Lattice Avant Embedded Memory User Guide

Preliminary Technical Note

FPGA-TN-02289-0.82

September 2024

Disclaimers

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

Inclusive Language

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

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AW	Address Width
DW	Data Width
ECC	Error Correction Code
EBR	Embedded Block RAM
FIFO	First In First Out
GSR	Global Set/Reset
IP	Intellectual Property
LUT	Look Up Table
LSB	Least Significant Bit
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
VHDL	VHSIC Hardware Description Language

1. Introduction

This technical note discusses memory usage for the Lattice Semiconductor devices built on the Lattice Avant™ platform. It is intended to be used by design engineers as a guide when integrating the Embedded Block RAM (EBR)-based and Programmable Function Unit(PFU)-based memories for these device families in the 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 capabilities of the EBR Block RAM and PFU RAM are described in this document. You 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 Parameterizable Module Inferencing (PMI) – PMI allows you to skip the graphical interface and utilize the configurable memory primitives on-the-fly from the Lattice Radiant project navigator. You can set the parameters and the control signals needed either in Verilog or VHDL. 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 the 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 use the IP Catalog to easily specify a variety of memories in the 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 as follows:

- Distributed Memory Modules
  - Distributed Pseudo 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)

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

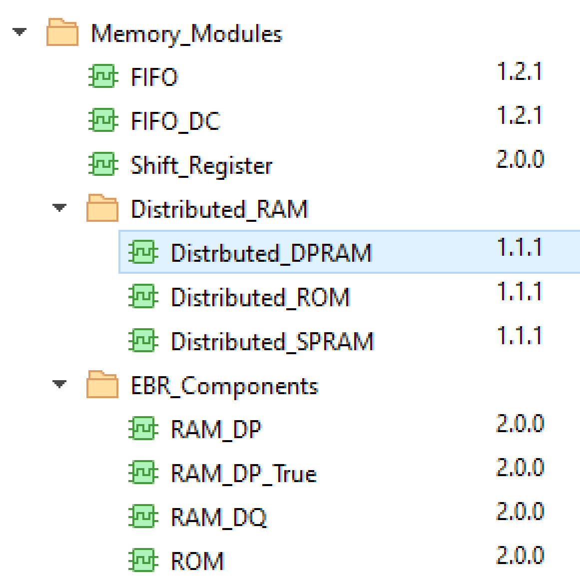

Figure 2.1. Memory Modules Available in IP Catalog

2.1. IP Catalog Flow

The IP Catalog allows you to generate, create, or open any of the available modules for Lattice Avant 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.

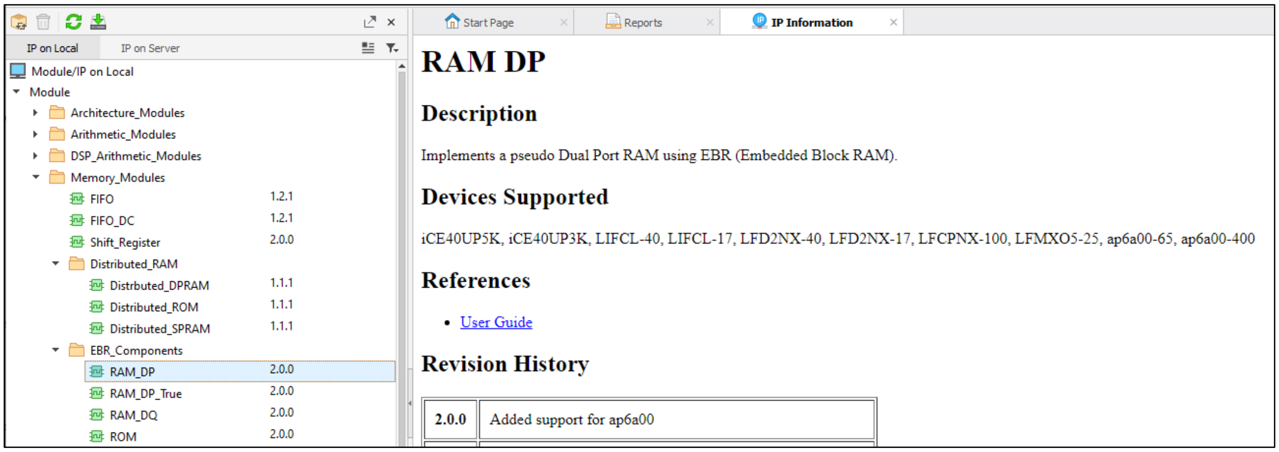

Figure 2.2. IP Catalog in the Lattice Radiant Software

The left section of the IP Catalog window includes the Module Tree. The Memory Modules are categorized as Distributed RAM, EBR Components, FIFOs, 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 36.

To generate an EBR-based Pseudo Dual Port RAM:

1. Double-click **ram_dp** under the **EBR_Components**.
2. Fill out the module information, as shown in Figure 2.3.
3. Click the **Next** button.

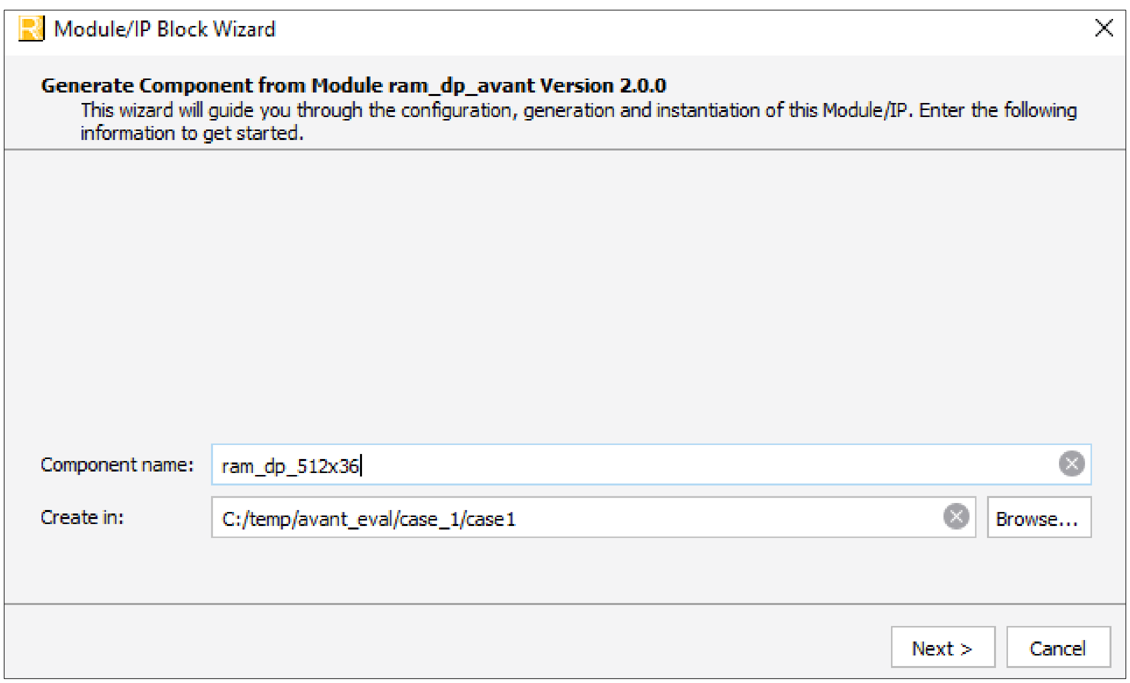

Figure 2.3. Example: Generating Pseudo Dual Port RAM (RAM_DP) Using IP Catalog

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

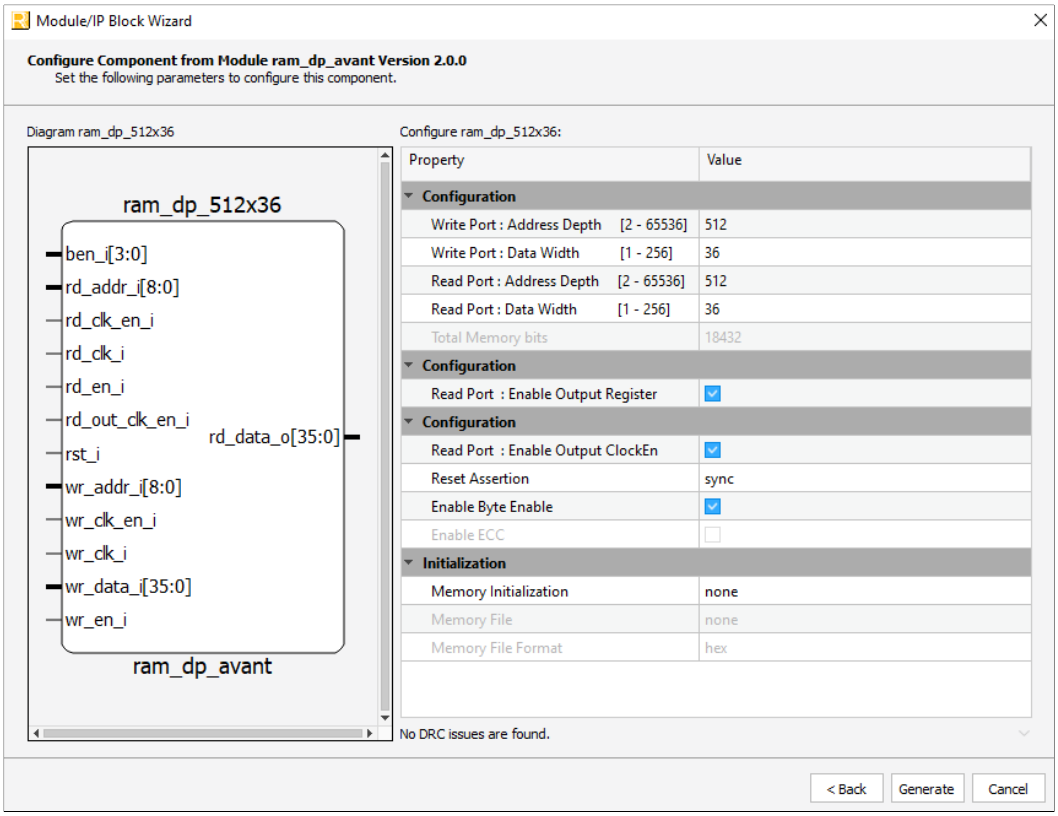

Figure 2.4. Example: Generating Pseudo Dual Port RAM (RAM_DP) Module Customization – General Options

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

Once this module is in the Lattice Radiant project, it can be instantiated within other modules.

2.2. Utilizing PMI

You can set the parameters and control signals needed 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 Online Help system.

To do this, create a Verilog or VHDL behavior code for the memory and the synthesis tool automatically identifies it as memory and synthesizes it as a distributed or EBR memory. In general, memory sizes smaller than 1 kb are automatically mapped to Distributed mode and those larger than 1 kb are implemented using EBRs. This default option can be over-ridden using the syn_ramstyle or syn_romstyle 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 Avant platform devices. When instantiating the primitives, you must 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 its 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 enable features that mask selective bits. These features are available in the EBR-based RAM modules.

3.1. ECC in Memory Modules

An error-correction code (ECC) 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. ECC write is supported only in x64 write mode with 8-bit ECC. ECC read may be performed in x1, x2, x4, x8, x16, x32, or x64 modes.

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

- one_err_det_o = 1 Indicates there is a 1-bit error that is fixed.
- two_err_det_o = 1 Indicates there is a 2-bit error that cannot be corrected.

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

3.2. Byte Enable

Byte enable is a feature available in selected RAM modules where you can mask the bytes written in the RAM. The byte enable feature is available only when the data width is greater than nine bits. Each byte enable bit controls the write data operation enable for the nine 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 a 72-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, 35, 44, 53, 62, and 71 are parity bits, which you ignore in x8, x16, x32, and x64 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)
ByteEn(4)	Data(44:36)
ByteEn(5)	Data(53:45)
ByteEn(6)	Data(62:54)
ByteEn(7)	Data(71:63)

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 Distributed 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, 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 Avant 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 that go into the EBR block are also registered.

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. The global set/reset (GSR) signal also controls the output registers.

4.1.3. Timing

To correctly write to a memory cell in the EBR block, the correct address must 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, lead to the corruption of memory contents.

During a read cycle, a similar issue can occur such 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. For more information, refer to the timing preferences in the Online Help documents.

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

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

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

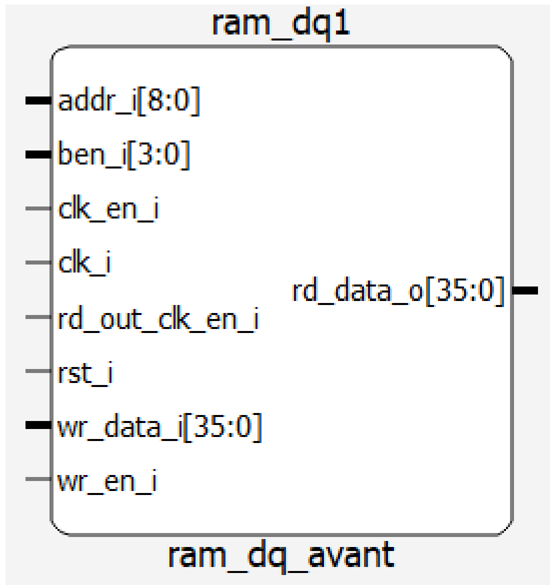

Figure 4.1. Single-Port Memory Module Generated by IP Catalog

Figure 4.2 shows the primitive that can be instantiated for the Single Port RAM. The primitive name is SP32K 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 the Lattice Radiant software installation folder.

Note that each EBR can accommodate 36 kb of memory. If the memory required is larger than 36 kb, then cascading can be used, using the CS port (CS[2:0] in this case).

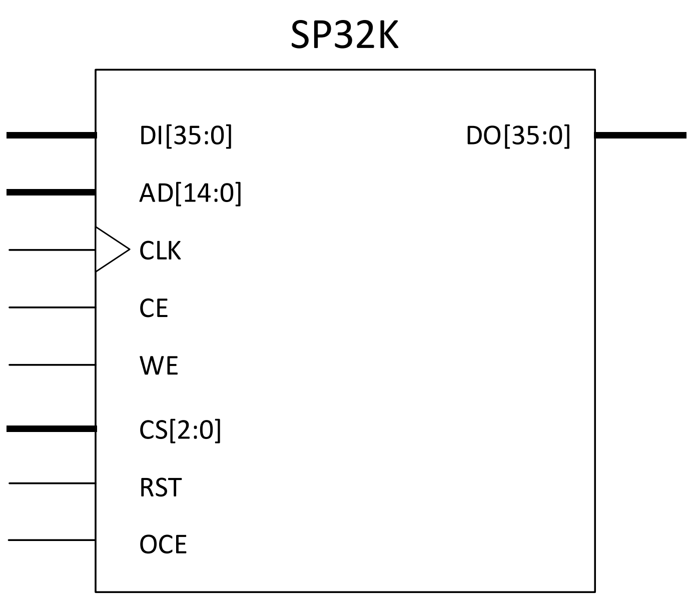

Figure 4.2. Single Port RAM Primitive for Lattice Avant Platform Devices

The various ports and definitions for Single-Port Memory are listed in Table 4.1. The table below lists the corresponding ports for the module generated by IP Catalog.

Table 4.1. EBR-Based Single-Port Memory Port Definitions[1]

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 if 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	Byte Enable Width	Byte Enable

Note:

1. Address width is calculated from address depth.

Each EBR block consists of 36,864 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 36 kb Memory in Lattice Avant Platform Devices

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
32,768 × 1	DI	DO	AD[14:0]
16,384 × 2	DI[1:0]	DO[1:0]	AD[13:0]
8,192 × 4	DI[3:0]	DO[3:0]	AD[13:0]
4,096 × 9	DI[8:0]	DO[8:0]	AD[11:0]
2,048 × 18	DI[17:0]	DO[17:0]	AD[10:0]
1,024 × 36	DI[35:0]	DO[35:0]	AD[9:0]
512 × 72	DI[71:0]	DO[71:0]	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 Lattice Avant 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
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 the memories to all 1s, 0s, or use a custom initialization by providing a memory file.	none, 0s, 1s, file	none
Memory File	When a memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	—

Configuration Tab Attributes	Description	Values	Default Value
Memory File Format	This option allows you to select if the memory file is formatted as binary or hex.	binary, hex	hex

You have the option to enable the output registers for RAM_DQ. The waveforms in Figure 4.3 and Figure 4.4 show the internal timing waveforms for the Single Port RAM (RAM_DQ) without output registers and with output registers respectively.

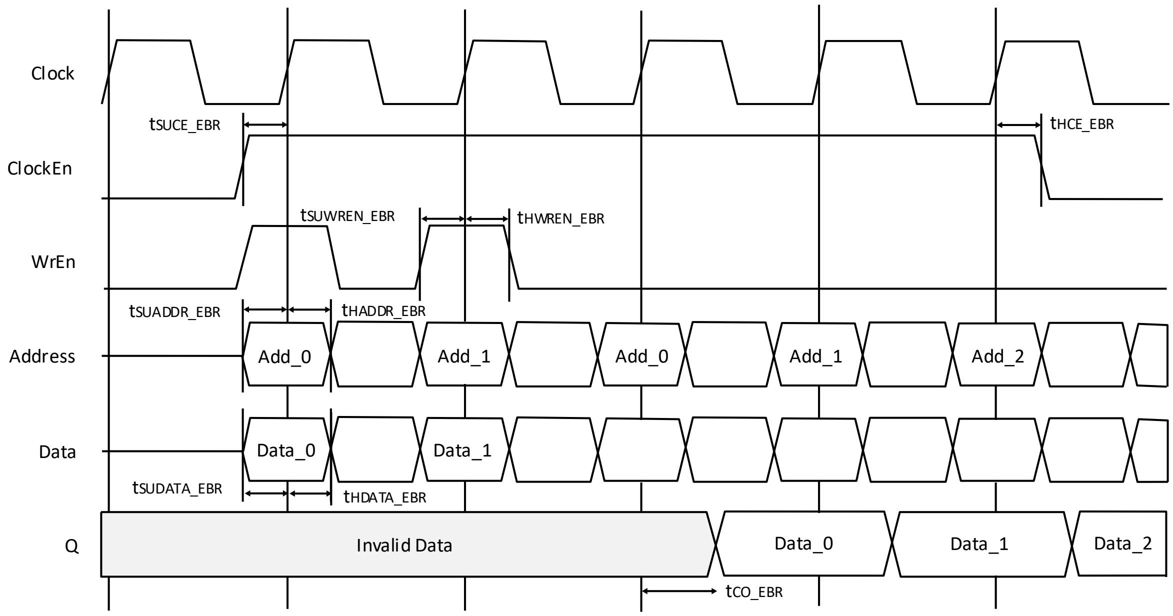

Figure 4.3. Single Port RAM Timing Waveform – without Output Registers

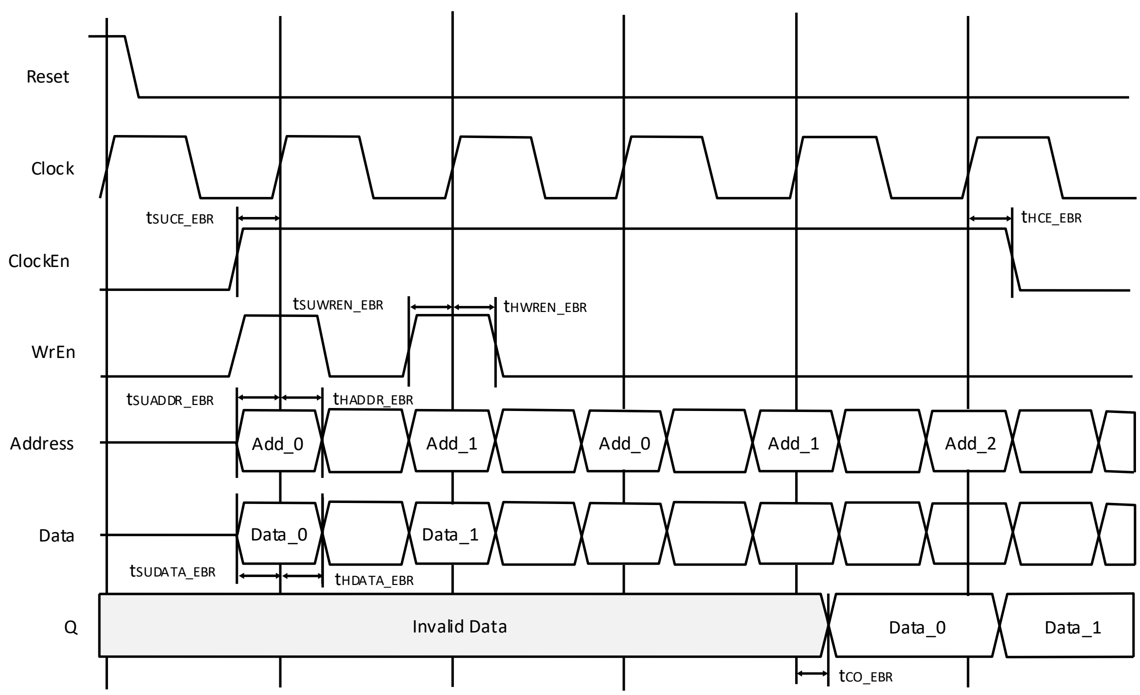

Figure 4.4. Single Port RAM Timing Waveform – with Output Registers

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

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

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

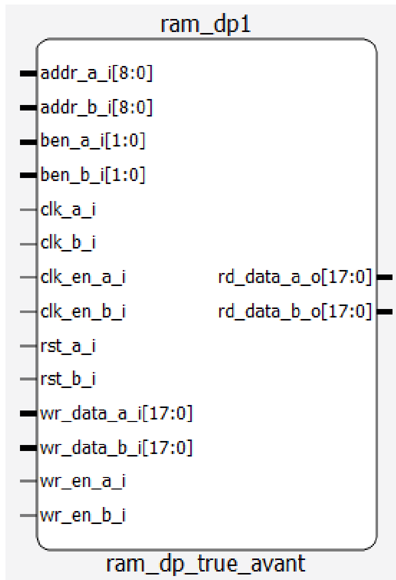

Figure 4.5. True Dual-Port Memory Module Generated by IP Catalog

Figure 4.6 provides the primitive that can be instantiated for the True Dual Port RAM. The primitive name is DP32K 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 the Lattice Radiant software installation folder.

Note that each EBR can accommodate 36 kb of memory. If the memory required is larger than 36 kb, then, cascading can be used, using the CS port (CSA[2:0] and CSB[2:0] in this case).

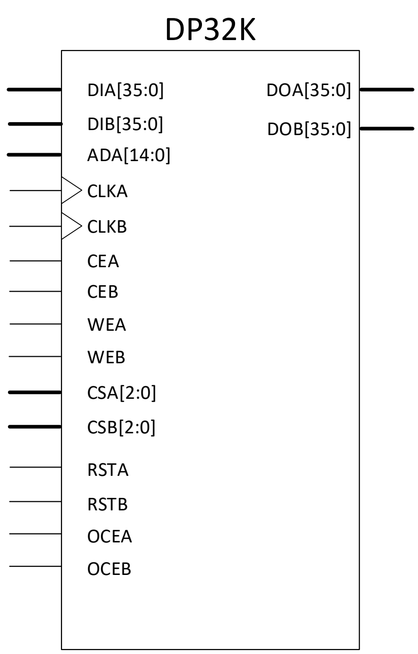

Figure 4.6. True Dual Port RAM Primitive for Lattice Avant Platform Devices

The various ports and definitions for True Dual-Port RAM are listed in Table 4.4. The table below 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	Byte Enable Width	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	Byte Enable Width	Byte Enable for Port B

Each EBR block consists of 36,864 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 36 kb Memory for Avant 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
32,768 × 1	DataInA	DataInB	QA	QB	AddressA(14:0)	AddressB(14:0)
16,384× 2	DataInA(1:0)	DataInB(1:0)	QA(1:0)	QB(1:0)	AddressA(13:0)	AddressB(13:0)
8,192 × 4	DataInA(3:0)	DataInB(3:0)	QA(3:0)	QB(3:0)	AddressA(12:0)	AddressB(12:0)
4,096 × 9	DataInA(8:0)	DataInB(8:0)	QA(8:0)	QB(8:0)	AddressA(11:0)	AddressB(11:0)
2,048 × 18	DataInA(17:0)	DataInB(17:0)	QA(17:0)	QB(17:0)	AddressA(10:0)	AddressB(10:0)
1,024 × 36	DataInA(35:0)	DataInB(35:0)	QA(35:0)	QB(35:0)	AddressA(9:0)	AddressB(9:0)

Table 4.6 shows the various attributes available for True Dual-Port Memory (RAM_DP). You can select some of these attributes through the IP Catalog interface.

Table 4.6. True Dual-Port RAM Attributes for Lattice Avant 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	1–512	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	1–512	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

Configuration Tab Attributes	Description	Values	Default Value
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 the memories to all 1s, 0s, or use a custom initialization by providing a memory file.	none, 0s, 1s, file	none
Memory File	When a memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	—
Memory File Format	This option allows you to select if the memory file is formatted as binary or hex.	binary, hex	hex

You have the option to enable the output registers for RAM_DP_TRUE. Waveforms in Figure 4.7 and Figure 4.8 show the internal timing waveforms for the True Dual Port RAM (RAM_DP_TRUE) without output registers and with output registers respectively.

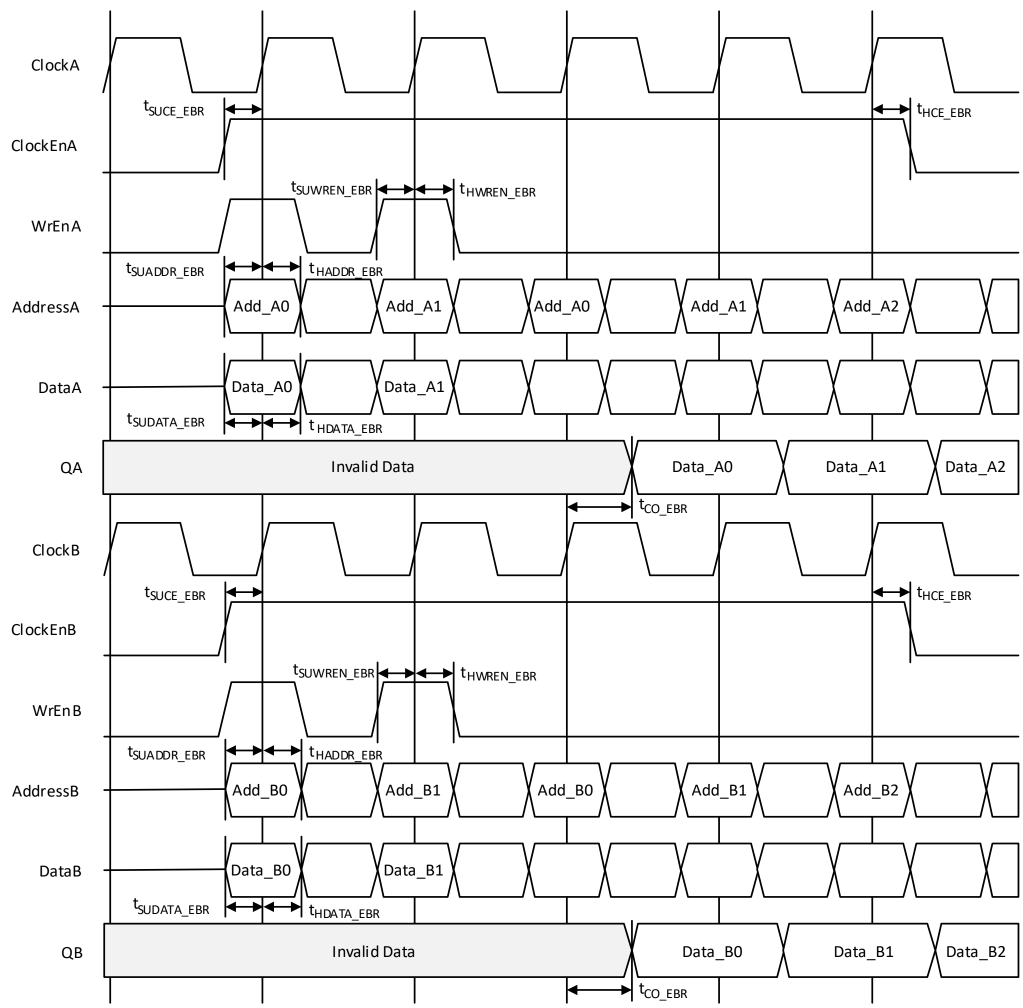

Figure 4.7. True Dual Port RAM Timing Waveform – without Output Registers

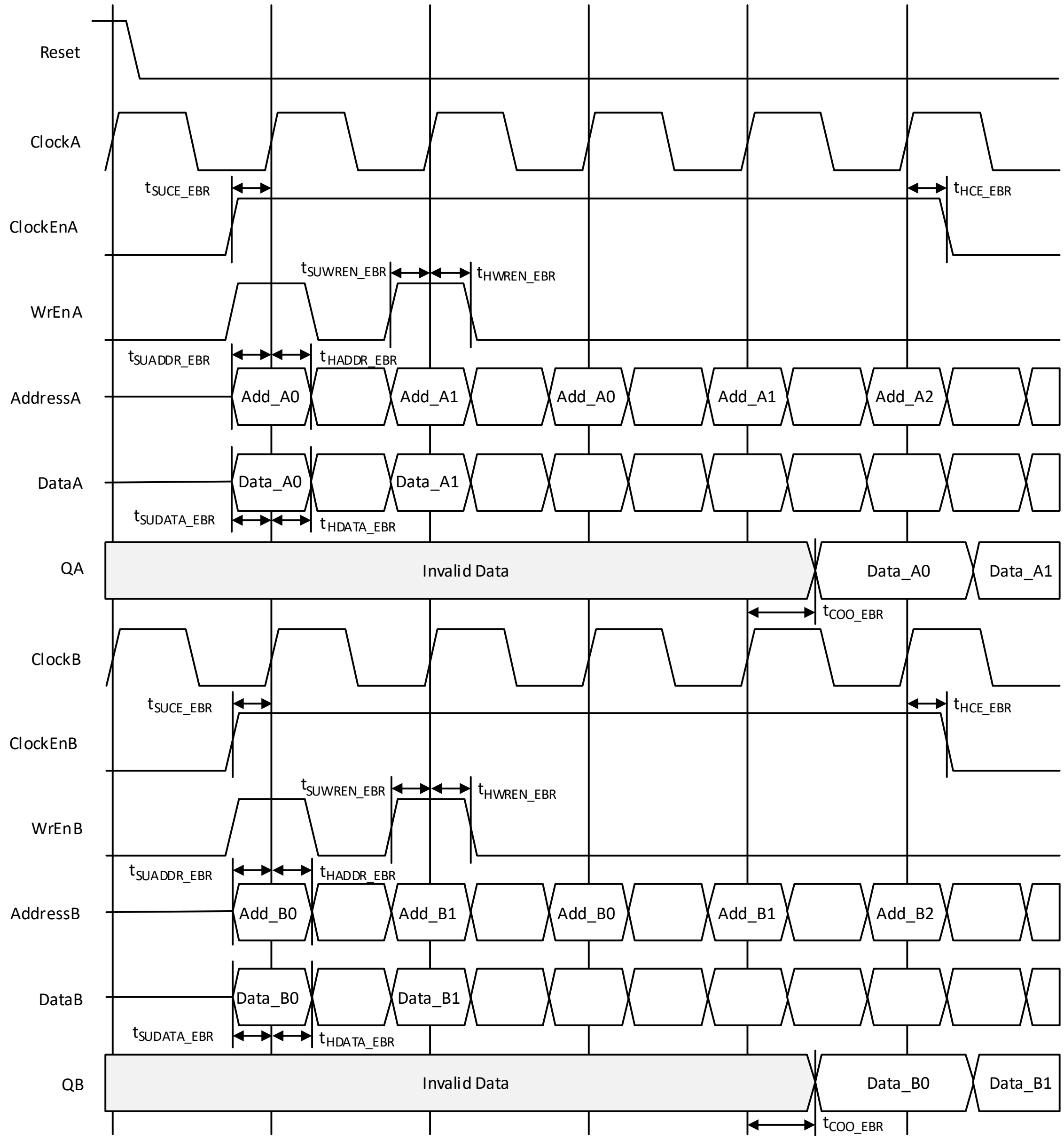

Figure 4.8. True Dual Port RAM Timing Waveform – with Output Registers

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

FPGAs built on the Lattice Avant 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 for the memory size as per design requirement.

IP Catalog generates the memory module, as shown in Figure 4.9.

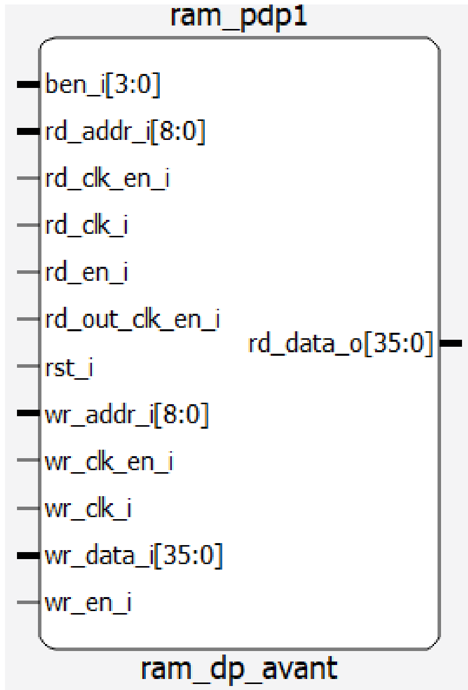

Figure 4.9. Pseudo Dual-Port Memory Module Generated by IP Catalog

Figure 4.10 shows the primitive that can be instantiated for the Pseudo-Dual Port RAM. The primitive name is PDP32K 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 the Lattice Radiant software installation folder.

Note that each EBR can accommodate 36 kb of memory. If the memory required is larger than 36 kb, then, cascading can be used, using the CS ports (CSW[2:0] and CSR[2:0] in this case).

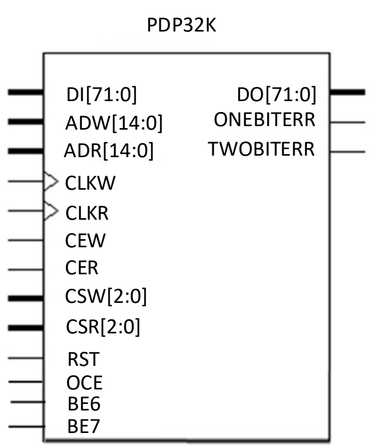

Figure 4.10. Pseudo-Dual Port RAM Primitive for Lattice Avant Platform Devices

The various ports and definitions for Pseudo Dual-Port memory are listed in Table 4.7. The table below 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	Byte Enable Width	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 36,864 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 36 kb Memory for Lattice Avant Platform Devices

Dual Port Memory Size	Input Data Write Port	Output Data Read Port	Address Write Port	Address Read Port
32,768 × 1	Data	Q	WrAddress(14:0)	RdAddress(14:0)
16,384 × 2	Data(1:0)	Q(1:0)	WrAddress(13:0)	RdAddress(13:0)
8,192 × 4	Data(3:0)	Q(3:0)	WrAddress(12:0)	RdAddress(12:0)
4,096 × 9	Data(8:0)	Q(8:0)	WrAddress(11:0)	RdAddress(11:0)
2,048 × 18	Data(17:0)	Q(17:0)	WrAddress(10:0)	RdAddress(10:0)
1,024 × 36	Data(35:0)	Q(35:0)	WrAddress(9:0)	RdAddress(9:0)
512 × 72	Data(71:0)	Q(71:0)	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 Lattice Avant 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
Enable ECC	This option allows you to enable error correction codes.	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 the memories to all 1s, 0s, or use a custom initialization by providing a memory file.	none, 0s, 1s, file	none

Configuration Tab Attributes	Description	Values	Default Value
Memory File	When a memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	—
Memory File Format	This option allows you to select if the memory file is formatted as binary or hex.	binary, hex	hex

You have the option to enable the output registers for Pseudo-Dual Port RAM (RAM_DP). Waveforms in Figure 4.11 and Figure 4.12 show the internal timing waveforms for Pseudo-Dual Port RAM (RAM_DP) without output registers and with output registers respectively.

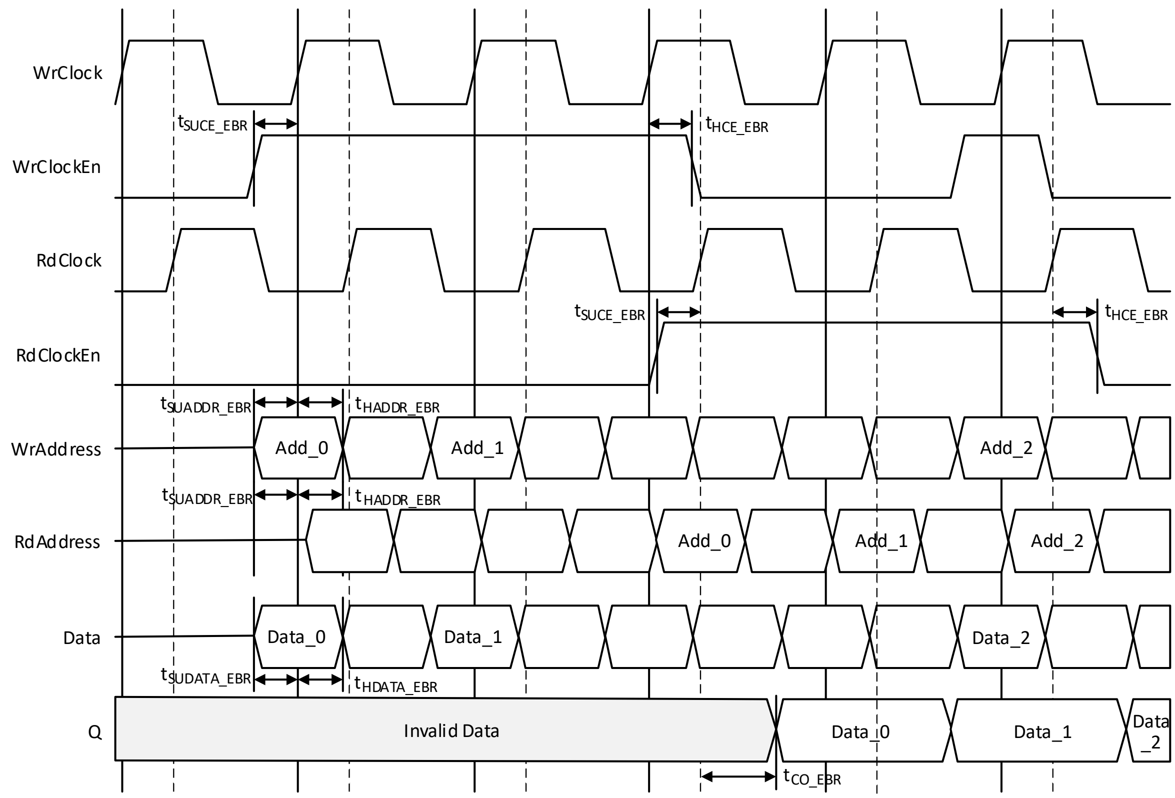

Figure 4.11. PSEUDO DUAL PORT RAM Timing Diagram – without Output Registers

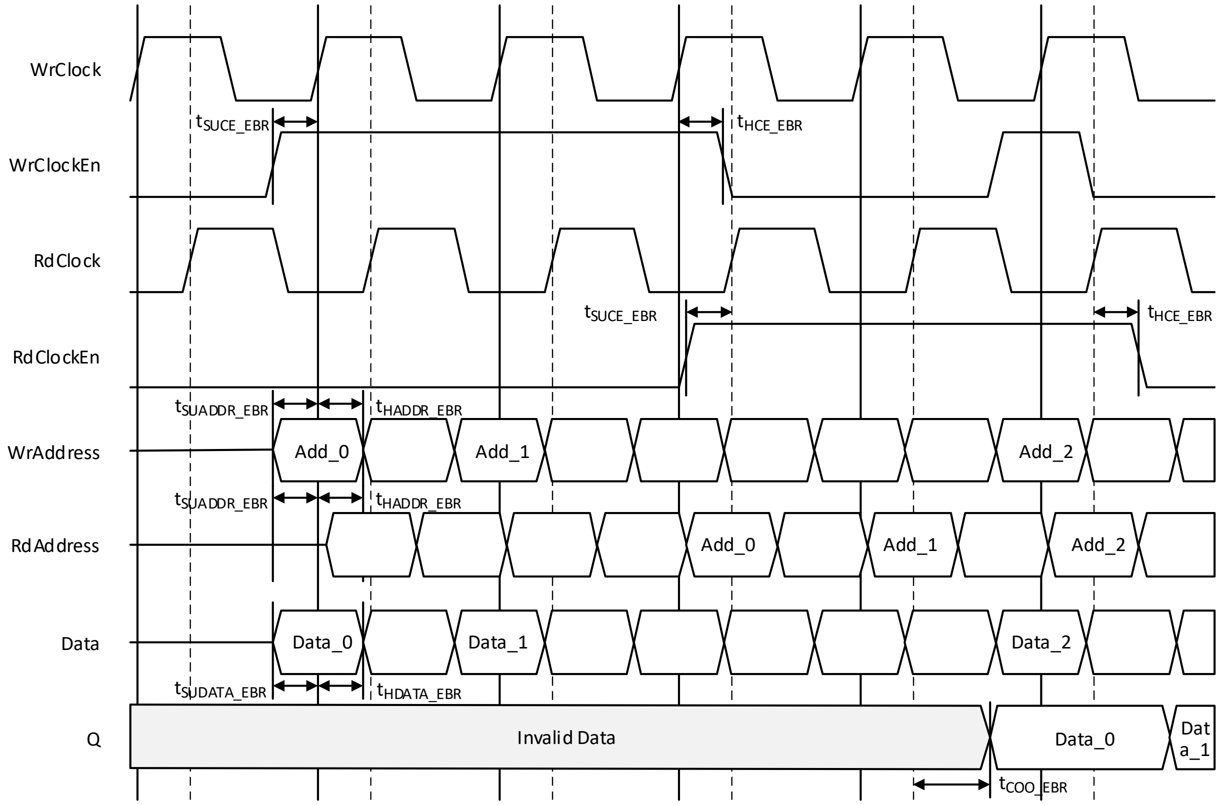

Figure 4.12. PSEUDO DUAL PORT RAM Timing Diagram – with Output Registers

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

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

IP Catalog generates the memory module, as shown in Figure 4.13.

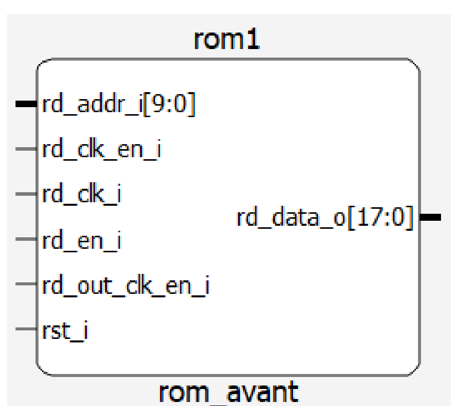

Figure 4.13. ROM – Read Only Memory Module Generated by IP Catalog

The various ports and definitions are listed in Table 4.10. The table below 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, you must provide the initialization file to pre-initialize the contents of the ROM. These files are the *.mem files and they can be of binary or hex formats. The initialization files are discussed in detail in the Initializing Memory section.

Each EBR block consists of 36,864 bits of RAM. The values for xs (for address) and ys (data) for each EBR block for the devices are included in Table 4.11.

Table 4.11. ROM Memory Sizes for 36 kb Memory for Lattice Avant Platform Devices

Dual Port Memory Size	Output Data Read Port	Address Read Port
32,768 × 1	Q	RdAddress(14:0)
16,384 × 2	Q(1:0)	RdAddress(13:0)
8,192 × 4	Q(3:0)	RdAddress(12:0)
4,096 × 9	Q(8:0)	RdAddress(11:0)
2,049 × 18	Q(17:0)	RdAddress(10:0)
1,024 × 36	Q(35:0)	RdAddress(9:0)
512 × 72	Q(71:0)	RdAddress(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 Lattice Avant Platform Devices

Configuration Tab Attributes	Description	Values	Default Value
Address Depth	Address depth of the read and write port	2–65536	512
Data Width	Data word width of the read and write port	1–512	36
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 the memories to all 1s, 0s, or use a custom initialization by providing a memory file.	0s, 1s, file	0s
Memory File	When a memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	—
Memory File Format	This option allows you to select if the memory file is formatted as binary or hex.	binary, hex	hex

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 without output registers and with output registers respectively.

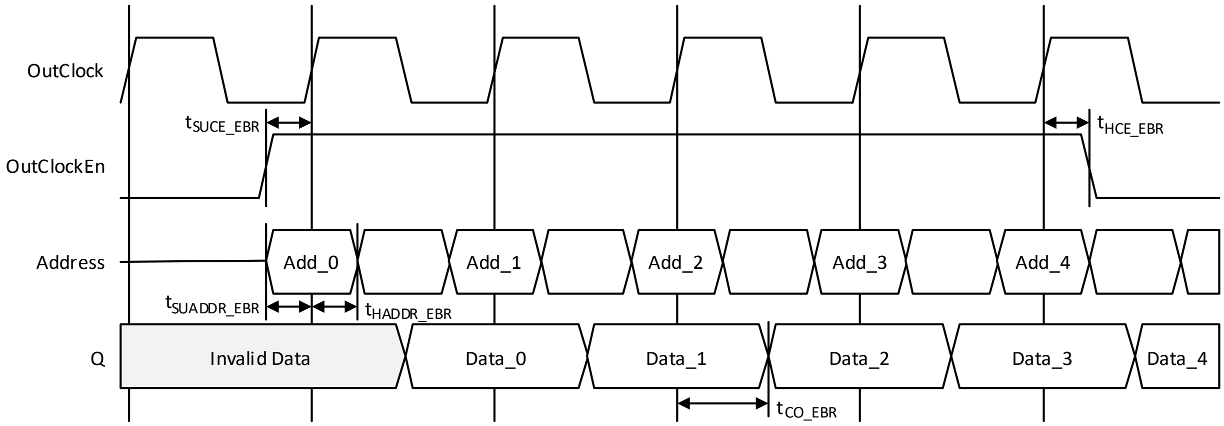

Figure 4.14. ROM Timing Waveform – without Output Registers

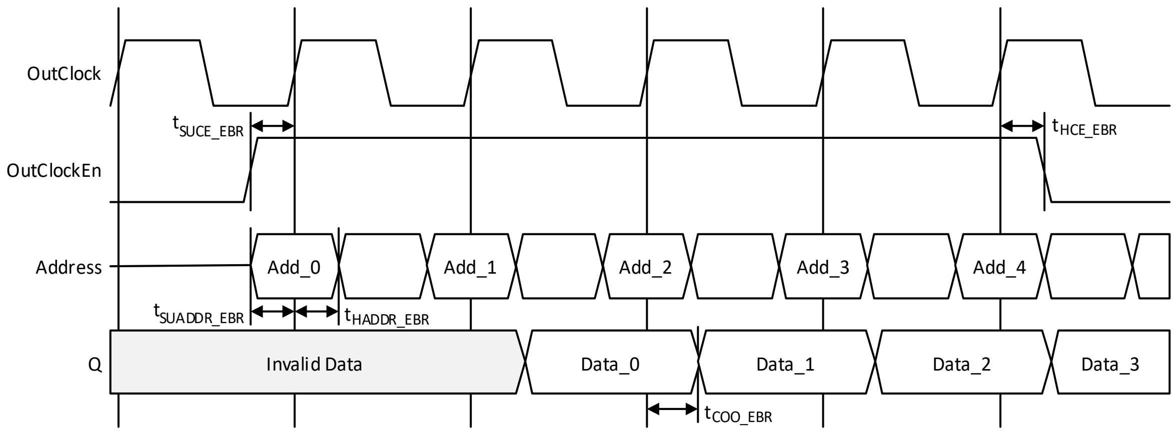

Figure 4.15. ROM Timing Waveform – with Output Registers

5. First In First Out (FIFO) Memory

Lattice Avant platform devices support two different types of FIFOs:

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

The EBR blocks in Lattice Avant 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 the operation are discussed in detail in the following section.

5.1. Single-Clock FIFO (FIFO) – EBR-Based or LUT-Based

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

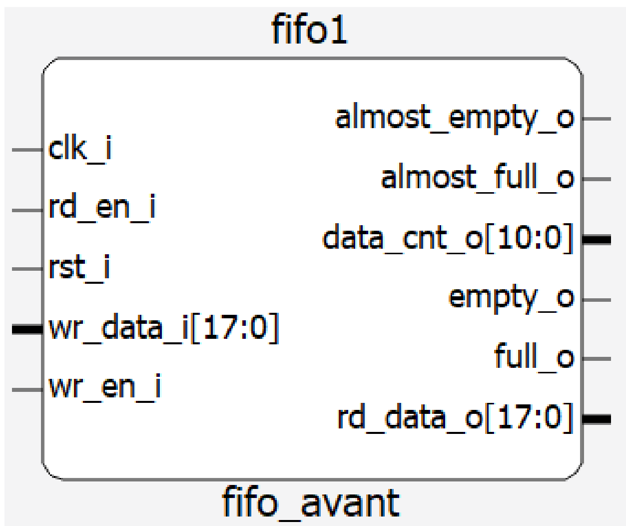

Figure 5.1. FIFO Module Generated by IP Catalog

The various ports and 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 Lattice Avant 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 two.	2–<Max that can fit in the device>	1024
Controller Implementation	FIFO Controller Implementation Option To enable the implementation type as LUT-based, or enable the Data Count output, you need to select LUT.	LUT, HW	HW
Enable FWFT	Enables the First Word Fall Through mode. This is only available in HW mode.	true, false	false
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
Reset Assertion	Selection for the Reset to be synchronous or asynchronous to the Clock	async, sync	sync
Enable 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 the 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 Assert Level	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 De-assert Level	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	2
Enable 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. 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 Assert Level	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 De-assert Level	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	1022
Enable 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 the FIFO.

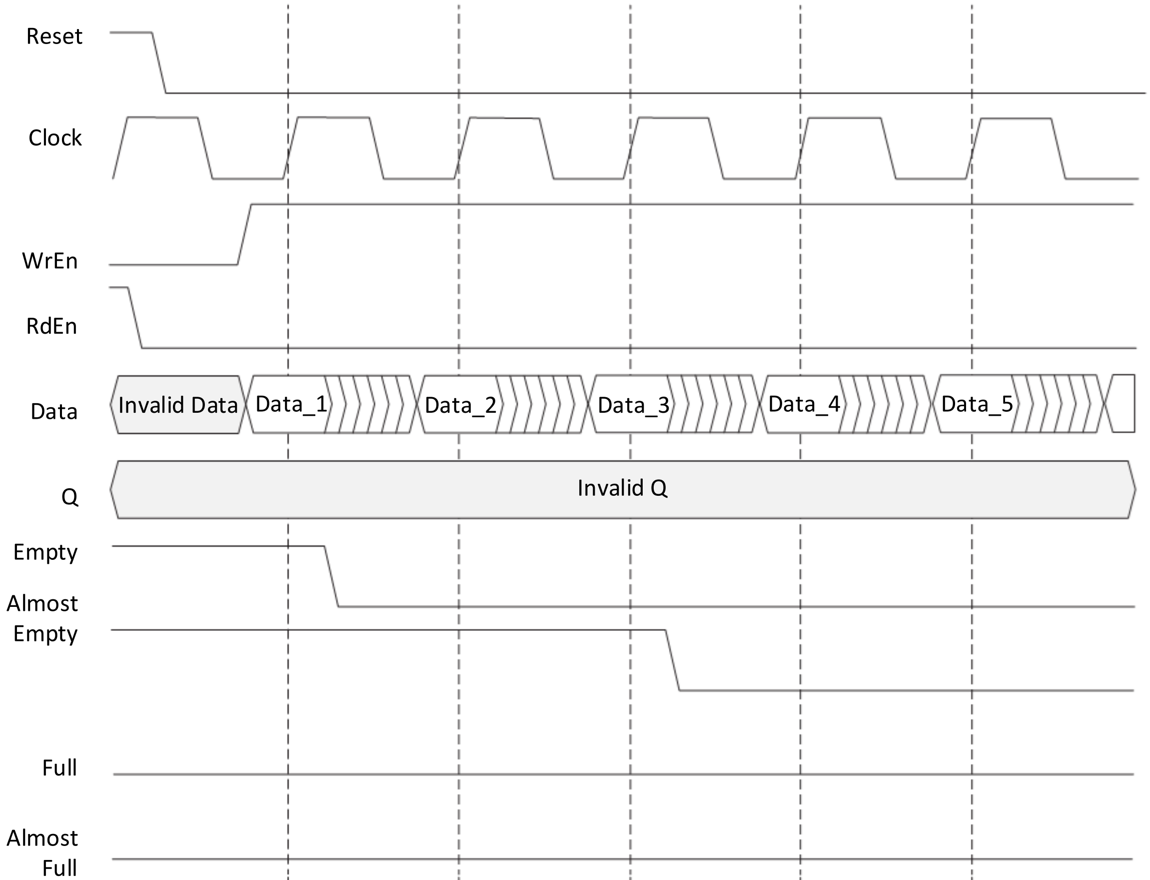

Figure 5.2. FIFO Without Output Registers – Start of Data Write Cycle

The WrEn signal must be high to start writing to 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 to 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.

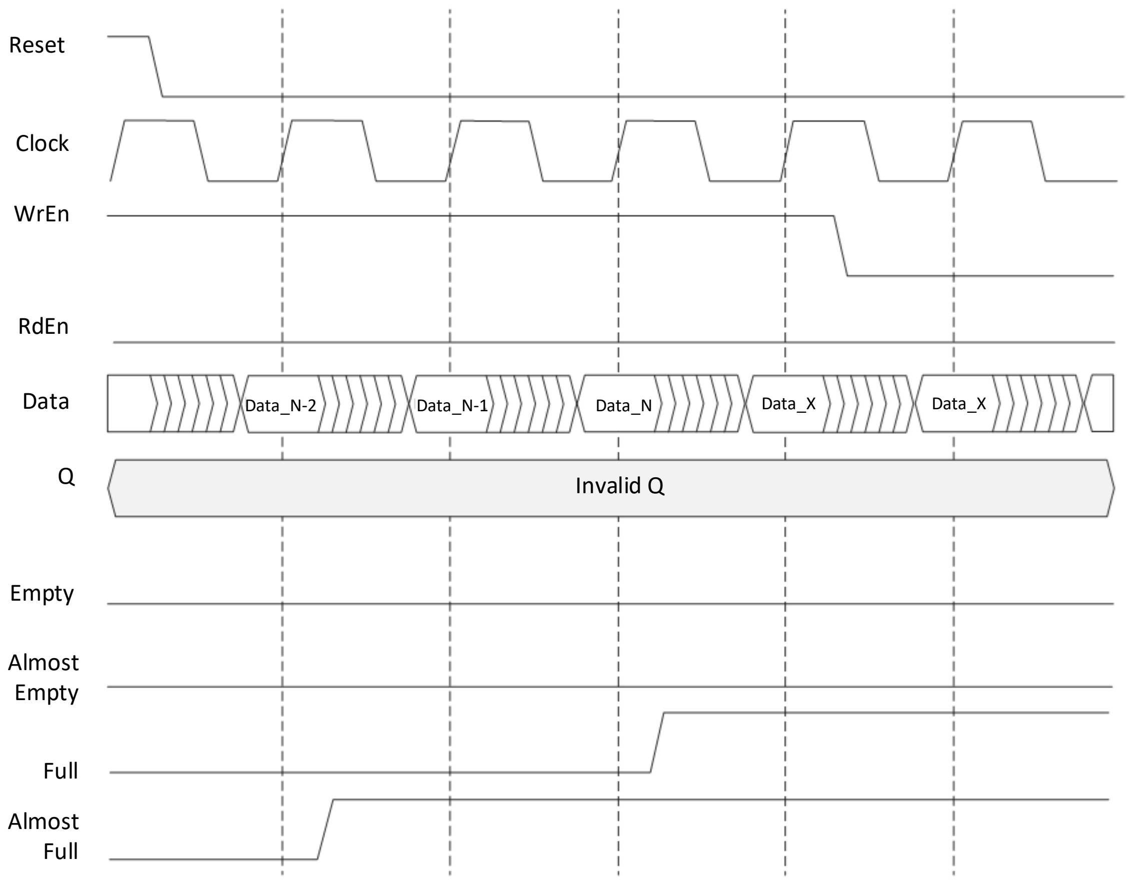

Figure 5.3. FIFO Without Output Registers – End of Data Write Cycle

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 because the FIFO is full, with the Full flag being high.

Examine the waveforms when the contents of the FIFO are read out. Figure 5.4 shows the start of the read cycle. The RdEn signal goes high and the data read starts. The Full and Almost Full flags are de-asserted.

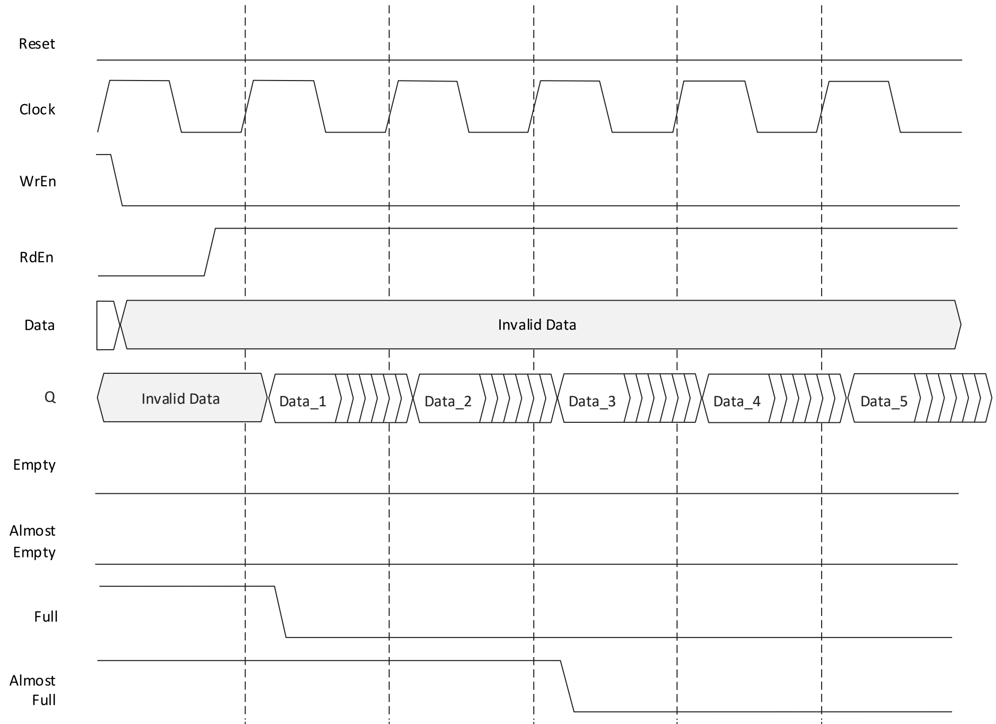

Figure 5.4. FIFO Without Output Registers – Start of Data Read Cycle

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

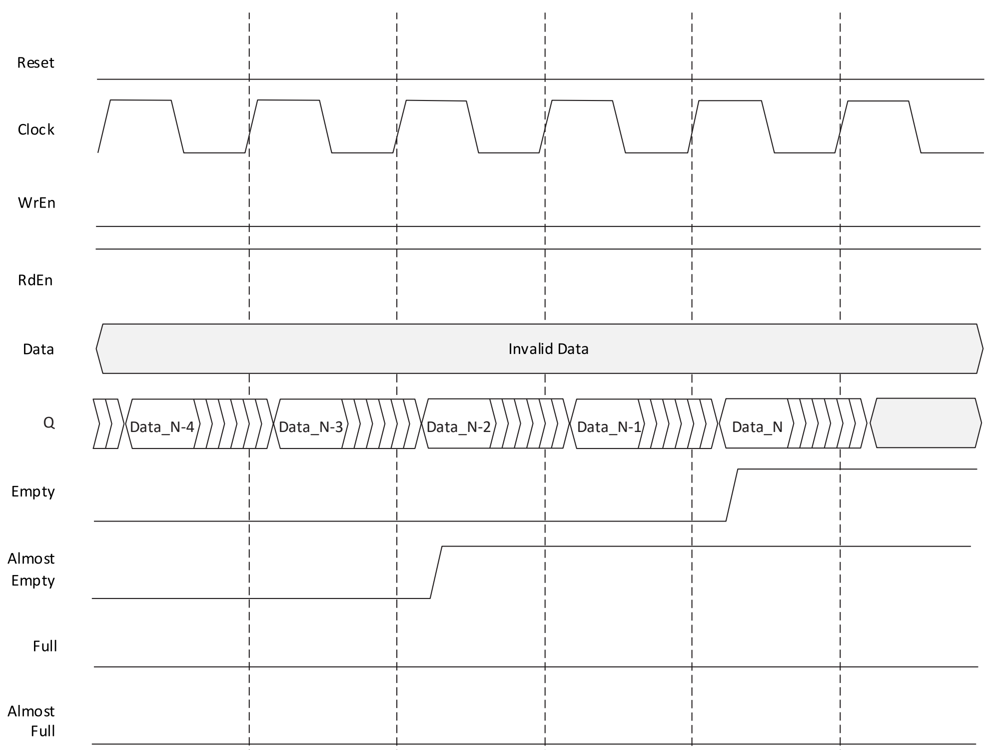

Figure 5.5. FIFO Without Output Registers – End of Data Read Cycle

Figure 5.2 to Figure 5.5 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 output registers and Figure 5.10 shows a FIFO with output registers 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.

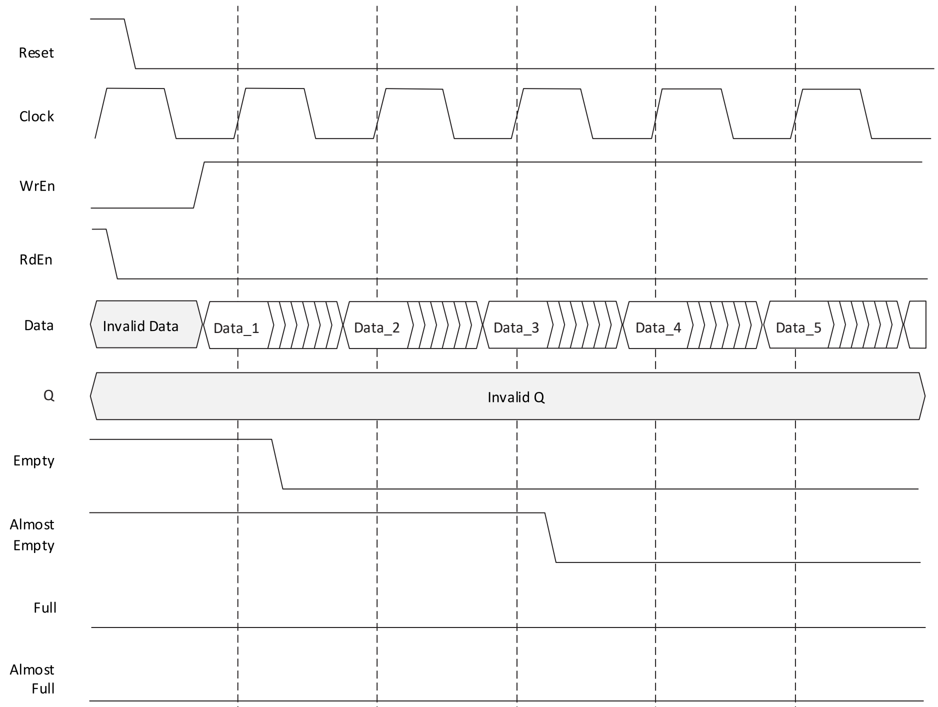

Figure 5.6. FIFO with Output Registers – Start of Data Write Cycle

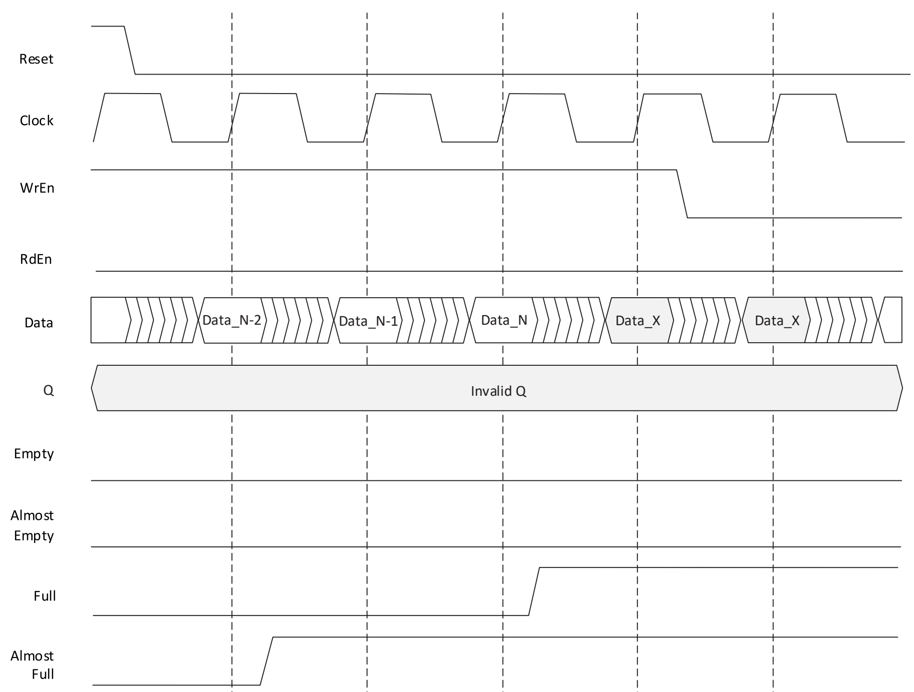

Figure 5.7. FIFO with Output Registers – End of Data Write Cycle

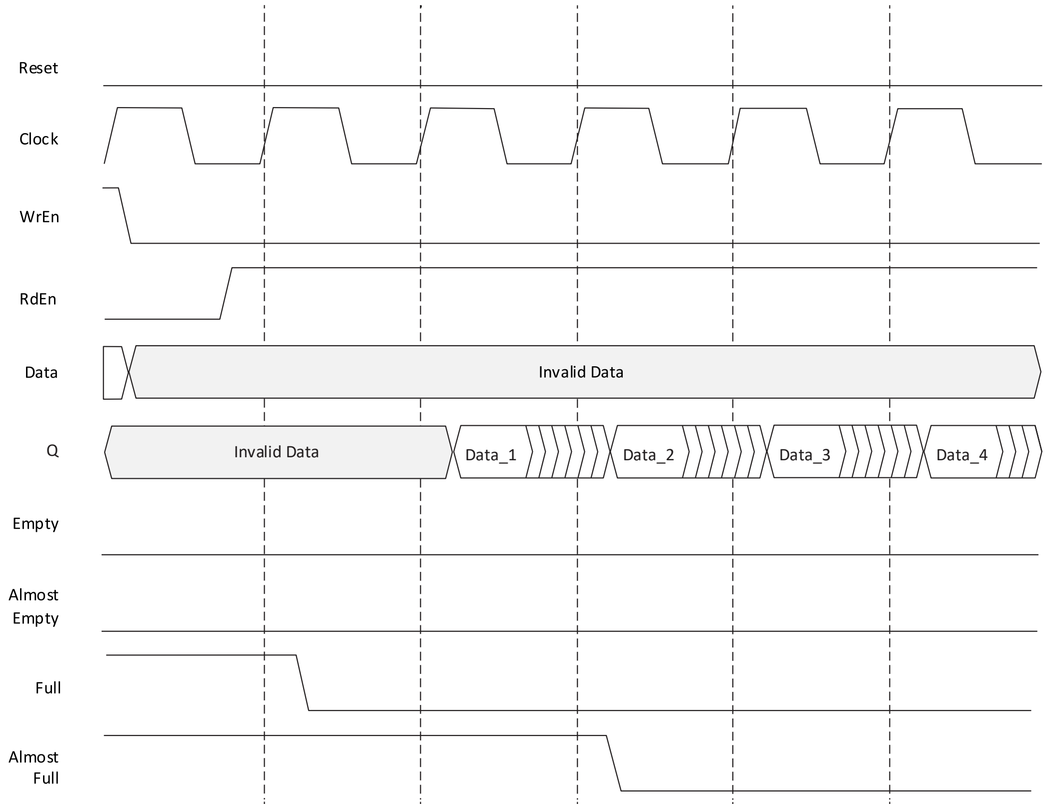

Figure 5.8. FIFO with Output Registers – Start of Data Read Cycle

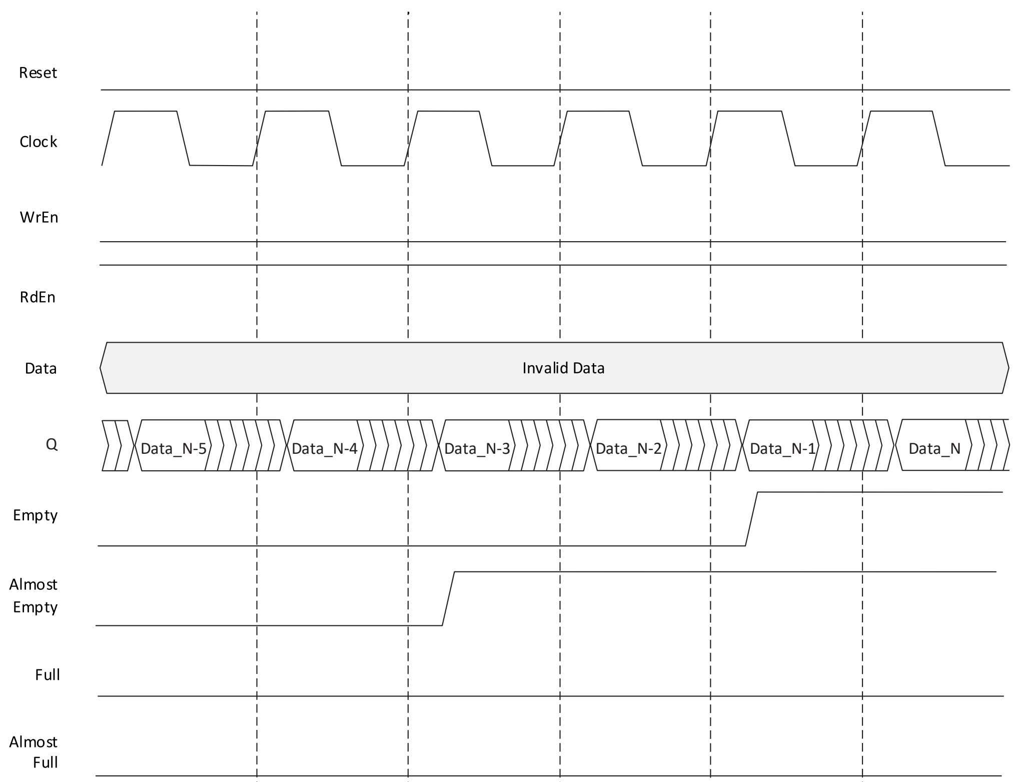

Figure 5.9. FIFO with Output Registers – End of Data Read Cycle

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

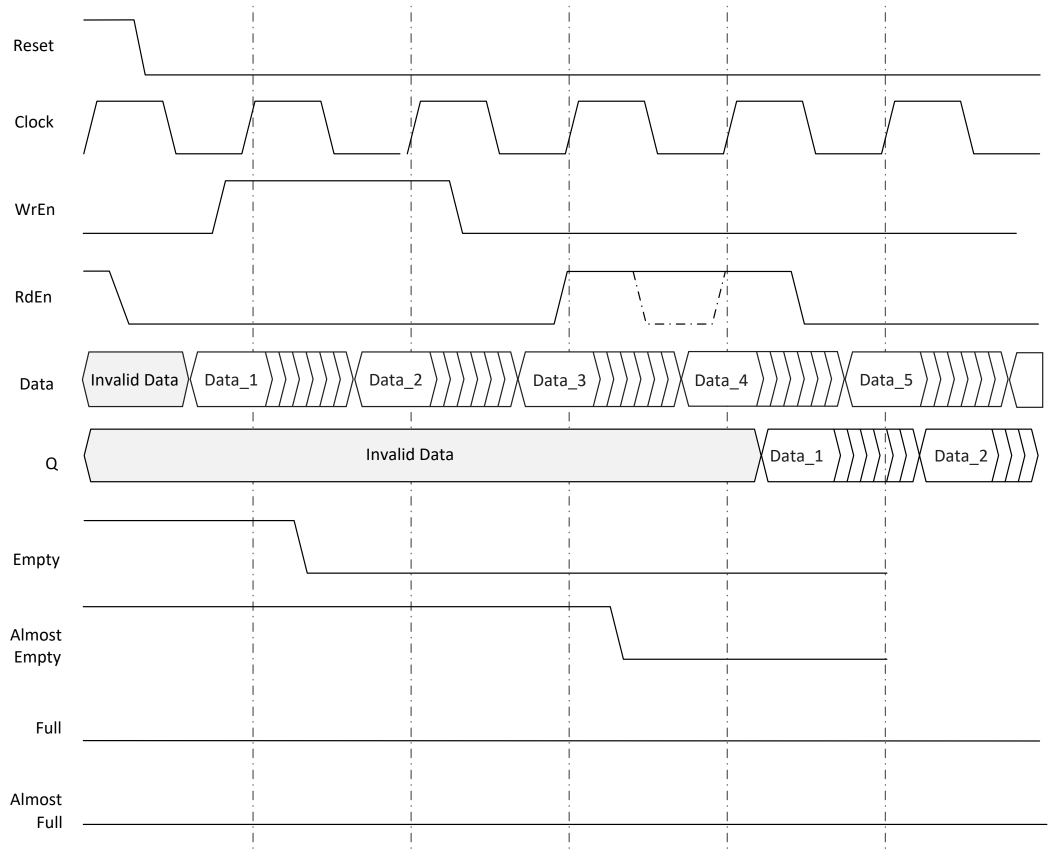

Figure 5.10. FIFO with Output Registers and RdEn on Output Registers

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.

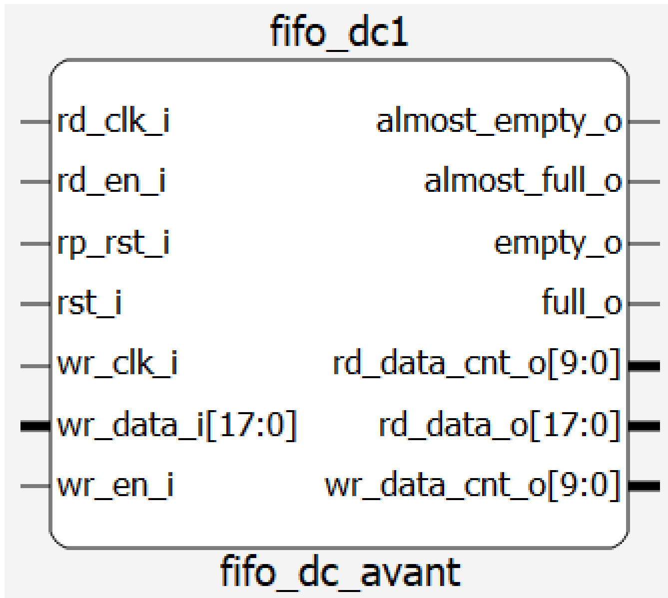

Figure 6.1. FIFO_DC Module Generated by IP Catalog

The various ports and 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 Lattice Avant Platform Devices

Configuration Tab Attributes	Description	Values	Default Value
Write Address Depth	Address depth of the write port Values are powers of two.	2–<Max that can fit in the device>	512
Write Data Width	Data word width write port	1–256	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 two.	2–<Max that can fit in the device>	512
Read Data Width	Data word width read port	1–256	18
Controller Implementation	FIFO Controller Implementation Option To enable the implementation type as LUT-based, or enable the Data Count output, you need to select LUT.	LUT, HW	HW
Enable FWFT	Enables the First Word Fall Through mode. This is only available in HW mode.	true, false	false
Implementation Type	EBR-based or LUT-based	EBR, LUT	EBR
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
Enable 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. 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 Assert Level	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 De-assert Level	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	2
Enable Almost Full Flag	Enables the generation of Almost Full Flag.	true, false	false
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. 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 Assert Level	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 De-assert Level	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	510
Enable Data Count (Synchronized to Write clock)	This options allows you to enable generation of write data count.	true, false	false
Enable 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.

Figure 6.2 and Figure 6.3 show 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 the FIFO.

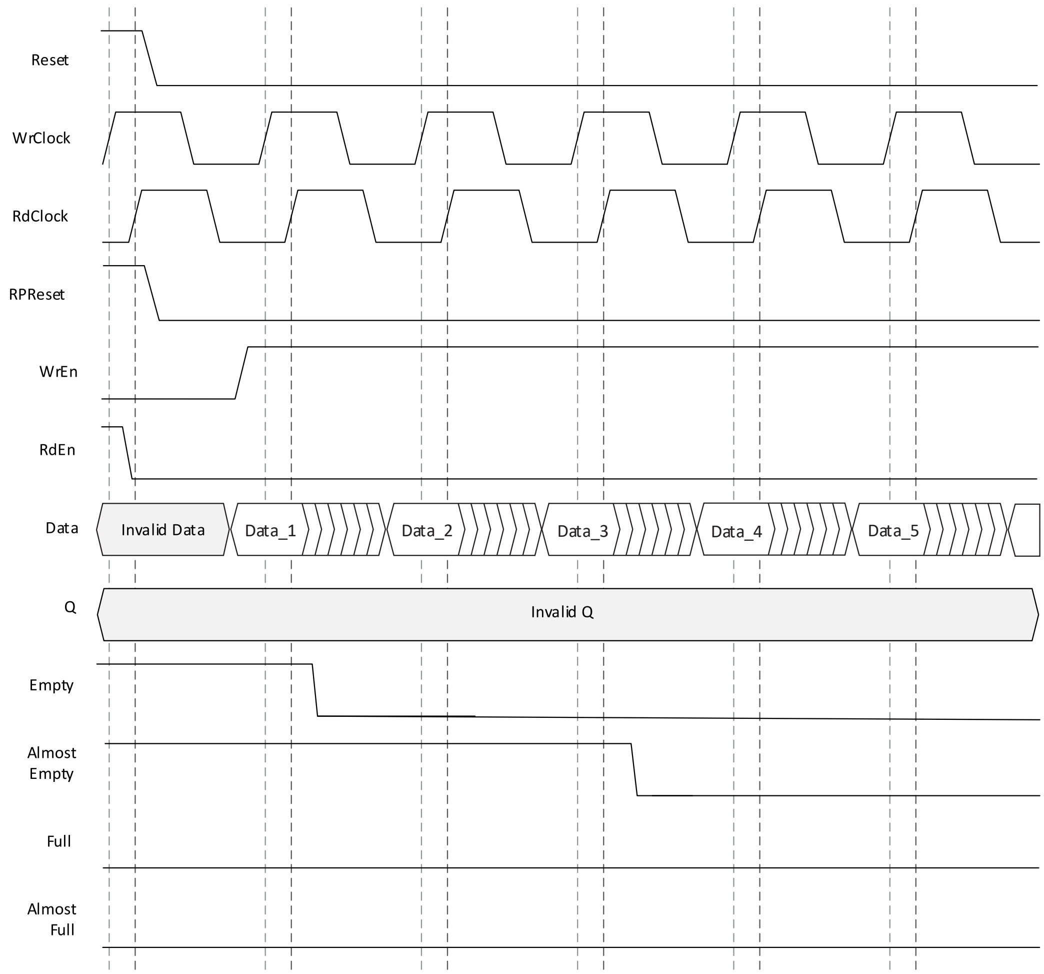

Figure 6.2. FIFO_DC Without Output Registers – Start of Data Write Cycle

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.

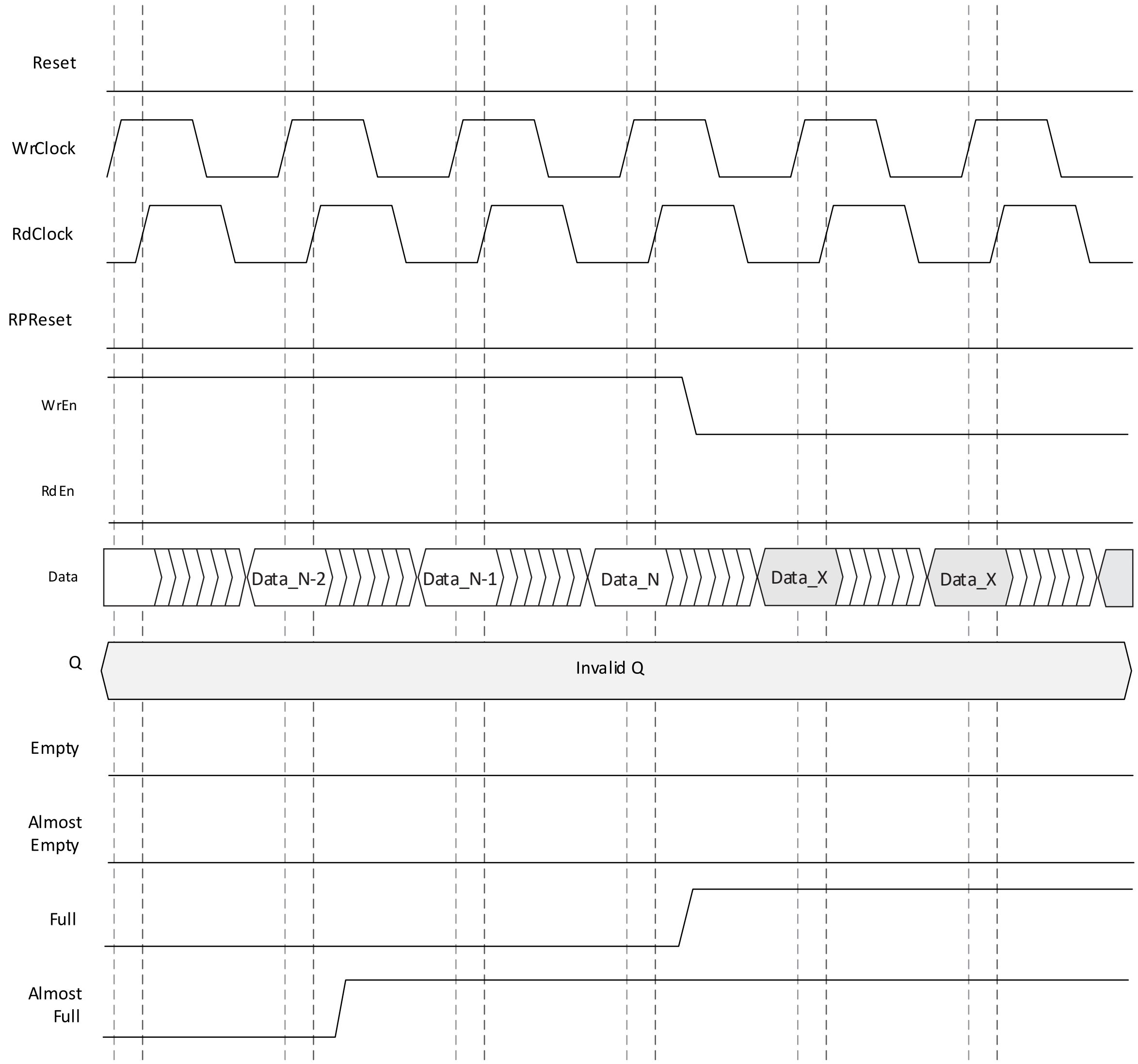

Figure 6.3. FIFO_DC Without Output Registers – End of Data Write Cycle

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 because the FIFO_DC is full, with the Full flag being 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 in Figure 6.4.

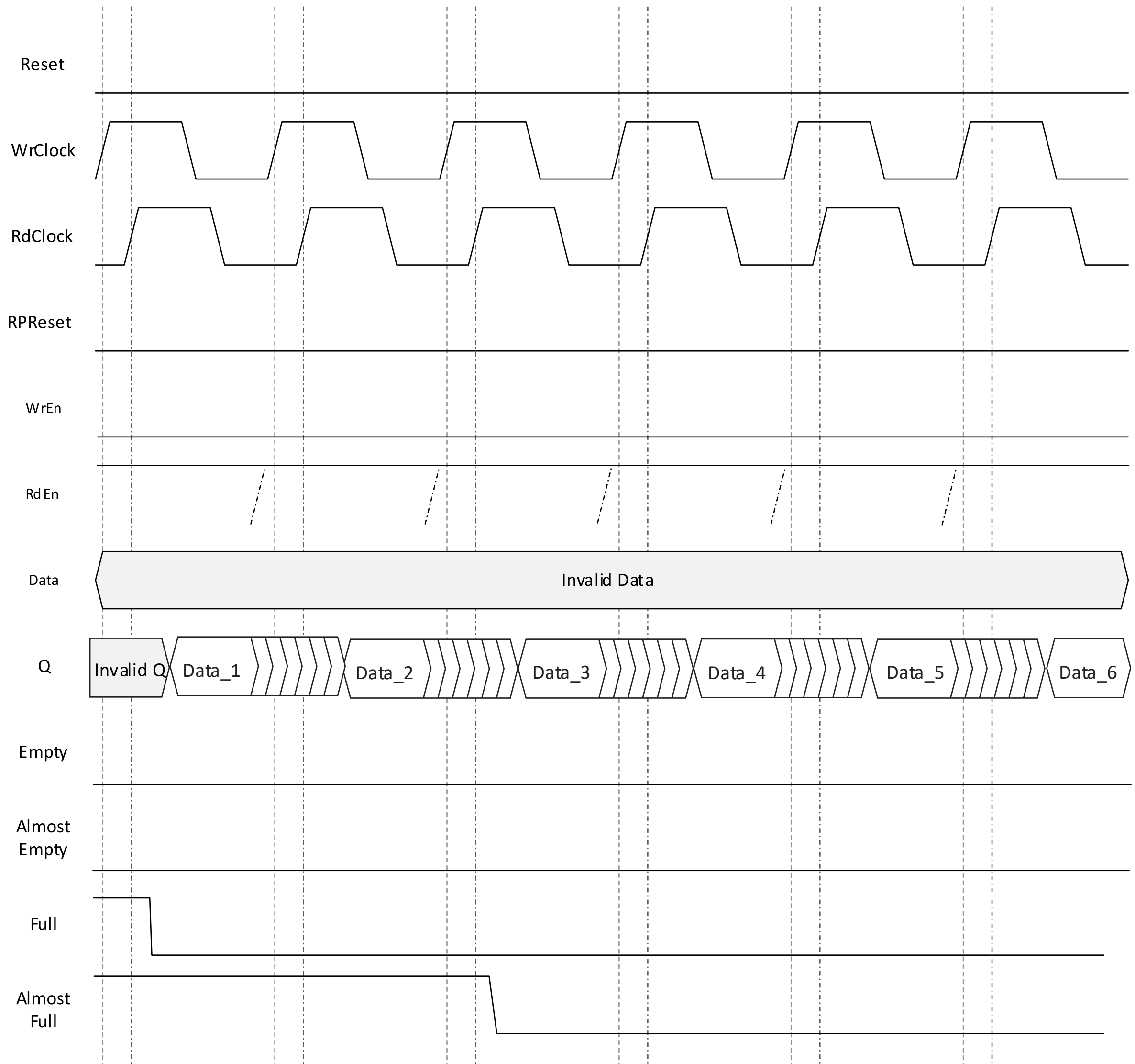

Figure 6.4. FIFO_DC Without Output Registers – Start of Data Read Cycle

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

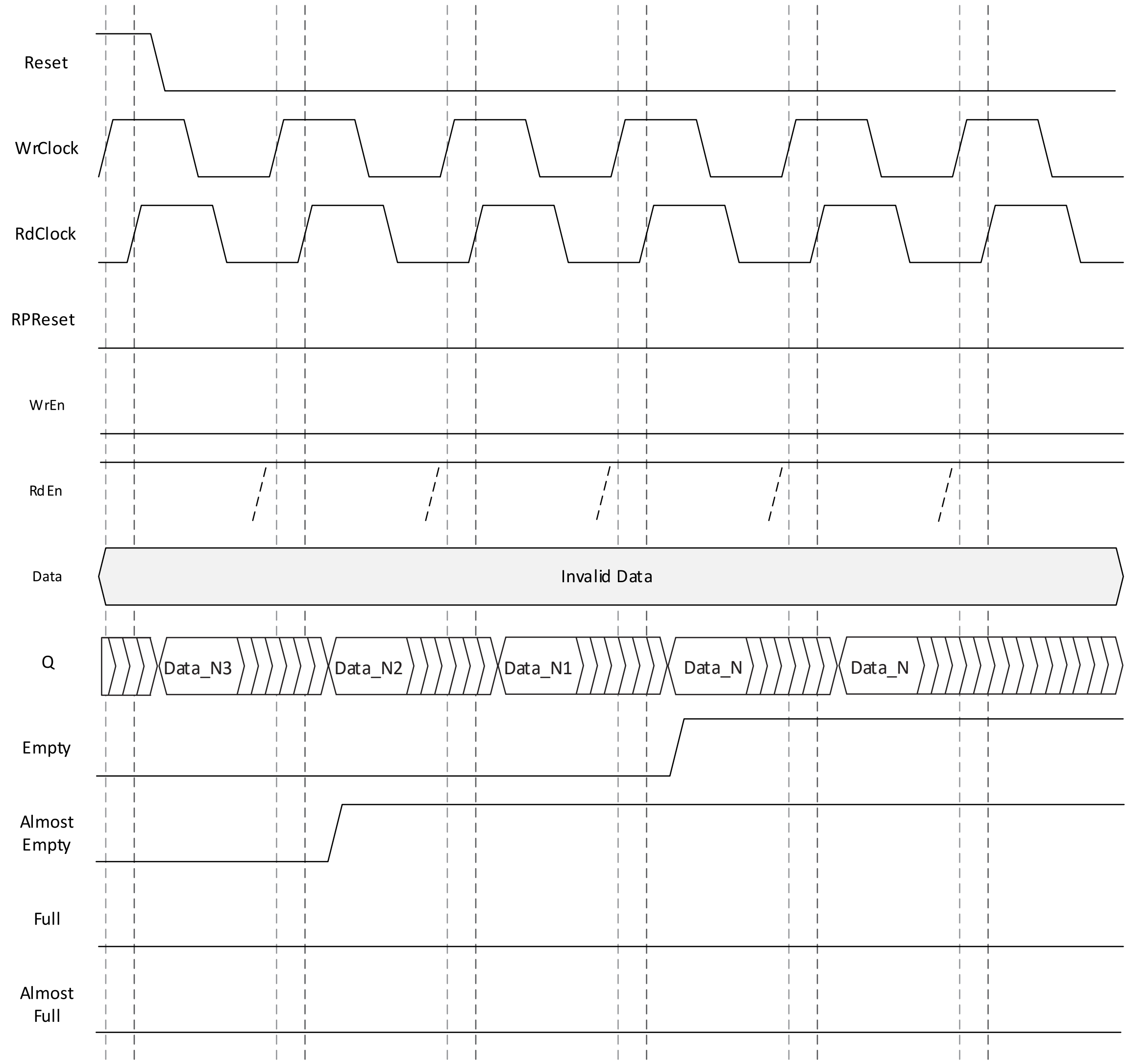

Figure 6.5. FIFO_DC Without Output Registers – End of Data Read Cycle

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 6.9 show similar waveforms for FIFO_DC with output registers and Figure 6.10 shows FIFO_DC with output registers enable by 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.

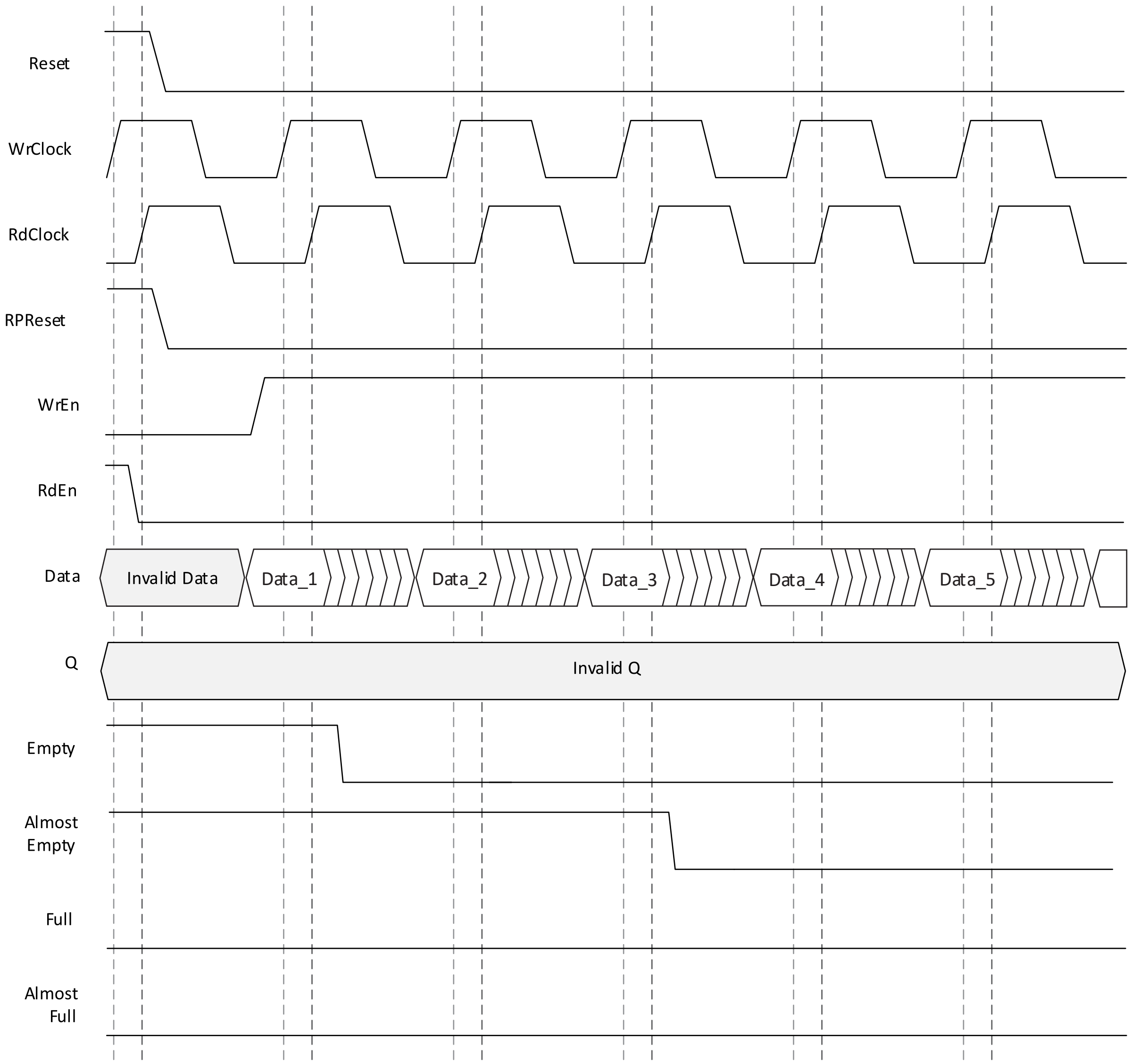

Figure 6.6. FIFO_DC with Output Registers – Start of Data Write Cycle

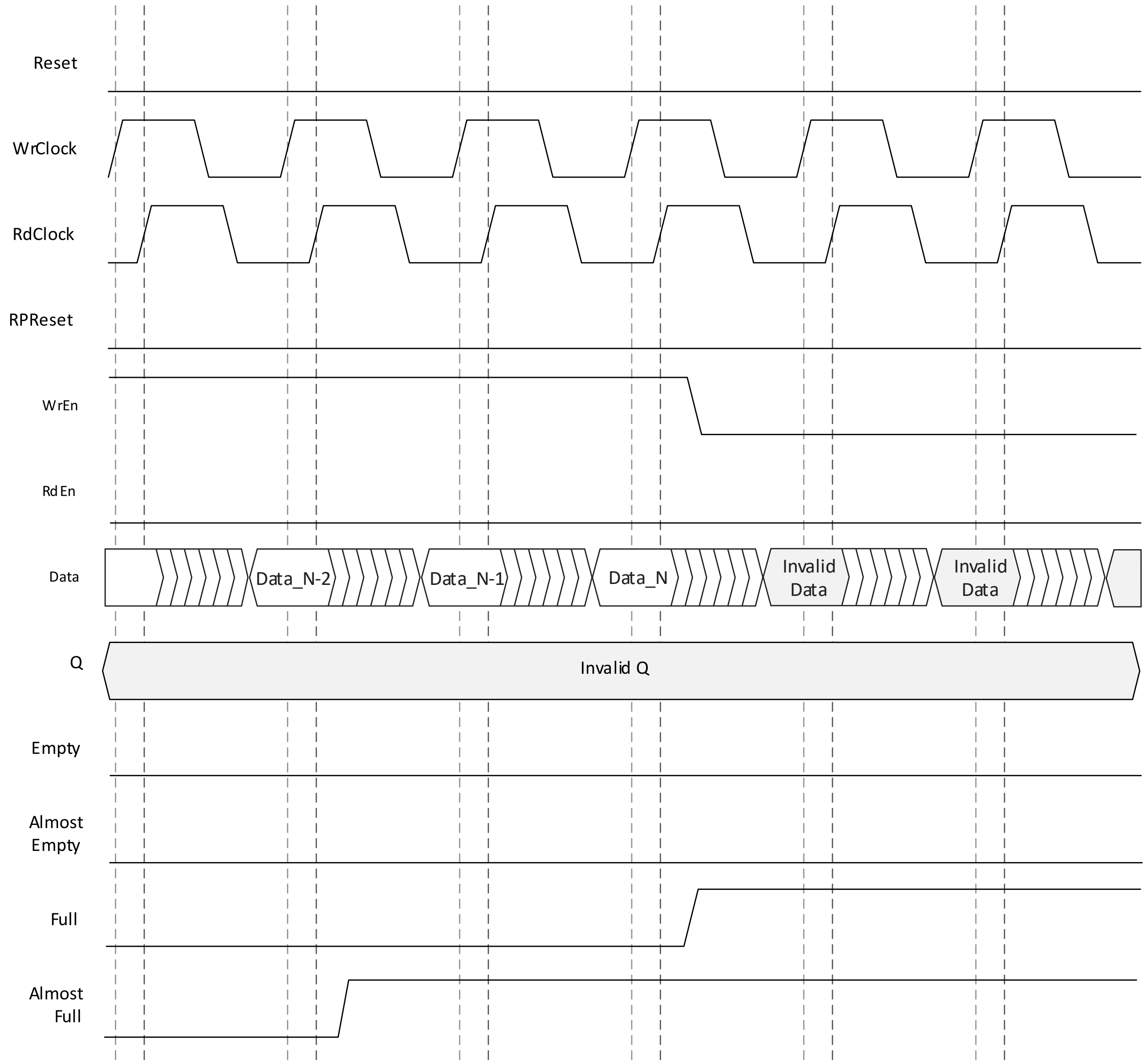

Figure 6.7. FIFO_DC with Output Registers – End of Data Write Cycle

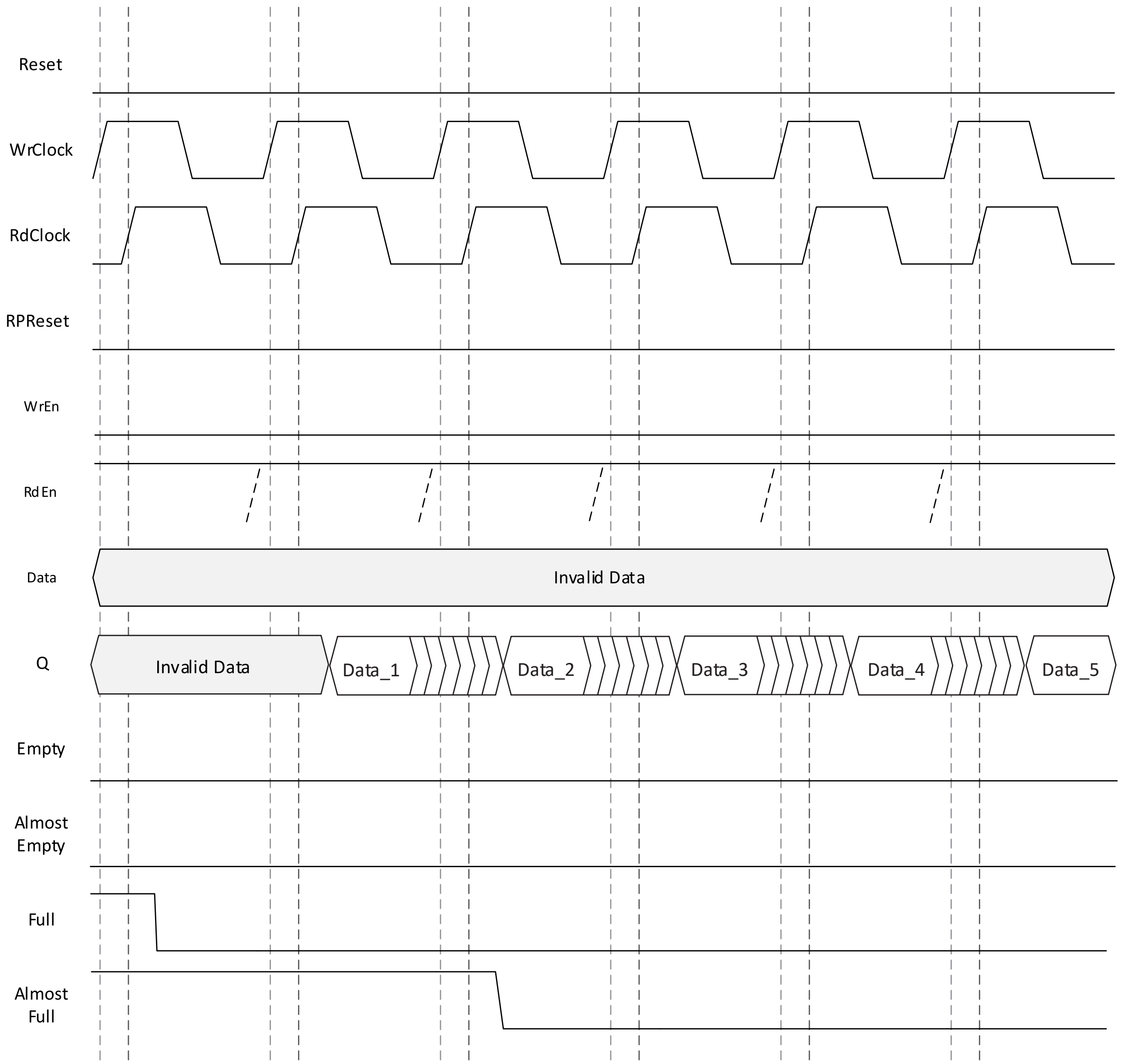

Figure 6.8. FIFO_DC with Output Registers – Start of Data Read Cycle

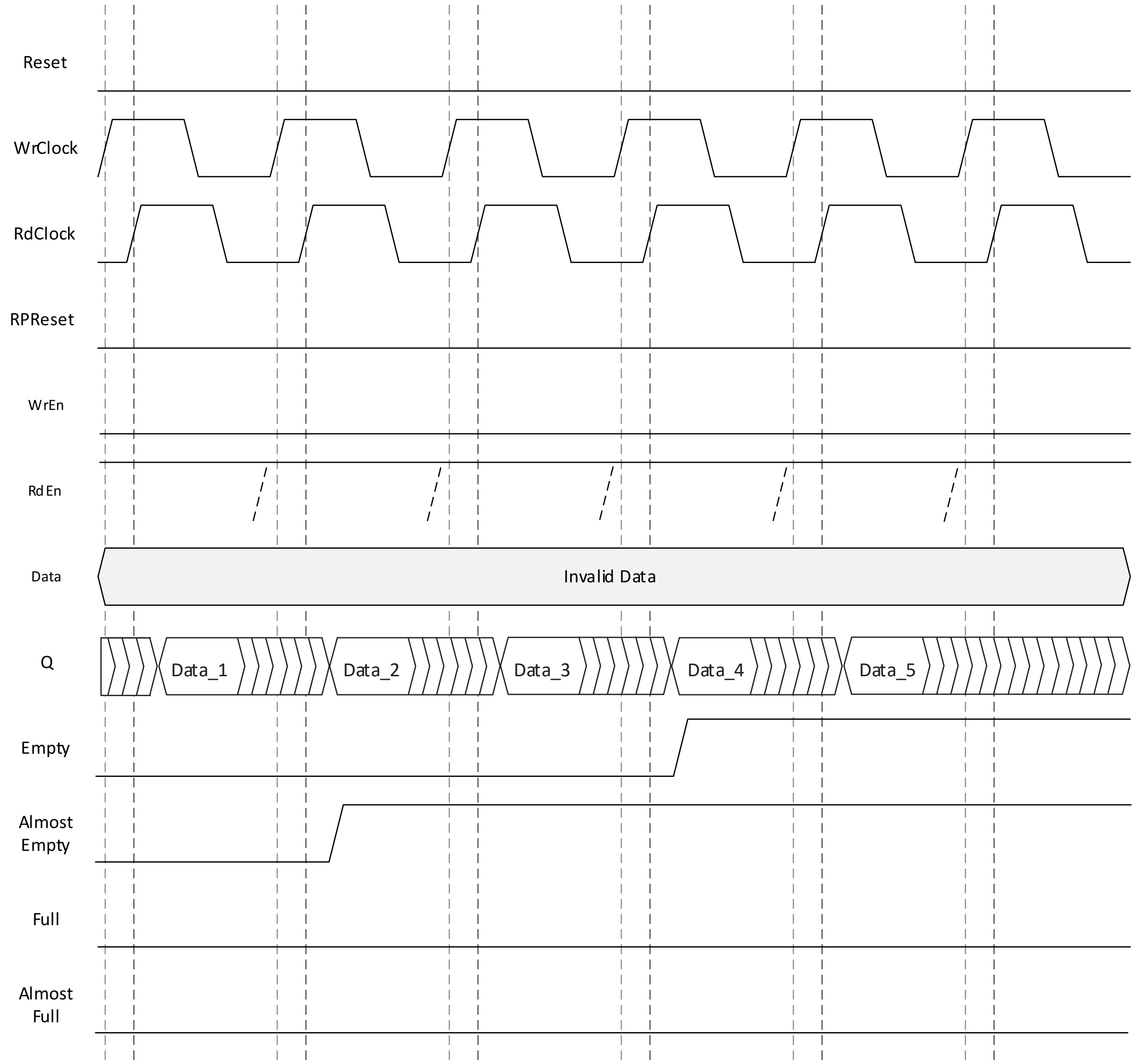

Figure 6.9. FIFO_DC with Output Registers – End of Data Read Cycle

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. The RdEn signal must be high during that clock cycle. Otherwise, the data takes an extra clock cycle when the RdEn signal goes true.

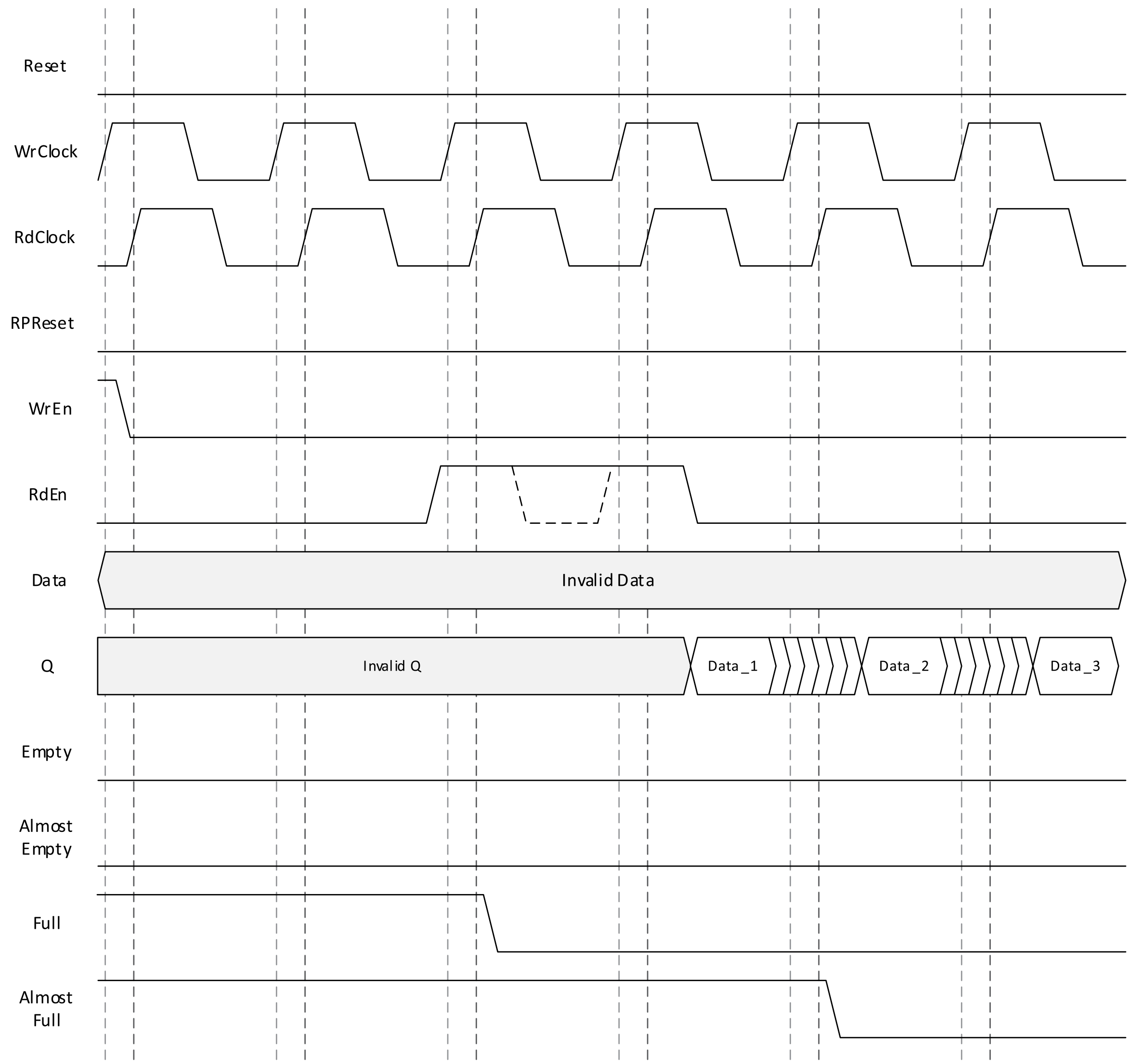

Figure 6.10. FIFO_DC with Output Registers and RdEn on Output Registers

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 must 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 generated by IP Catalog.

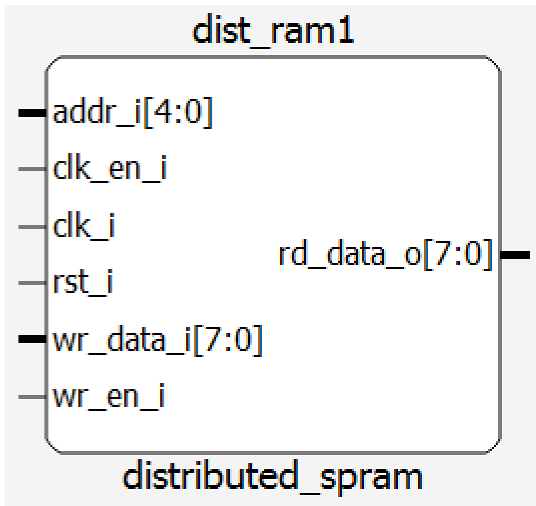

Figure 7.1. Distributed Single-Port RAM Module Generated by IP Catalog

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 want 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 SPR16X4A 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 the Lattice Radiant software installation folder.

It is to be noted that every three slices can accommodate 64 bits of memory. If the memory required is larger than 64 bits, then, cascading can be used. Furthermore, the ports can be registered using external PFU registers.

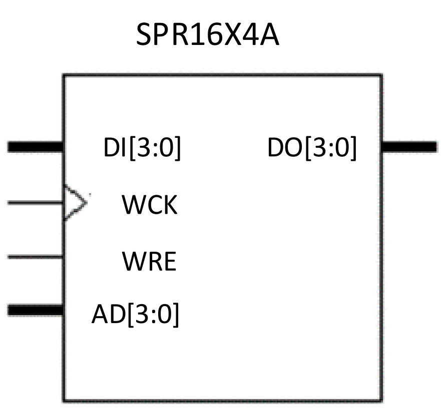

Figure 7.2. Single Port Distributed RAM Primitive for Lattice Avant Platform Devices

The various ports and definitions listed in Table 7.1. The table below 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 Lattice Avant 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
Reset Mode	Selection for the Reset to be synchronous or asynchronous to the Clock	async, sync	sync
Memory Initialization	Allows you to initialize the memories to all 1s, 0s, or use a custom initialization by providing a memory file.	none, 0s, 1s, memory file	none
Memory File	When a memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	—
Memory File Format	This option allows you to select if the memory file is formatted as binary or hex.	binary, hex	hex

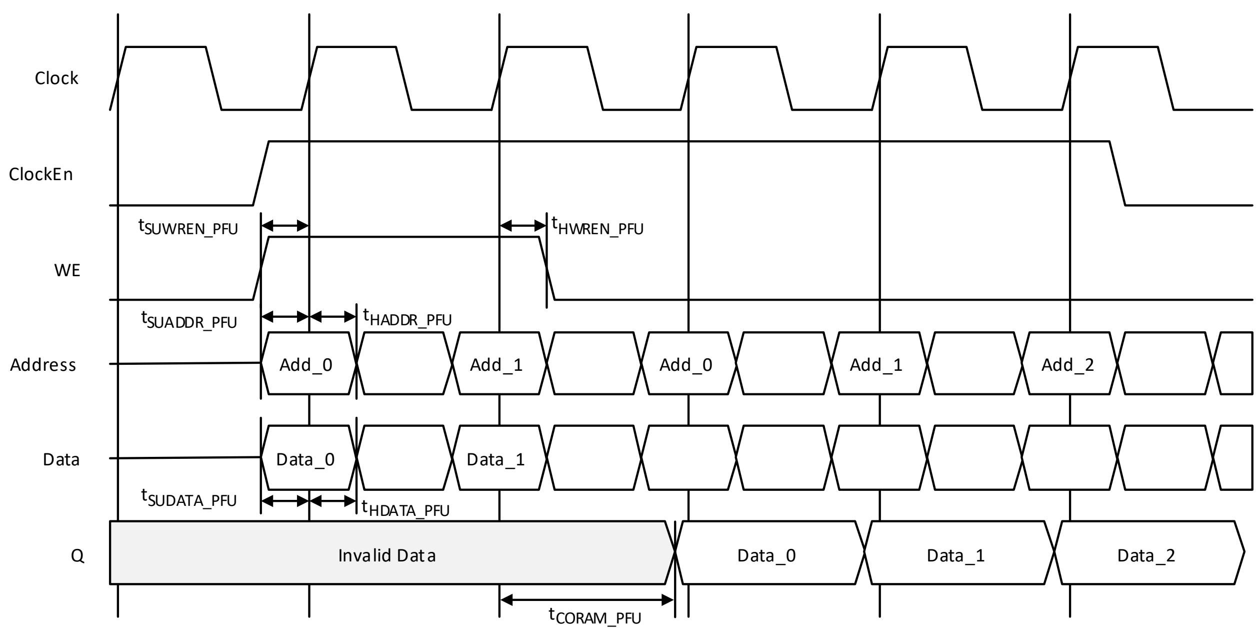

Figure 7.3. PFU-Based Distributed Single Port RAM Timing Waveform – without Output Registers

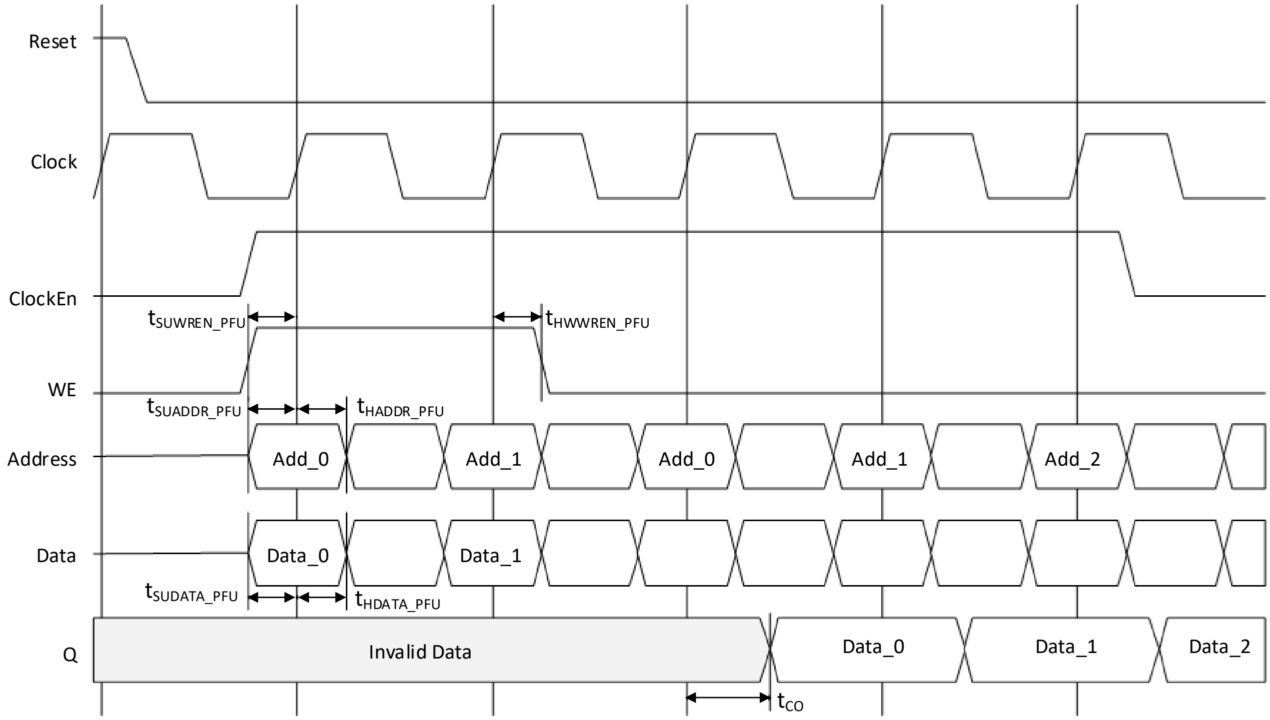

Figure 7.4. PFU-Based Distributed Single Port RAM Timing Waveform – with Output Registers

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.

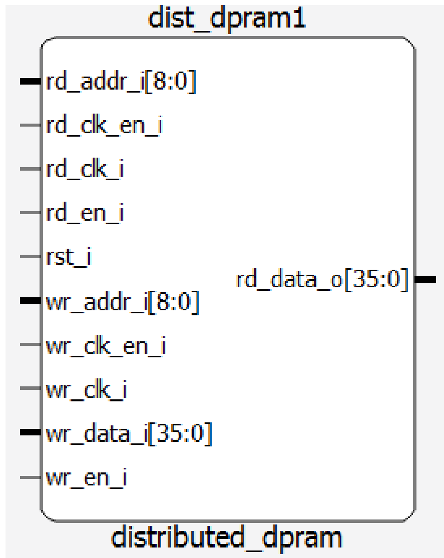

Figure 8.1. Distributed Dual-Port RAM Module Generated by IP Catalog

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 DPR16X4A 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 the Lattice Radiant software installation folder.

It is to be noted that every three slices can accommodate 64 bits of memory. If the memory required is larger than 64 bits, then, cascading can be used. Furthermore, the ports can be registered using external PFU registers.

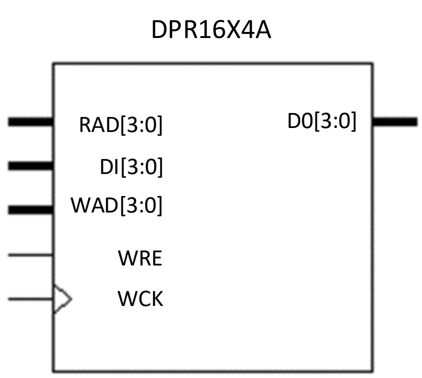

Figure 8.2. Dual Port Distributed RAM Primitive for Lattice Avant Platform Devices

The various ports and the 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 Lattice Avant 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	1–256	36
Enable Output Register	Data Out port (Q) can be registered or not using this selection.	true, false	true
Reset Mode	Selection for the Reset to be synchronous or asynchronous to the Clock	async, sync	sync
Memory Initialization	This option allows you to initialize memories to all 1s, 0s, or use a custom initialization by providing a memory file.	none, 0s, 1s, memory file	none
Memory File	When a memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	—
Memory File Format	This option allows you to select if the memory file is formatted as binary or hex.	binary, hex	hex

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) without and with output registers repsectively.

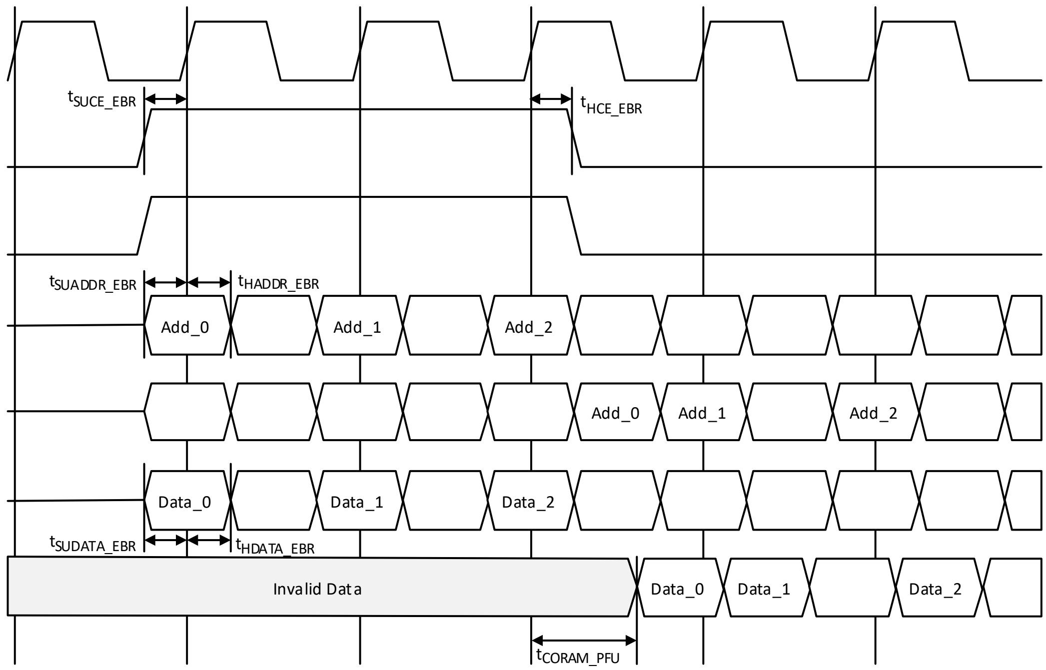

Figure 8.3. PFU-Based Distributed Dual Port RAM Timing Waveform – without Output Registers

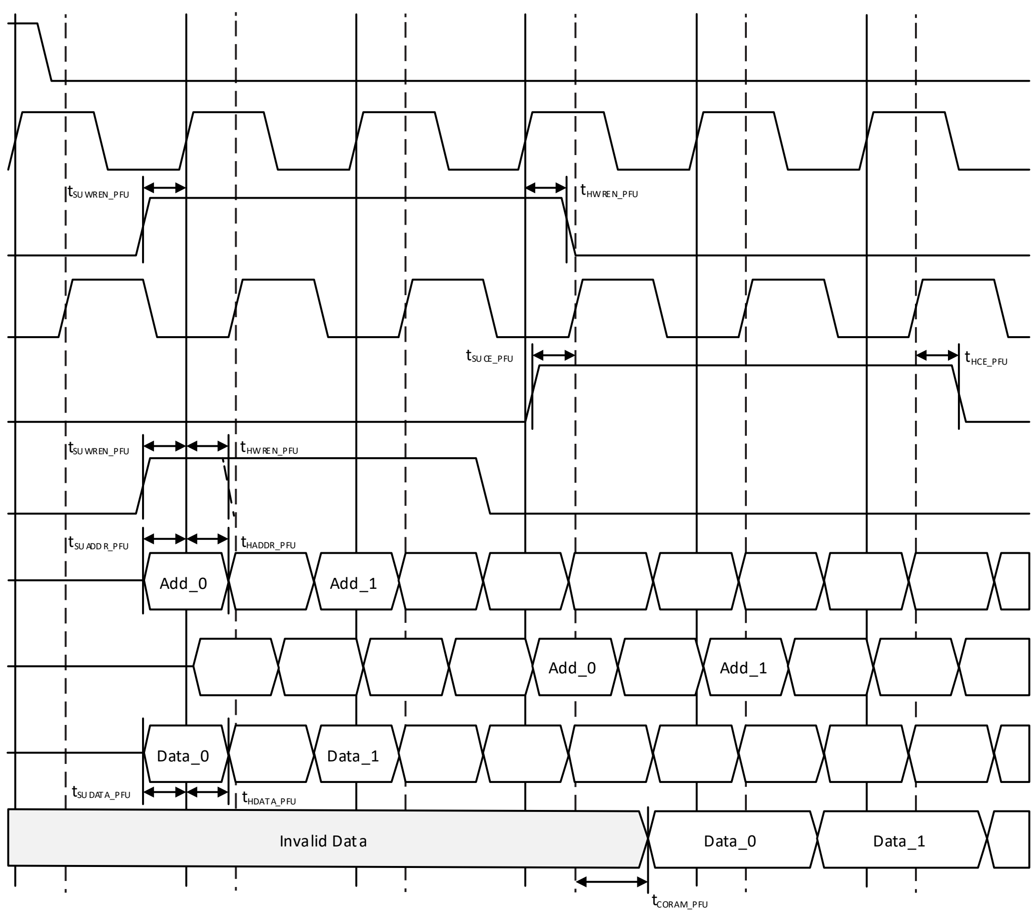

Figure 8.4. PFU-Based Distributed Dual Port RAM Timing Waveform – with Output Registers

9. Distributed ROM (Distributed_ROM) – PFU-Based

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

Figure 9.1 shows the Distributed ROM module generated by IP Catalog.

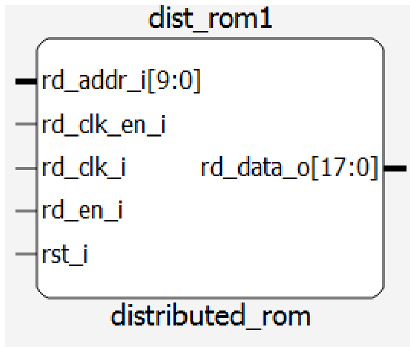

Figure 9.1. Distributed ROM Generated by IP Catalog

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. Furthermore, the ports can be registered by using external PFU registers.

The various ports and the definitions are listed in Table 9.1. The table below 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 Lattice Avant 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>	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
Reset Mode	Selection for the Reset to be synchronous or asynchronous to the Clock	async, sync	sync
Memory File	When a memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	—
Memory File Format	This option allows you to select if the memory file is formatted as binary or hex.	binary, hex	hex

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 without and with output registers respectively.

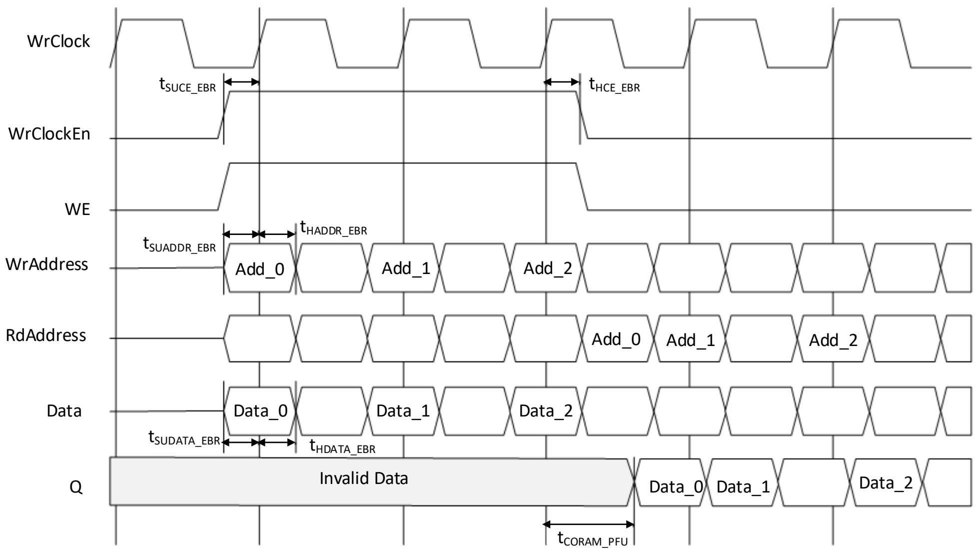

Figure 9.2. PFU-Based Distributed Dual Port RAM Timing Waveform – without Output Registers

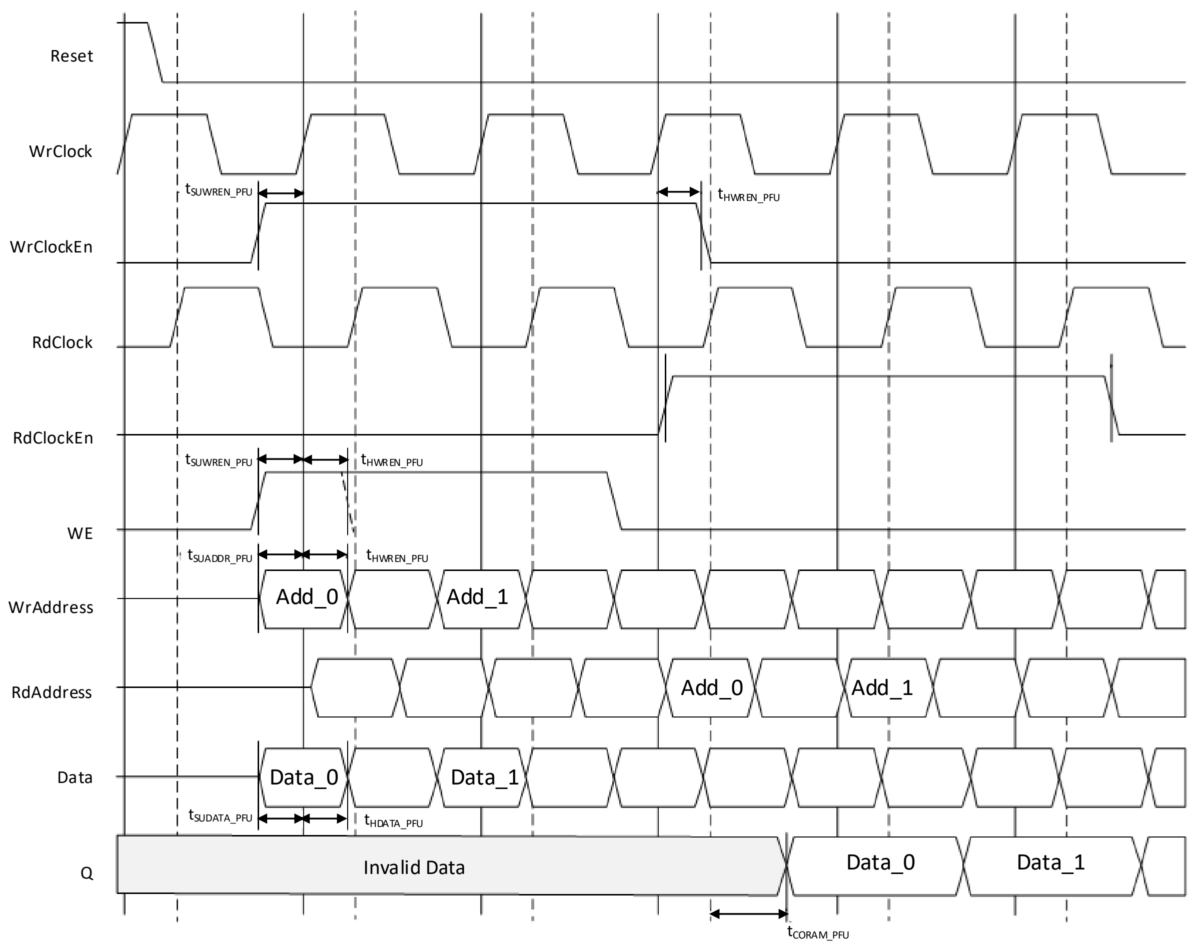

Figure 9.3. PFU-Based Distributed Dual Port RAM Timing Waveform – with Output Registers

10. Initializing Memory

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

10.1. Initialization File Formats

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

- Binary file
- Hex file

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 represent the number of bits for each address or the width of the memory module respectively.

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.

10.1.1. Binary File

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

Memory Size 20×32

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

10.1.2. Hex File

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

Memory Size 8×16

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

References

For more information, refer to:

- Avant-E web Page
- Avant-G web Page
- Avant-X web Page
- Lattice Avant Platform web page
- IP Cores and Reference Designs for Avant Devices
- Kits, Boards, and Demonstrations for Avant Devices
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 0.82, September 2024

Section	Change Summary
Disclaimers	Updated this section.
Inclusive Language	Added this section.
Memory Modules	Updated Figure 4.3. Single Port RAM Timing Waveform – without Output Registers to add the previously missing waveform for *WrEn*.
References	Added this section.

Revision 0.81, December 2022

Section	Change Summary
All	Minor adjustments in formatting across the document.
Memory Generation	Updated Utilizing PMI section content to change memory size to 1 kb and change attribute to syn_ramstyle and syn_romstyle.
Memory Features	• Updated ECC in Memory Modules section to add 8-bit ECC code. • Updated Byte Enable section to add info that byte-enable is only available if data width is greater than 9 bits.
Memory Modules	• Updated Reset section to add GSR signal information. • Updated Figure 4.2. Single Port RAM Primitive for Avant Platform Devices and Figure 4.6. True Dual Port RAM Primitive for Avant Platform Devices to add OCE and OCEA/OCEB pins respectively.

Revision 0.80, April 2022

Section	Change Summary
All	First preliminary release.

www.latticesemi.com