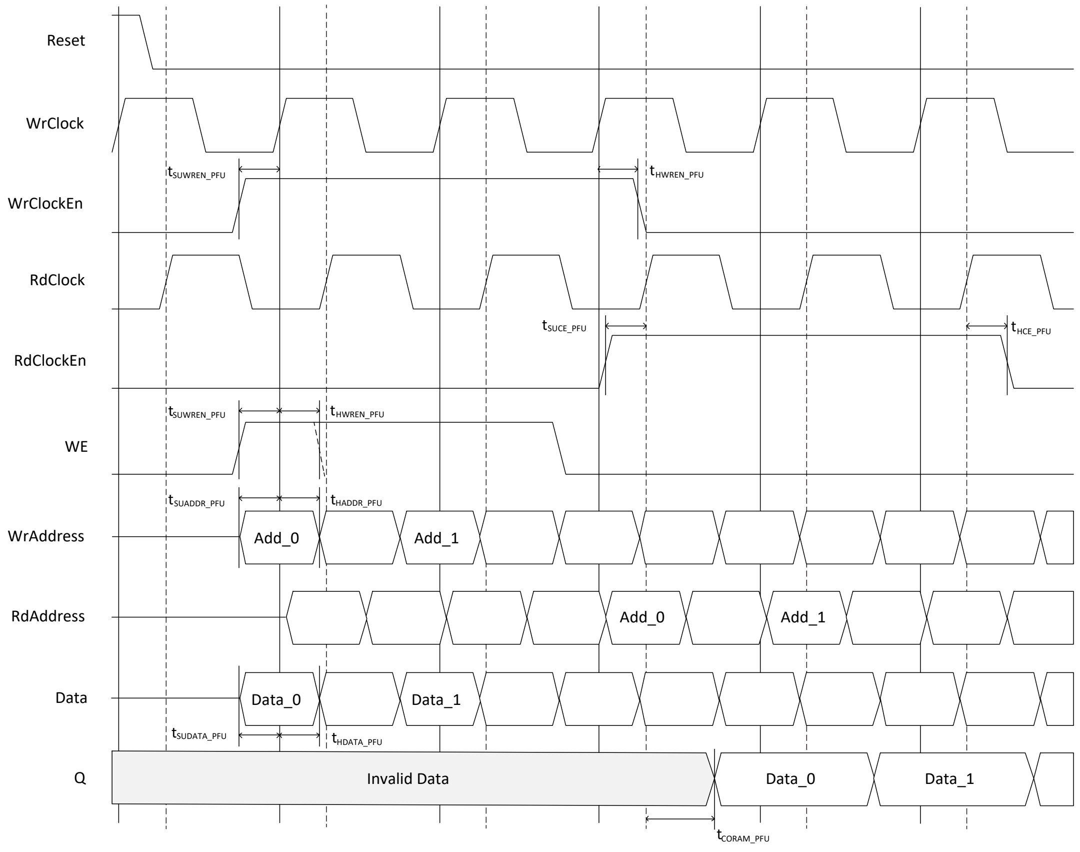


**Figure 9.3. PFU-Based Distributed Dual Port RAM Timing Waveform – with Output Registers**

# 10. Large RAM (LRAM)

The Lattice Semiconductor Large Random-Access Memory (LRAM) IP Core is designed to work as Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM, and ROM memories. It is meant to function as additional memory resources beyond what is available in the EBR and PFU. The following sections cover each of the LRAM configuration modes.

## 10.1.Single Port LRAM

In the Single-Port Mode, only one port is used to write and read. Input can be configured as register in and output can be configured as register out. The SRAM enclosed in the Large RAM IP is synchronous. IP Catalog generates the memory module, as shown in Figure 10.1.

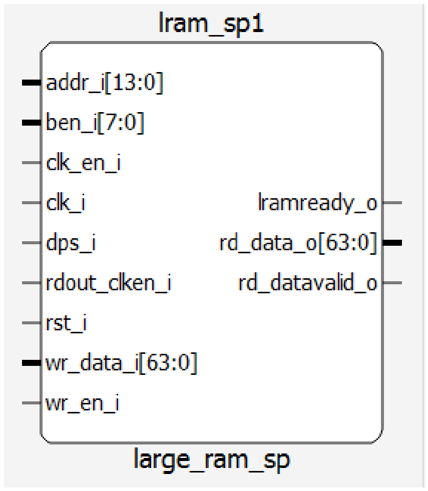


**Figure 10.1. Single Port Large RAM Generated by IP Catalog**

Table 10.1 lists the ports and definitions for Single-Port mode of the Large RAM primitive.

**Table 10.1. Single-Port Mode Signals**

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Clock for Port A |
| wr_data_i[DWA-1:0] | Input | Data Width | Input Data Port A (1 – 32 bits) |
| addr_i[AWA-1:0] | Input | Address Width | Port A Address (10 – 16 bits) |
| wr_en_i | Input | 1 | Port A Write Enable |
| clk_en_i | Input | 1 | Port A Clock Enable |
| ben_i[n-1:0] | Input | 4 | Port A Byte Enable (n takes values from 1 to 4)<br>Optional signal<br>For each bit position:<br>*0* – The corresponding byte should be written.<br>*1* – The corresponding byte should not be written. |
| rst_i | Input | 1 | Port A Logic Reset |
| rdout_clken_i | Input | 1 | Port A Output Register Clock Enable |
| rd_datavalid_o | Output | 1 | Output Enable Port A |
| rd_data_o[DWA-1:0] | Output | Data Width | Output Data Port A |
| dps_i | Input | 1 | Dynamic Power Select |
| lramready_o | Output | 1 | Large RAM IP ready indicator |
| errdeca_o[1:0] | Output | 2 | Error Correction indicator |
| errdet_o | Output | 1 | Large RAM IP error status |

Table 10.2 shows the attributes for the Single-Port mode of the Large RAM primitive.

**Table 10.2. Attributes Summary for Single-Port Mode**

| Attribute | Description | Selectable Values | Default |
|---|---|---|---|
| LRAM Type | Type of memory | Single Port; True Dual Port; Pseudo Dual Port; ROM | Single Port |
| Clock Polarity | Select polarity of data clock | Active High; Active Low | Active High |
| Internal Clock Delay Control Source | Choose internal or CIB control of clock delay control | Internal Clock Delay Value; Input Port: cib_clkdly_ctrl_i | Internal Clock Delay Value |
| Internal Clock Delay Value | Choose clock delay code | 00; 01; 10; 11 | 00 |
| Preserve Array Enable | Keeps array size from being modified | Unchecked; Checked | Unchecked |
| Global Reset Enable | Allows global reset to affect memory | Unchecked; Checked | Unchecked |
| Provide Byte Enables | Allows you to select Byte Enable options | Unchecked; Checked | Unchecked |
| Unaligned Read Enable | Allows asynchronous reads | Unchecked; Checked | Unchecked |
| Enable ECC | Allows you to enable Error Correction Codes. | Unchecked; Checked | Unchecked |
| Reset Assertion | Selection for the reset assertion to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| Reset Release | Selection for the reset release to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| INIT Bus Write ID | ID for writing initialization data | 0 – 2047 | 0 |
| Memory Initialization | This option allows you to initialize memories to all 1s, 0s, or provide a custom initialization by providing a memory file. | None; Initialize to all 0s; Initialize to all 1s; Memory File | None |
| Memory File | When memory file is selected, you can browse to the mem file for custom initialization of RAM. | Button; File browser | Unselected |
| Memory File Format | This option allows you to select if the memory file is formatted as Binary, Hex, or address Hex. | Binary; Hex; Addressed Hex | Binary |
| Data Width | Data word width of read and write port | 1 – 32 | 32 |
| Address Width | Address depth of read and write port | 8 – 16 | 14 |
| Write Mode | Selectable write mode timing | Normal; Write Through; Read Before Write | Normal |
| Write Enable Polarity | Select enable polarity of WE | Active High; Active Low | Active High |
| Clock Enable Polarity | Select enable polarity for CE | Active High; Active Low | Active High |
| Reset Polarity | Select polarity of reset | Active High; Active Low | Active High |
| Input Register | Data in port (wr_data_i) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |
| Output Register | Data out port (rd_data_o) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |

The waveforms on the following figures show the internal timing for the single port LRAM with the various input and output register enable permutations.

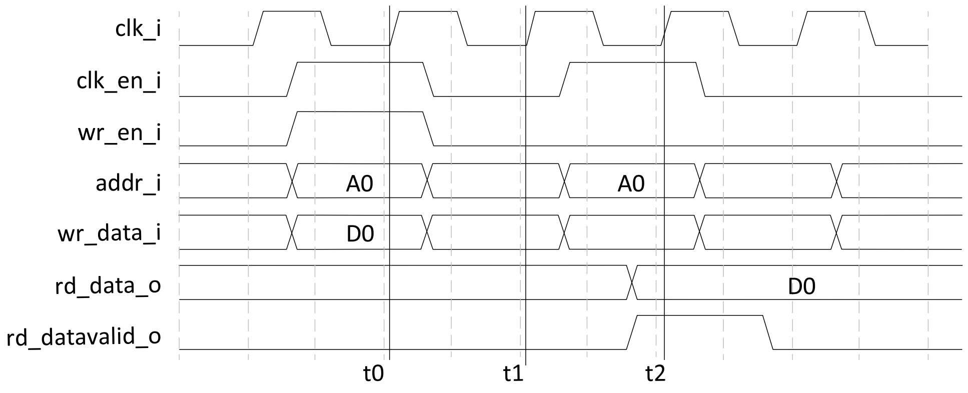


**Figure 10.2. Single-Port Mode Timing Diagram (Both Input and Output Registers Disabled)**

As shown in Figure 10.2, data flow is as follows:

1. addr_i and wr_data_i are clocked in the SRAM at t0.
2. When you read the data, set clk_en_i and wr_en_i port values after t1.
3. You get the read back data at t2.

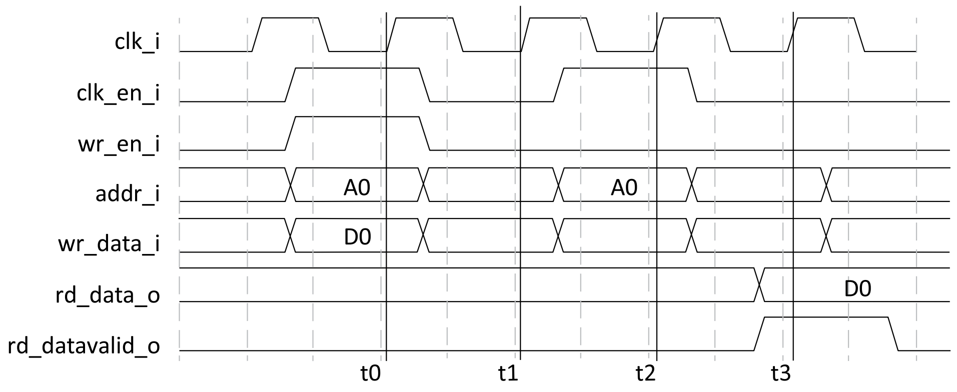


**Figure 10.3. Single-Port Mode Timing Diagram (Either Input Register Enabled/Output Register Disabled or Input Register Disabled/Output Register Enabled)**

As shown in Figure 10.3, data flow is as follows:

1. For both cases, first, addr_i and wr_data_i are clocked in the SRAM at t0 in user's view;
2. For the Input Register Enabled/Output Register Disabled case, Large RAM registers input signals clk_en_i, wr_en_i, addr_i, wr_data_i with input registers and those signals are clocked in the SRAM at t1 in user's view;
3. When you read the data, Large RAM IP registers the input signals after t2 and connects those input signals to SRAM input ports;
4. SRAM clocks in input signals; output data D0 gets ready;
5. You get the read back data at t3.
6. For the Input Register Disabled/Output Register Enabled case, when you read the data, set clk_en_i and wr_en_i port values after t1;
7. Large RAM connects those signals to SRAM; SRAM clocks in addr_i and wr_data_i, data D0 gets ready;
8. Large RAM registers the output data with output register after t2 and connects it to output port rd_data_o;
9. You get the read back data at t3.

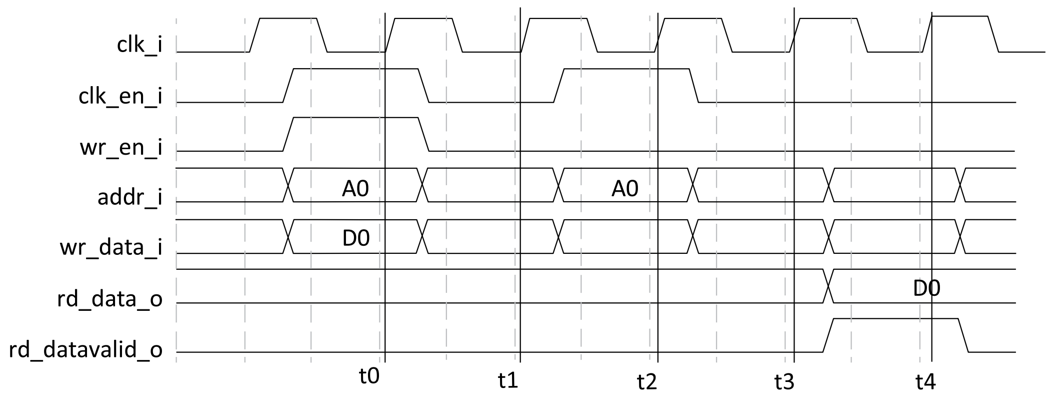


**Figure 10.4. Single-Port Mode Timing Diagram (Both Input and Output Registers Enabled)**

As shown in Figure 10.4, data flow is as follows:

1. addr_i and wr_data_i are clocked in the SRAM at t0 in user's view.
2. Large RAM registers the input signals clk_en_i, wr_en_i, addr_i, and wr_data_i with input registers; those signals are clocked in the SRAM at t1 in user's view.
3. When you read data, Large RAM IP registers the input signals after the positive edge of t2 and connects them to SRAM input ports.
4. SRAM clocks them in, outputs data gets ready.
5. The large RAM registers the output data with output register after t3 and connects it to output port rd_data_o.
6. You get the read back data at t4.

## 10.2. True Dual Port LRAM

In True Dual-Port Mode, both ports can be used to write and read. Input can be configured as register in and output can be configured as register out. Dual-Port Mode is implemented from the Single-Port SRAM Model. To accommodate the requests from both ports at the same time, the enclosed Single-Port RAM runs the clock with doubled frequency. In the Dual-Port Mode, if reading and writing operations come at one CIB clock cycle, those operations are mapped to the Single-Port SRAM model.

The SRAM enclosed in the Large RAM IP is synchronous. IP Catalog generates the memory module, as shown in Figure 10.5.

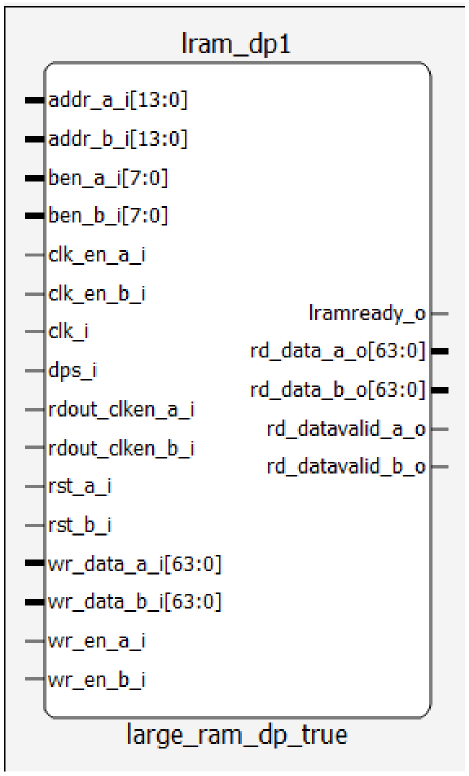


**Figure 10.5. True Dual-Port Large RAM Generated by IP Catalog**

Table 10.3 lists the ports and definitions for True Dual-Port mode of the Large RAM primitive.

**Table 10.3. True Dual-Port Mode Signal**

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Clock for Port A |
| wr_data_a_i[DWA-1:0] | Input | Data Width | Input Data Port A (1 – 32 bits) |
| addr_a_i[AWA-1:0] | Input | Address Width | Port A Address (10 – 16 bits) |
| wr_en_a_i | Input | 1 | Port A Write Enable |
| clk_en_a_i | Input | 1 | Port A Clock Enable |
| ben_a_i[n-1:0] | Input | 4 | Port A Byte Enable (n takes values from 1 to 4)<br>Optional signal<br>For each bit position:<br>*0* – The corresponding byte should be written.<br>*1* – The corresponding byte should not be written. |
| rsta_i | Input | 1 | Port A Logic Reset |

| Port Name | Direction | Width | Description |
|---|---|---|---|
| rdout_clken_a_i | Input | 1 | Port A Output Register Clock Enable |
| rd_datavalid_a_o | Output | 1 | Output Enable Port A |
| rd_data_a_o[DWA-1:0] | Output | Data Width | Output Data Port A |
| dps_i | Input | 1 | Dynamic Power Select |
| wr_data_b_i[DWB-1:0] | Input | Data Width | Input Data Port B (1 – 32 bits) |
| addr_b_i[AWB-1:0] | Input | Address Width | Port B Address (10 – 16 bits) |
| wr_en_b_i | Input | 1 | Port B Write Enable |
| clk_en_b_i | Input | 1 | Port B Clock Enable |
| ben_b_i[3:0] | Input | 4 | Port B Byte Enable (n takes values from 1 to 4). Optional signal<br>For each bit position:<br>*0* – The corresponding byte should be written.<br>*1* – The corresponding byte should not be written. |
| rstb_i | Input | 1 | Port B Logic Reset |
| rdout_clken_b_i | Input | 1 | Port B Output Register Clock Enable |
| rd_datavalid_b_o | Output | 1 | Output Enable Port B |
| rd_data_b_o[DWB-1:0] | Output | Data Width | Output Data Port B |
| lramready_o | Output | 1 | Large RAM IP ready indicator |
| errdeca_o[1:0] | Output | 2 | Error Correction indicator |
| errdecb_o[1:0] | Output | 2 | Error Correction indicator |
| errdet_o | Output | 1 | Large RAM IP error status |

Table 10.4 shows the attributes for the True Dual-Port mode of the Large RAM primitive.

**Table 10.4. Attributes Summary for True Dual-Port Mode**

| Attribute | Description | Selectable Values | Default |
|---|---|---|---|
| LRAM Type | Type of memory | Single Port; True Dual Port; Pseudo Dual Port; ROM | Single Port (you choose True Dual ) |
| Clock Polarity | Select polarity of data clock | Active High; Active Low | Active High |
| Internal Clock Delay Control Source | Choose internal or CIB control of clock delay control | Internal Clock Delay Value; Input Port: cib_clkdly_ctrl_i | Internal Clock Delay Value |
| Internal Clock Delay Value | Choose clock delay code | 00; 01; 10; 11 | 00 |
| Preserve Array Enable | Keeps array size from being modified | Unchecked; Checked | Unchecked |
| Global Reset Enable | Allows global reset to affect memory | Unchecked; Checked | Unchecked |
| Provide Byte Enables | Allows you to select Byte Enable options | Unchecked; Checked | Unchecked |
| Unaligned Read Enable | Allows asynchronous reads | Unchecked; Checked | Unchecked |
| Enable ECC | Allows you to enable Error Correction Codes. | Unchecked; Checked | Unchecked |
| Reset Assertion | Selection for the reset assertion to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| Reset Release | Selection for the reset release to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| INIT Bus Write ID | ID for writing initialization data | 0 – 2047 | 0 |
| Memory Initialization | Allows you to initialize memories to all 1s, 0s, or provide a custom initialization by providing a memory file. | None; Initialize to all 0s; Initialize to all 1s; Memory File | None |
| Memory File | When memory file is selected, you can browse to the mem file for custom initialization of RAM. | Button; File browser | Unselected |
| Memory File Format | This option allows you to select if the memory file is formatted as Binary, Hex, or address Hex. | Binary; Hex; Addressed Hex | Binary |
| Data Width A | Data word width of read and write port | 1 – 32 | 32 |
| Address Width A | Address depth of read and write port | 8 – 16 | 14 |
| Write Mode A | Selectable write mode timing | Normal; Write Through; | Normal |
| Write Enable Polarity A | Select enable polarity of WE | Active High; Active Low | Active High |
| Clock Enable Polarity A | Select enable polarity for CE | Active High; Active Low | Active High |
| Reset Polarity A | Select polarity of reset | Active High; Active Low | Active High |
| Input Register A | Data in port (wr_data_a_i) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |
| Output Register A | Data out port (rd_data_a_o) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |
| Data Width B | Data word width of read and write port | 1 – 32 | 32 |
| Address Width B | Address depth of read and write port | 8 – 16 | 14 |
| Write Mode B | Selectable write mode timing | Normal; Write Through; | Normal |
| Write Enable Polarity B | Select enable polarity of WE | Active High; Active Low | Active High |
| Clock Enable Polarity B | Select enable polarity for CE | Active High; Active Low | Active High |
| Reset Polarity B | Select polarity of reset | Active High; Active Low | Active High |
| Input Register B | Data in port (wr_data_b_i) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |
| Output Register B | Data out port (rd_data_b_o) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |

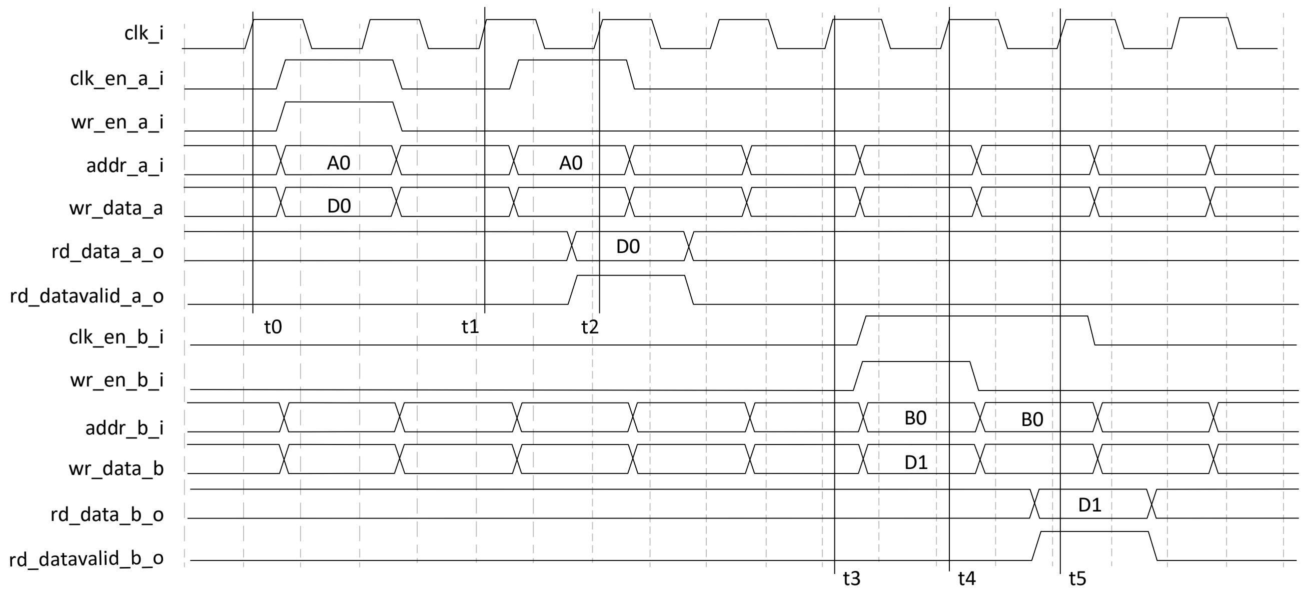


**Figure 10.6. Dual-Port Mode Timing Diagram with Port A and Port B Working in Different Cycles (Both Input and Output Registers Disabled for Both Ports)**

As shown in Figure 10.6, data flow is as follows:

For Port A:

1. You prepare the data at t0;
2. Large RAM clocks in clk_en_a_i, wr_en_a_i, addr_a_i, and wr_data_a_i to SRAM;
3. When you read the data, sets clk_en_a_i and wr_en_a_i port values after t1. Large RAM connects those signals to SRAM;
4. SRAM clocks in addr_a_i and wr_data_a_i, output data gets ready;
5. Large RAM connects it to rd_data_a_o, and you get the read back data at t2.

For Port B:

1. You prepare the data at t3;
2. Large RAM clocks in clk_en_b_i, wr_en_b_i, addr_b_i, and wr_data_b_i to SRAM;
3. When you read the data, set clk_en_b_i and wr_en_b_i port values. Large RAM connects those signals to SRAM at t4.
4. SRAM clocks in addr_b_i and wr_data_b_i, output data gets ready.
5. Large RAM connects it to rd_data_b_o, and you get the read back data at t5.

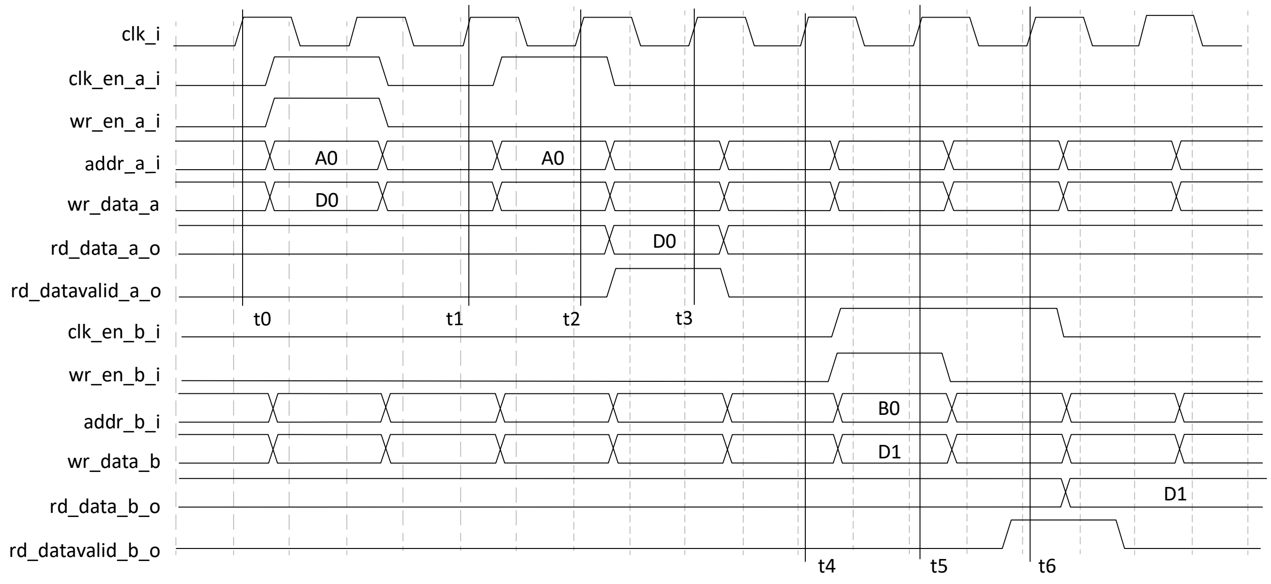


**Figure 10.7. Dual-Port Mode Timing Diagram with Port A and Port B Working in Different Cycles (Both Input and Output Registers Disabled for Both Ports)**

As shown in Figure 10.7, data flow is as follows:

For Port A:

1. You prepare the data at t0;
2. Large RAM clocks in clk_en_a_i, wr_en_a_i, addr_a_i, and wr_data_a_i to SRAM;
3. When you read the data, set clk_en_a_i and wr_en_a_i port values after t1. Large RAM connects those signals to SRAM.
4. SRAM clocks in addr_a_i and wr_data_a_i, output data gets ready.
5. Large RAM registers the output data with output register after t2 and connects it to output port rd_data_a_o.
6. You get the read back data at t3.

For Port B:

1. You prepare the data at t4;
2. Large RAM clocks in clk_en_b_i, wr_en_b_i, addr_b_i, and wr_data_b_i to SRAM at t4;
3. When you read the data, you set clk_en_b_i and wr_en_b_i port values. Large RAM connects those signals to SRAM.
4. SRAM clocks in addr_b_i and wr_data_b_i, output data gets ready after t5.
5. Large RAM registers the output data with output register after t6 and connects it to output port rd_data_b_o.
6. You get the read back data at the next positive edge of the clock.

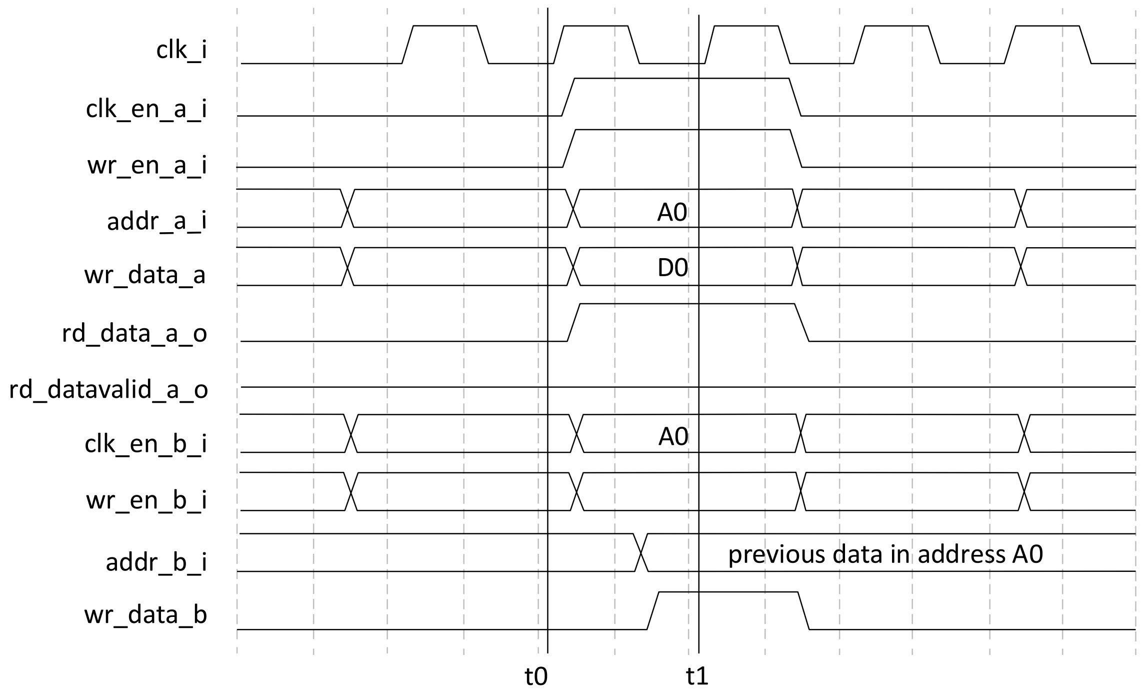


**Figure 10.8. Dual-Port Mode Timing Diagram with Port A and Port B Working in the Same Cycle (Both Input and Output Registers Disabled for Both Ports)**

As shown in Figure 10.8, data flow as follows:

1. Port A writes address A0 and Port B reads address A0 at the same clock cycle;
2. At t0, Port B address is clocked into SRAM, output data is ready after t0;
3. At t1, Port A's address and data are clocked into SRAM for writing;
4. You get Port B read back data at t1.

**Note:** When both ports are writing and reading the same address, reading takes precedence over writing in one cycle, and the output of reading operation is previous data in address.

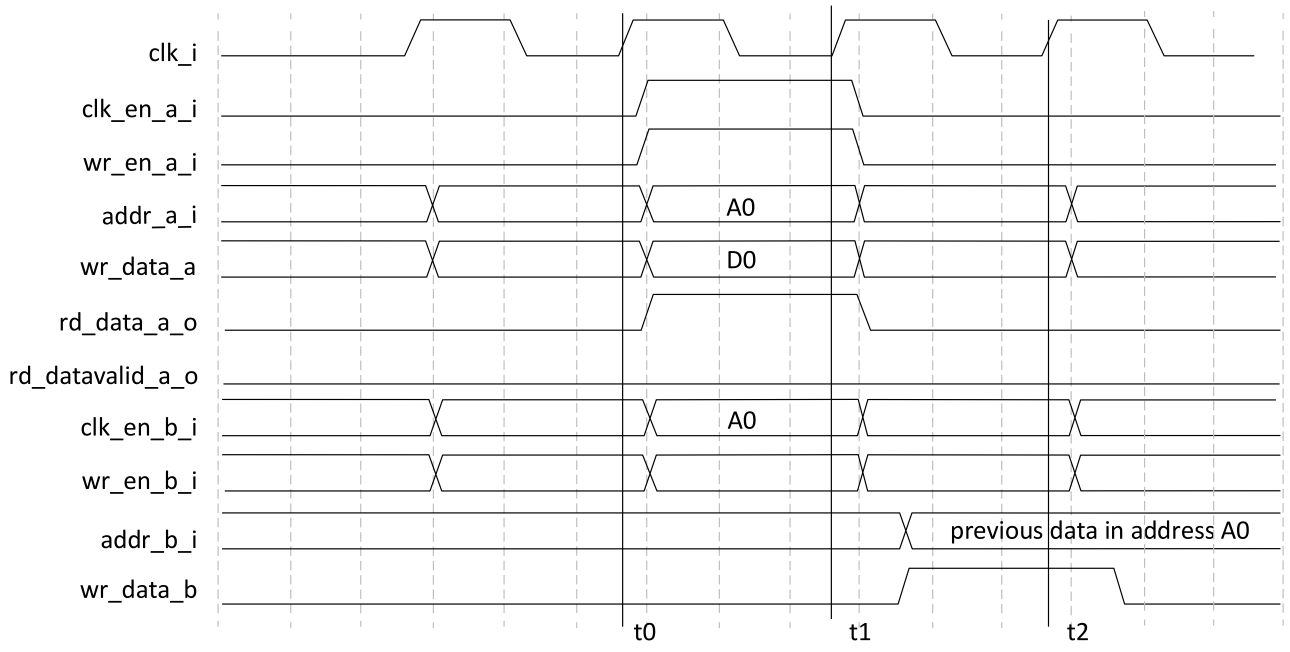


**Figure 10.9. Dual-Port Mode Timing Diagram with Ports A and B Working in the Same Cycle (Input Register Disabled/Output Register Enabled for Both Ports)**

As shown in Figure 10.9, data flow is as follows:

1. Port A writes address A0 and Port B reads address A0 at the same clock cycle;
2. Port B address is clocked into SRAM, and output data is ready after t0;
3. At t1, Port A's address and data are clocked into SRAM for writing;
4. Large RAM registers the output data from Port B with the output register after t1 and connects it to output port rd_data_b_o.
5. You get the Port B read back data at t2.

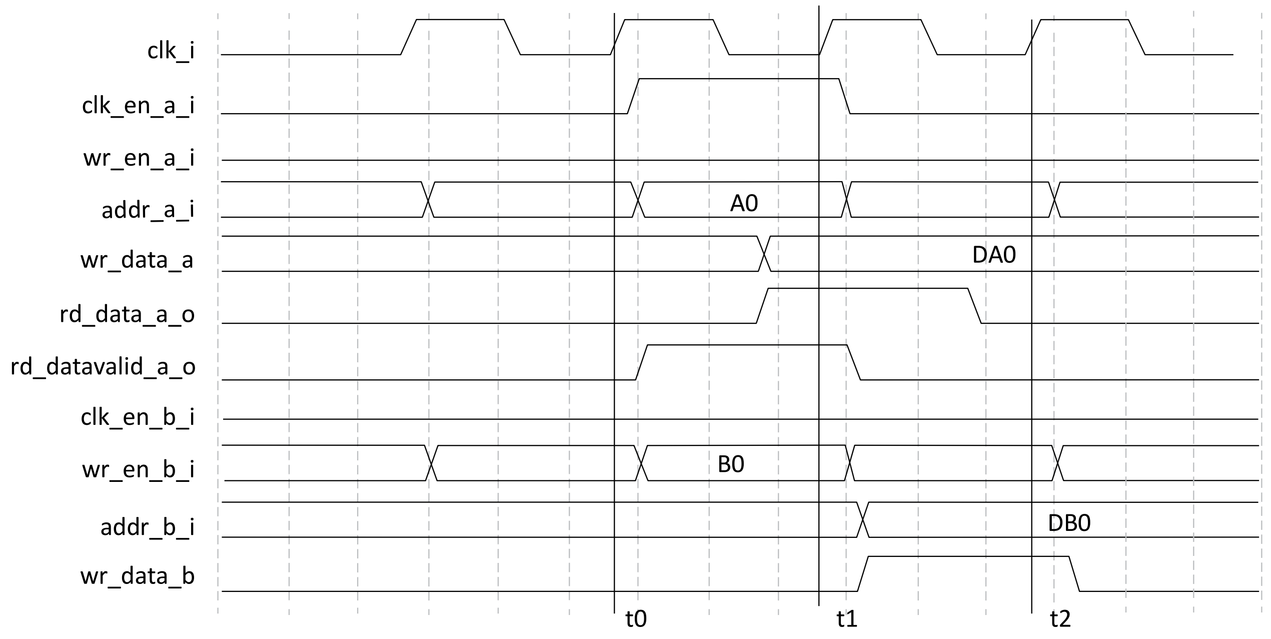


**Figure 10.10. Dual-Port Mode Timing Diagram with Ports A and B Reading in the Same Cycle (Both Input and Output Registers Disabled for Both Ports)**

As shown in Figure 10.10, data flow is as follows:

1. Both Port A and Port B read different address at the same clock cycle;
2. Port A address is clocked into SRAM, and output data DA0 is ready after t0;
3. You get the Port A read back data at t1.
4. Port B address is clocked into SRAM, and output data DB0 is ready after t1;
5. You get the Port B read back data at t2.

**Note:** When reading from both ports in the same cycle but from various addresses, data for port B comes with one clock delay. This is because the LRAM primitive has just one port and both addresses cannot be read without delay.

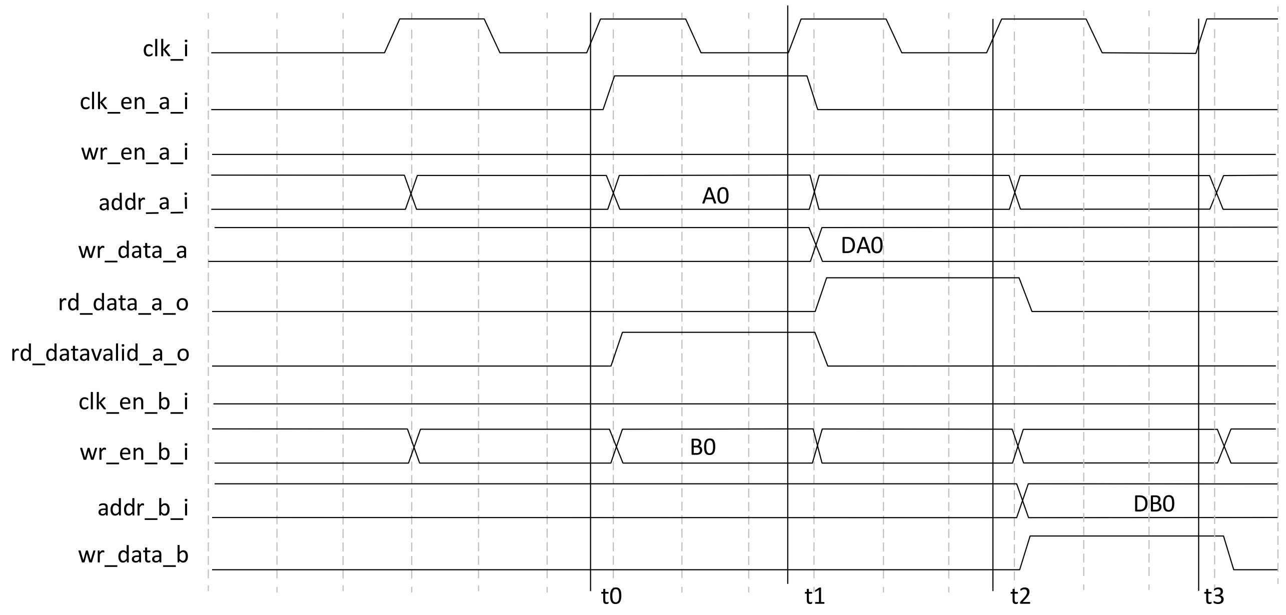


**Figure 10.11. Dual-Port Mode Timing Diagram with Ports A and B Working in the Same Cycle (Input Register Disabled/Output Register Enabled for Both Ports)**

As shown in Figure 10.11, data flow as follows:

1. Both Port A and Port B read different address at the same clock cycle;
2. Port A address is clocked into SRAM, and output data DA0 is ready after t0;
3. Large RAM registers output data DA0 with output register and connects it to output port rd_data_a_o.
4. You get Port A read back data after t1.
5. Port B address is clocked into SRAM, and output data DB0 is ready after t2;
6. Large RAM registers the output data DB0 with output register after t2 and connects it to output port rd_data_b_o.
7. You get Port B read back data at t3.

## 10.3. Pseudo Dual Port LRAM

In Pseudo Dual-Port Mode, Port A works as a writing port and Port B works as a reading port. Input and output register bypass mode is supported in the Single Clock Pseudo Dual-Port Mode. In this mode, both ports are writing to and reading from the same address. The reading takes precedence over writing in one cycle, so the output of reading is the previous data in the address. IP Catalog generates the memory module, as shown in Figure 10.12.

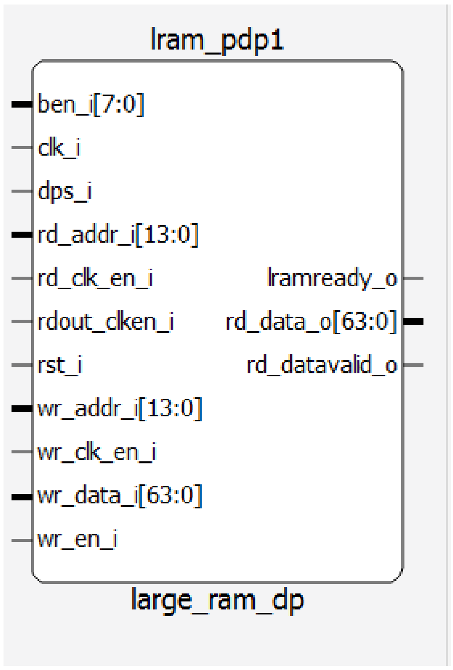


**Figure 10.12. Pseudo Dual Port Large RAM Generated by IP Catalog**

Table 10.5 lists the ports and definitions for Pseudo Dual-Port mode of the Large RAM primitive.

**Table 10.5. Pseudo Dual-Port Mode Signals**

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Clock for Port A |
| wr_data_i[DWA-1:0] | Input | Data Width | Input Data Port A (1 – 32 bits) |
| wr_addr_i[AWA-1:0] | Input | Address Width | Port A Address (10 – 16 bits) |
| wr_clk_en_i | Input | 1 | Port A Clock Enable |
| ben_i[n-1:0] | Input | 4 | Port A Byte Enable (n takes values from 1 to 4) Optional signal<br>For each bit position:<br>*0* – The corresponding byte should be written.<br>*1* – The corresponding byte should not be written. |
| rst_i | Input | 1 | Port A and B Logic Reset |
| dps_i | Input | 1 | Dynamic Power Select |
| rd_addr_i[AWB-1:0] | Input | Address Width | Port B Address (10 – 16 bits) |
| rd_clk_en_i | Input | 1 | Port B Clock Enable |
| rdout_clken_i | Input | 1 | Port B Output Register Clock Enable |
| wr_en_i | Input | 1 | Write Enable |
| rd_data_o[DWB-1:0] | Output | Data Width | Output Data Port B |
| rd_datavalid_o | Output | 1 | Output Enable Port B |
| lramready_o | Output | 1 | Large RAM IP ready indicator |
| errdecb_o[1:0] | Output | 2 | Error Correction indicator |
| errdet_o | Output | 2 | Large RAM IP error status |

Table 10.6 shows the attributes for the Pseudo Dual-Port mode of the Large RAM primitive.

**Table 10.6. Attributes Summary for Pseudo Dual-Port Mode**

| Attribute | Description | Selectable Values | Default |
|---|---|---|---|
| LRAM Type | Type of memory | Single Port; True Dual Port; Pseudo Dual Port; ROM | Single Port (you choose Pseudo Dual ) |
| Clock Polarity | Select polarity of data clock | Active High; Active Low | Active High |
| Internal Clock Delay Control Source | Choose internal or CIB control of clock delay control | Internal Clock Delay Value; Input Port: cib_clkdly_ctrl_i | Internal Clock Delay Value |
| Internal Clock Delay Value | Choose clock delay code | 00; 01; 10; 11 | 00 |
| Preserve Array Enable | Keeps array size from being modified | Unchecked; Checked | Unchecked |
| Global Reset Enable | Allows global reset to affect memory | Unchecked; Checked | Unchecked |
| Provide Byte Enables | Allows you to select Byte Enable options | Unchecked; Checked | Unchecked |
| Unaligned Read Enable | Allows asynchronous reads | Unchecked; Checked | Unchecked |
| Enable ECC | Allows you to enable Error Correction Codes. | Unchecked; Checked | Unchecked |
| Reset Assertion | Selection for the reset assertion to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| Reset Release | Selection for the reset release to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| INIT Bus Write ID | ID for writing initialization data | 0 – 2047 | 0 |
| Memory Initialization | Allows you to initialize memories to all 1s, 0s, or provide a custom initialization by providing a memory file. | None; Initialize to all 0s; Initialize to all 1s; Memory File | None |
| Memory File | When memory file is selected, you can browse to the mem file for custom initialization of RAM. | Button; File browser | Unselected |
| Memory File Format | This option allows you to select if the memory file is formatted as Binary, Hex, or address Hex. | Binary; Hex; Addressed Hex | Binary |
| Write Data Width | Data word width of write port | 1 – 32 | 32 |
| Write Address Width | Address depth of write port | 8 – 16 | 14 |
| Write Enable Polarity | Select enable polarity of WE | Active High; Active Low | Active High |
| Clock Enable Polarity (Write Port) | Select enable polarity for CE | Active High; Active Low | Active High |
| Reset Polarity | Select polarity of reset | Active High; Active Low | Active High |
| Address Width | Address depth of read and write port | 8 – 16 | 14 |
| Clock Enable Polarity (Read Port) | Select enable polarity for CE | Active High; Active Low | Active High |
| Write Input Register | Data in port (wr_data_i) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |
| Read Output Register | Data out port (rd_data_o) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |

For relevant timing diagrams, refer to Figure 10.8 and Figure 10.9.

## 10.4. ROM LRAM

When used as ROM, only one port is used to read. Input can be configured as register in and output can be configured as register out. The SRAM enclosed in the Large RAM IP is synchronous. IP Catalog generates the memory module, as shown in Figure 10.13.

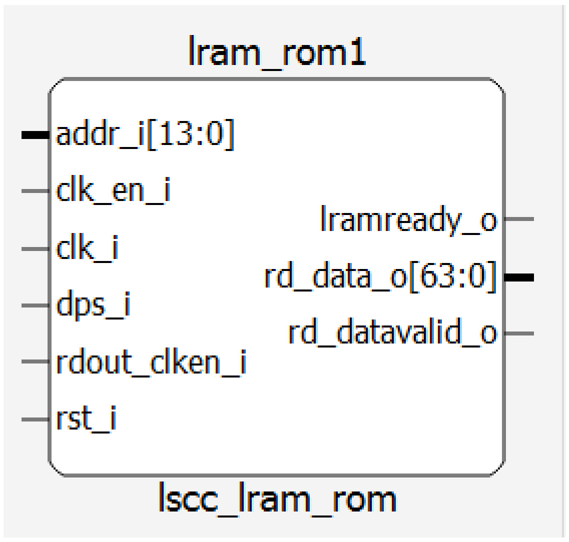


**Figure 10.13. ROM Large RAM Generated by IP Catalog**

The following table lists the ports and definitions for ROM mode of the Large RAM primitive.

**Table 10.7. ROM Mode Signals**

| Port Name | Direction | Width | Description |
|---|---|---|---|
| clk_i | Input | 1 | Clock for Port A |
| addr_i[AWA-1:0] | Input | Address Width | Port A Address (10 – 16 bits) |
| clk_en_i | Input | 1 | Port A Clock Enable |
| rst_i | Input | 1 | Port A Logic Reset |
| dps_i | Input | 1 | Dynamic Power Select |
| rdout_clken_i | Input | 1 | Port A Output Register Clock Enable |
| rd_datavalid_o | Output | 1 | Output Enable Port A |
| rd_data_o[DWB-1:0] | Output | Data Width | Output Data Port A |
| lramready_o | Output | 1 | Large RAM IP ready indicator |
| errdeca_o[1:0] | Output | 2 | Error Correction indicator |
| errdet_o | Output | 1 | Large RAM IP error status |

Table 10.8 shows the attributes for the ROM mode of the Large RAM primitive.

**Table 10.8. Attributes Summary for ROM Mode**

| Attribute | Description | Selectable Values | Default |
|---|---|---|---|
| LRAM Type | Type of memory | Single Port; True Dual Port; Pseudo Dual Port; ROM | Single Port (you choose ROM) |
| Clock Polarity | Select polarity of data clock | Active High; Active Low | Active High |
| Internal Clock Delay Control Source | Choose internal or CIB control of clock delay control | Internal Clock Delay Value; Input Port: cib_clkdly_ctrl_i | Internal Clock Delay Value |
| Internal Clock Delay Value | Choose clock delay code | 00; 01; 10; 11 | 00 |
| Unaligned Read Enable | Allows asynchronous reads | Unchecked; Checked | Unchecked |
| Enable ECC | Allows you to enable Error Correction Codes. | Unchecked; Checked | Unchecked |
| Reset Assertion | Selection for the reset assertion to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| Reset Release | Selection for the reset release to be synchronous or asynchronous to the clock. | Async; Sync | Sync |
| INIT Bus Write ID | ID for writing initialization data | 0 – 2047 | 0 |
| Memory Initialization | Allows you to initialize memories to all 1s, 0s, or provide a custom initialization by providing a memory file. | None; Initialize to all 0s; Initialize to all 1s; Memory File | None |
| Memory File | When memory file is selected, you can browse to the mem file for custom initialization of RAM. | Button; File browser | Unselected |
| Memory File Format | This option allows you to select if the memory file is formatted as Binary, Hex, or address Hex. | Binary; Hex; Addressed Hex | Binary |
| Address Width A | Address depth of read and write port | 8 – 16 | 14 |
| Clock Enable Polarity A | Select enable polarity for CE | Active High; Active Low | Active High |
| Reset Polarity A | Select polarity of reset | Active High; Active Low | Active High |
| Output Register A | Data out port (rd_data_o) can be registered or not depending on this selection. | Unchecked; Checked | Unchecked |

The waveform in Figure 10.14 shows the internal timing for the ROM LRAM. The address is clocked into the SRAM when Clock Enable selection is enabled. In case the output registers are bypassed, the new data is available right after the rising edge of the same clock cycle on which read address is clocked into the SRAM with Clock Enable selection enable.

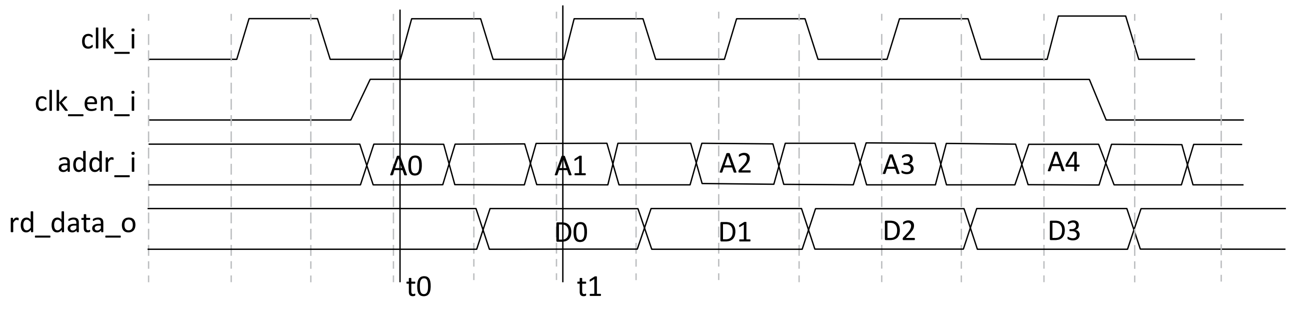


**Figure 10.14. ROM Timing Diagram (Output Register Disabled)**

As shown in Figure 10.14, data flow is as follows:

1. addr_i is clocked in the SRAM at t0.
2. You set clk_en_i port value and get the read back data at t1.

## 10.5. ECC and Byte Enable

This soft IP design implements an ECC module for the Lattice FPGA families that can be applied to increase memory reliability in critical applications. The ECC module provides Single Error Correction - Double Error Detection (SECDED) capability based on a class of optimal minimum odd weight error parity codes that provides better performance than typical Hamming-based SECDED codes. ECC syndrome is calculated over all four bytes of data. If you incorrectly enable byte write together with ECC, the inner ECC is disabled and byte write still works correctly.

Byte Enable feature enables you to 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 Module/IP Block Wizard, while generating the module.

Byte Enable and ECC are mutually exclusive and cannot be used together.

## 10.6. Using Various Data Widths on Various Ports

When LRAM memory is configured as True Dual-Port or Pseudo Dual-Port memory, it has separate Data Width (DW) and Address Width (AW) parameters for ports A and B. The parameters can be configured independently; however, there are a few constraints for their values.

- Data Width of the wide port should be the multiple of narrow port's Data Width. The ratio can be 1, 2 or 4.
- Full memory space for both ports should be the same:

  (2^AW_A)*DW_A = (2^AW_B)*DW_B,

  where AW_X is Address Width of port X and DW_X is Data Width of port X. If Data Widths are equal, Address Widths should be the same. If Data Widths ratio is two, Address Widths difference should be one. And, finally, if Data Widths ratio is four, Address Widths difference should be two.
- When Data Widths are not the same, the ECC is disabled even if Byte Enable is not checked.
- Number of Bytes (NB) used for each port can be calculated using following formulas:

  | | |
  |---|---|
  | For the Narrow Port: | NB = ceil(Data Width/8). |
  | For the Wide Port: | NB = (Data Widths Ratio) * (NB of Narrow port). |

  For example:

  if:

  DW_A = 2 and DW_B = 8,

  then:

  NB_A = 1 and NB_B = 4

If Byte Enable is set for a port, then its width (in bits) is equal to the Number of Bytes for that port.

Byte Enable can be set only if the Number of Bytes is greater than one on the corresponding port, and the number of bits is a multiple of 8 on both ports.

## 10.7. Write Mode Attribute

Any port, which has both Read access and Write access has a Write Mode attribute. This attribute is available for Port A in the Single Port Mode and for both Port A and Port B in True Dual-Port Mode. In Pseudo Dual-Port and ROM Modes, no Write Mode attribute is available as there are no ports with both Read access and Write access.

There are three possible values for Write Mode attribute: Normal, Write Through, and Read Before Write. All three modes are supported in Single Port Large RAM, while only the first two are supported in the True Dual-Port Large RAM.

- In Normal mode, the output data is not changed nor updated during the write operation.
- In Write Through mode, the output data is updated with the input data during the write cycle.
- In Read Before Write mode, the output data port is updated with the existing data stored in the write address, during a write cycle. This mode is supported only in the Single Port LRAM.

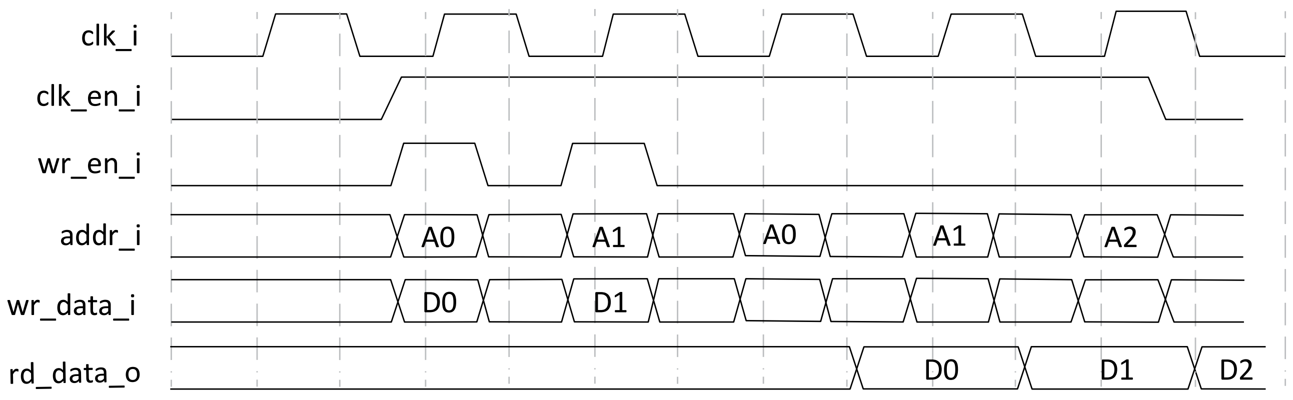


**Figure 10.15. Single-Port LRAM Timing Diagram in Normal Mode (Output Register Disabled)**

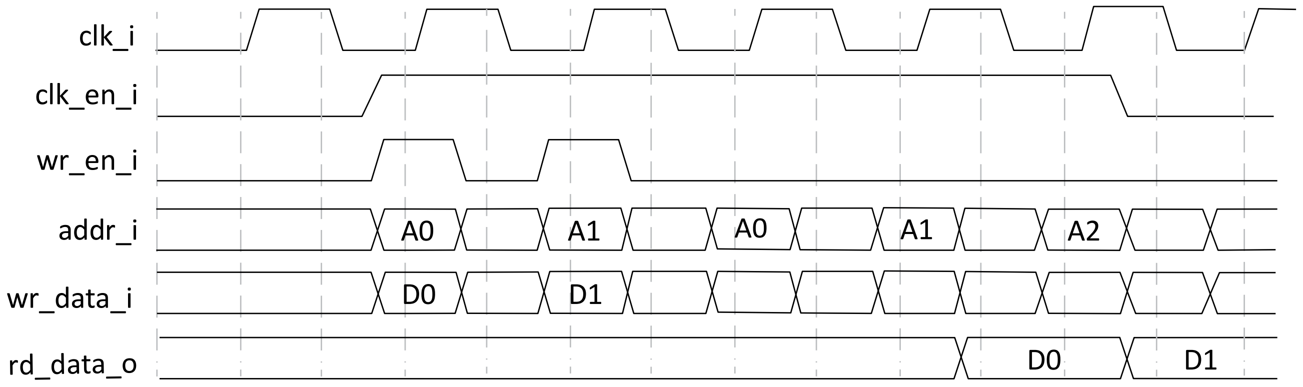


**Figure 10.16. Single-Port LRAM Timing Diagram in Normal Mode (Output Register Enabled)**

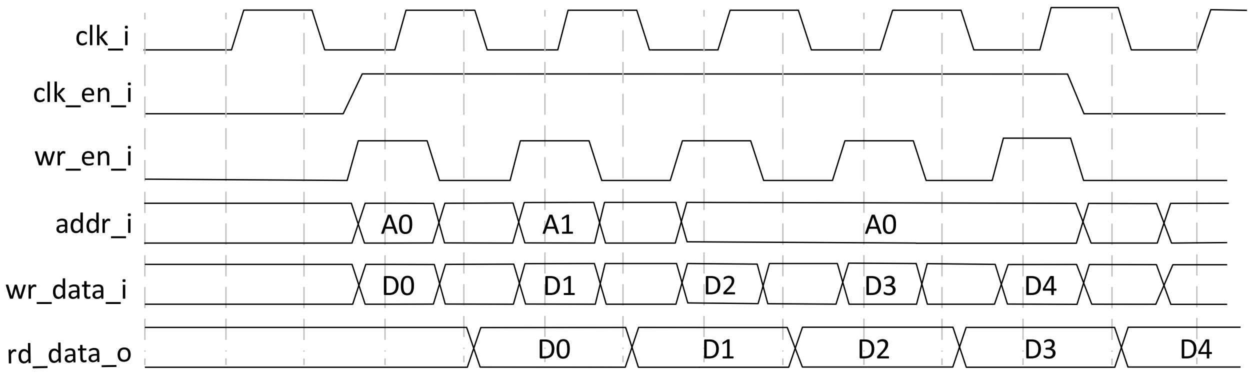


**Figure 10.17. Single-Port LRAM Timing Diagram in Write Through Mode (Output Register Disabled)**

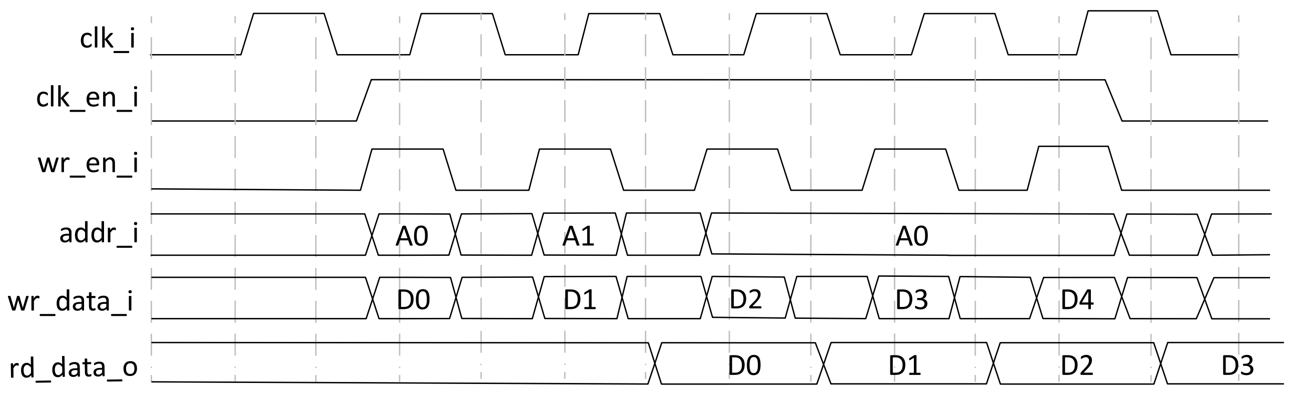


**Figure 10.18. Single-Port LRAM Timing Diagram in Write Through Mode (Output Register Disabled)**

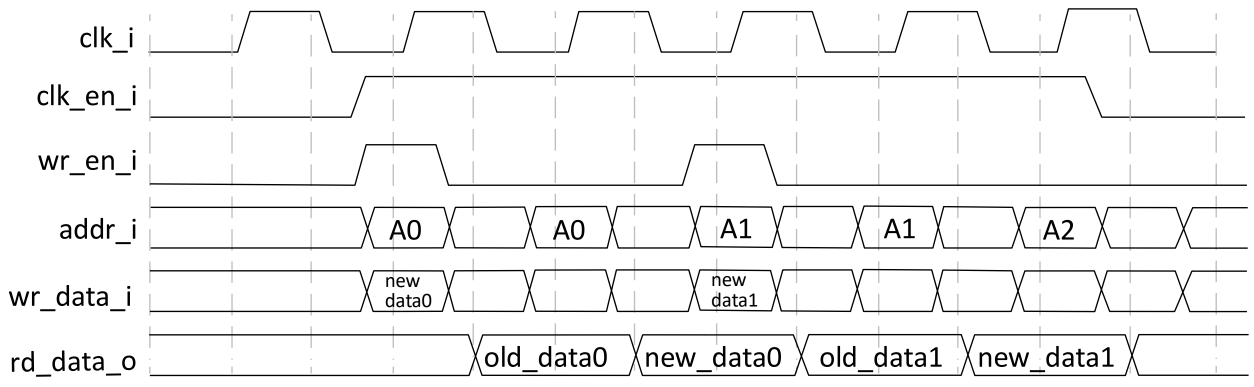


**Figure 10.19. Single-Port LRAM Timing Diagram in Read Before Write Mode (Output Register Disabled)**

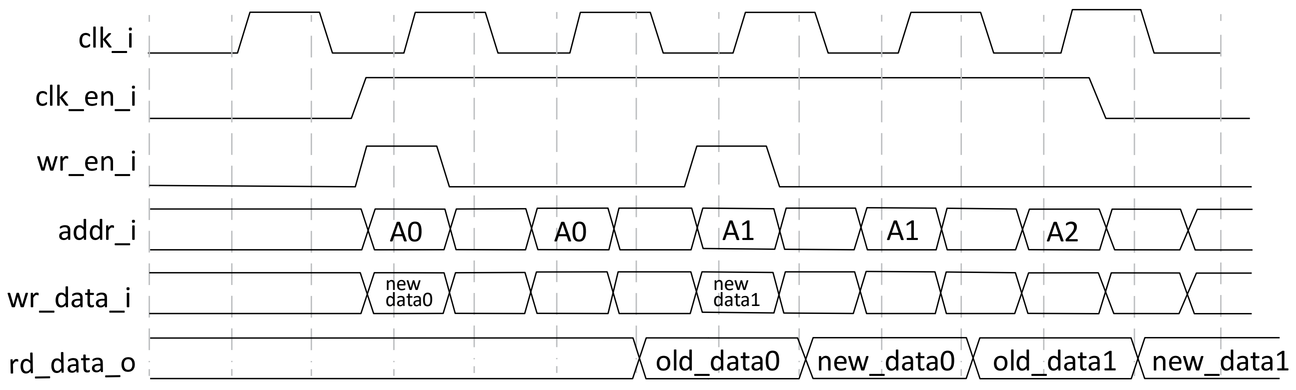


**Figure 10.20. Single-Port LRAM Timing Diagram in Read Before Write Mode (Output Register Enabled)**

# 11. 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.

## 11.1. Initialization File Formats

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

- Binary File
- Hex File
- Addressed Hex

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

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) is 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.

### 11.1.1. 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
```

### 11.1.2. 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
```

### 11.1.3. 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.

## Technical Support Assistance

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

# Revision History

**Revision 1.1, June 2020**

| Section | Change Summary |
|---|---|
| — | • Changed document title to Memory Usage Guide for Nexus Platform<br>• Added Certus-NX support. |
| Introduction | Added LRAM ROM information. |
| Memory Generation | Updated the following figures:<br>• Figure 2.1. Memory Modules Available in IP Catalog<br>• Figure 2.2. IP Catalog in Lattice Radiant Software |
| Large RAM (LRAM) | General revision |

**Revision 1.0, November 2019**

| Section | Change Summary |
|---|---|
| All | Initial release |