



Memory Modules - Lattice Radiant Software

User Guide

FPGA-IPUG-02033-2.5

June 2021

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Acronyms in This Document	8
1. Introduction	9
2. Memory Modules	10
2.1. Single Port RAM (RAM_DQ) – EBR Based	10
2.2. Pseudo Dual-Port RAM (RAM_DP) – EBR Based	13
2.3. True Dual-Port RAM (RAM_DP True) – EBR Based	16
2.4. Read Only Memory (ROM) – EBR Based	19
2.5. Pseudo Dual-Port RAM (Distributed_DPRAM) – LUT Based	21
2.6. Single Port RAM (Distributed_SPRAM) – LUT Based	24
2.7. Read Only Memory (Distributed ROM) – LUT Based	26
2.8. First in First Out Single Clock (FIFO)	28
2.9. First in First Out Dual Clock (FIFO_DC)	31
2.10. Shift Register (Shift_Register)	37
2.11. Single Port Large RAM (Large_RAM_SP)	39
2.11.1. Unaligned Access Feature	41
2.12. Pseudo Dual Port Large RAM (Large_RAM_DP)	43
2.12.1. Unaligned Access Feature	45
2.13. True Dual-Port Large RAM (Large_RAM_DP True)	46
2.13.1. Unaligned Access Feature	48
2.14. Large Read-Only Memory (Large_ROM)	51
2.14.1. Unaligned Access Feature	52
3. IP Generation	54
4. PMI Support	56
4.1. pmi_ram_dq	56
4.2. pmi_ram_dq_be	59
4.3. pmi_ram_dp	62
4.4. pmi_ram_dp_be	65
4.5. pmi_rom	68
4.6. pmi_ram_dp_true	70
4.7. pmi_distributed_spram	74
4.8. pmi_distributed_dpram	76
4.9. pmi_distributed_rom	78
4.10. pmi_fifo	80
4.11. pmi_fifo_dc	83
4.12. pmi_distributed_shift_reg	86
5. Initializing Memory	88
5.1. Initialization File Formats	88
5.2. Binary File	88
5.3. Hex File	89
6. Simulation of Memory Modules	90
Appendix A. Resource Utilization	95
Appendix B. Limitations	97
Technical Support Assistance	98
Revision History	99

Figures

Figure 2.1. Single-Port Memory Module Generated by Module/IP Block Wizard	10
Figure 2.2. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, Without Output Registers	12
Figure 2.3. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, With Output Registers	12
Figure 2.4. Pseudo Dual-Port Memory Module Generated by Module/IP Block Wizard	13
Figure 2.5. PSEUDO DUAL PORT RAM Timing Diagram, Without Output Registers	15
Figure 2.6. PSEUDO DUAL PORT RAM Timing Diagram, With Output Registers	15
Figure 2.7. PSEUDO DUAL PORT RAM Timing Diagram, Mixed-Width Without Output Registers	15
Figure 2.8. True Dual-Port Memory Module Generated by Module/IP Block Wizard	16
Figure 2.9. TRUE DUAL PORT RAM Timing Diagram, Without Output Registers	18
Figure 2.10. TRUE DUAL PORT RAM Timing Diagram, With Output Registers	18
Figure 2.11. TRUE DUAL PORT RAM Timing Diagram, Mixed Width Without Output Registers	18
Figure 2.12. Read Only Memory Module Generated by Module/IP Block Wizard	19
Figure 2.13. ROM Timing Diagram, Without Output Registers	20
Figure 2.14. ROM Timing Diagram, With Output Registers	20
Figure 2.15. Distributed Pseudo Dual Port Module Generated by Module/IP Block Wizard	21
Figure 2.16. Distributed PSEUDO DUAL PORT RAM Timing Diagram, without Output Registers	22
Figure 2.17. Distributed PSEUDO DUAL PORT RAM Timing Diagram, With Output Registers	23
Figure 2.18. Distributed Single Port Module Generated by Module/IP Block Wizard	24
Figure 2.19. Distributed Single Port RAM Timing Waveform in Normal (NORMAL) Mode, Without Output Registers	25
Figure 2.20. Distributed Single Port RAM Timing Waveform in Normal (NORMAL) Mode, With Output Registers	25
Figure 2.21. Distributed Read Only Memory Module Generated by Module/IP Block Wizard	26
Figure 2.22. Distributed ROM Timing Diagram, Without Output Registers	27
Figure 2.23. Distributed ROM Timing Diagram, With Output Registers	27
Figure 2.24. Distributed Pseudo Dual Port Module Generated by Module/IP Block Wizard	28
Figure 2.25. FIFO Single Clock Without Registers	30
Figure 2.26. FIFO Single Clock With Registers	30
Figure 2.27. FIFO Dual Clock Module Generated by Module/IP Block Wizard	31
Figure 2.28. FIFO Dual Clock Module Without registers	35
Figure 2.29. FIFO Dual Clock Module With Registers	35
Figure 2.30. FIFO Dual Clock Module, Mixed-Width Without registers	35
Figure 2.31. Hard Controller FIFO Dual Clock Module Without registers	35
Figure 2.32. Hard Controller FIFO Dual Clock Module Without registers operating in asynchronous clocks.	36
Figure 2.33. Shift Register Module Generated by Module/IP Block Wizard	37
Figure 2.34. Shift Register Without Output Register	38
Figure 2.35. Shift Register With Output Register	38
Figure 2.36. Shift Register Variable Configuration and Without Output Register	38
Figure 2.37. Shift Register Without Output Register (LUT Memory, Lattice Nexus Devices)	38
Figure 2.38. Shift Register Without Output Register (LUT Memory, iCE40 UltraPlus)	38
Figure 2.39. Single-Port Large RAM Module Generated by Module/IP Block Wizard	39
Figure 2.40. Single Port LRAM Timing Waveform in NORMAL Mode, Without Output Registers	41
Figure 2.41. Single Port RAM Timing Waveform in NORMAL Mode, With Output Registers	41
Figure 2.42. Single Port LRAM Timing Waveform in Write-Through Mode, Without Output Registers	42
Figure 2.43. Single Port RAM Timing Waveform in Read-Before-Write Mode, Without Output Registers	42
Figure 2.44. Pseudo Dual-Port Large RAM Module Generated by Module/IP Block Wizard	43
Figure 2.45. PSEUDO DUAL PORT Large RAM Timing Diagram, Without Output Registers	45
Figure 2.46. PSEUDO DUAL PORT Large RAM Timing Diagram, With Output Registers	45
Figure 2.47. True Dual-Port Large RAM Module Generated by Module/IP Block Wizard	46
Figure 2.48. Single Port LRAM Timing Waveform in NORMAL Mode, Without Output Registers	49
Figure 2.49. Single Port RAM Timing Waveform in NORMAL Mode, With Output Registers	49
Figure 2.50. Single Port LRAM Timing Waveform in Write-Through Mode, Without Output Registers	50
Figure 2.51. Large Read Only Memory Module Generated by Module/IP Block Wizard	51
Figure 2.52. Large ROM Timing Diagram, Without Output Registers	53

Figure 2.53. Large ROM Timing Diagram, With Output Registers	53
Figure 3.1. MemoryModules Under Module/IP on Local in the Lattice Radiant Software.....	54
Figure 3.2. Example: Generating Pseudo Dual Port RAM (RAM_DP) Using Module/IP Block Wizard.....	55
Figure 3.3. Example: Generating Pseudo Dual Port RAM (RAM_DP) Module Customization	55
Figure 6.1. Project with Instance of Pseudo Dual Port Memory.....	90
Figure 6.2. File Selection Window	91
Figure 6.3. Top Level Testbench Instance in Project.....	91
Figure 6.4. Simulation Wizard.....	92
Figure 6.5. Final Simulation Files	92
Figure 6.6. File Parsing and Top Module Selection	93
Figure 6.7. Active-HDL Initial Simulation	93
Figure 6.8. Active-HDL Final Simulation	94

Tables

Table 1.1. Memory Module Support	9
Table 2.1. EBR-Based Single-Port Memory Port Definitions	10
Table 2.2. EBR-Based Single-Port Memory Attribute Definitions	11
Table 2.3. EBR-Based Pseudo Dual-Port Memory Port Definitions	13
Table 2.4. EBR-Based Pseudo Dual-Port Memory Attribute Definitions.....	14
Table 2.5. EBR-Based True Dual-Port Memory Port Definitions.....	16
Table 2.6. EBR-Based True Dual-Port Memory Attribute Definitions	17
Table 2.7. EBR-Based Read Only Memory Port Definitions	19
Table 2.8. EBR-Based Read Only Memory Attribute Definitions	20
Table 2.9. LUT-Based Pseudo Dual-Port Memory Port Definitions.....	21
Table 2.10. LUT-Based Pseudo Dual-Port Memory Attribute Definitions.....	22
Table 2.11. LUT-Based Single-Port Memory Port Definitions	24
Table 2.12. LUT-Based Single-Port Memory Attribute Definitions	25
Table 2.13. LUT-Based Read Only Memory Port Definitions	26
Table 2.14. LUT-Based Read Only Memory Attribute Definitions	27
Table 2.15. First in First Out Single Clock Port Definitions.....	28
Table 2.16. First in First Out Single Clock Attribute Definitions.....	29
Table 2.17. First in First Out Dual Clock Port Definitions.....	31
Table 2.18. First in First Out Dual Clock Attribute Definitions	32
Table 2.19. Shift Register Port Definitions	37
Table 2.20. Shift Register Attribute Definitions	37
Table 2.21. Large-RAM based Single-Port Memory Port Definitions.....	39
Table 2.22. Large RAM-Based Single-Port Memory Attribute Definitions.....	40
Table 2.23. Unaligned Read shift function (Single Port LRAM).....	41
Table 2.24. Large RAM-Based Pseudo Dual-Port Memory Port Definitions	43
Table 2.25. Large RAM-Based Pseudo Dual-Port Memory Attribute Definitions	44
Table 2.26. Unaligned Read shift function (Pseudo Dual Port LRAM)	45
Table 2.27. Large RAM-Based True Dual-Port Memory Port Definitions.....	46
Table 2.28. Large RAM-Based True Dual-Port Memory Attribute Definitions.....	47
Table 2.29. Unaligned Read shift function (True Dual Port LRAM).....	48
Table 2.30. Large Read Only Memory Port Definitions	51
Table 2.31. Large Read Only Memory Attribute Definitions.....	52
Table 2.32. Unaligned Read Shift Function (LROM).....	52
Table 4.1. Single Port PMI Port Definitions.....	56
Table 4.2. Single Port PMI Attribute Definitions.....	56
Table 4.3. Single Port with Byte-Enable PMI Port Definitions	59
Table 4.4. Single Port with Byte-Enable PMI Attribute Definitions	59
Table 4.5. Pseudo Dual Port PMI Port Definitions	62
Table 4.6. Pseudo Dual Port PMI Attribute Definitions	62
Table 4.7. Pseudo Dual Port with Byte Enable PMI Port Definitions	65
Table 4.8. Pseudo Dual Port with Byte Enable PMI Attribute Definitions	65
Table 4.9. Read Only Memory PMI Port Definitions.....	68
Table 4.10. Read Only Memory PMI Attribute Definitions	68
Table 4.11. True Dual Port PMI Port Definitions.....	70
Table 4.12. True Dual Port PMI Attribute Definitions.....	71
Table 4.13. Single Port – Distributed Memory PMI Port Definitions	74
Table 4.14. Single Port – Distributed Memory PMI Attribute Definitions	74
Table 4.15. Distributed Pseudo Dual Port PMI Port Definitions	76
Table 4.16. Distributed Pseudo Dual Port PMI Attribute Definitions	76
Table 4.17. Read Only Memory PMI Port Definitions.....	78
Table 4.18. Read Only PMI Memory Attribute Definitions	78
Table 4.19. FIFO Single Clock PMI Port Definitions.....	80

Table 4.20. FIFO Single Clock PMI Attribute Definitions	80
Table 4.21. FIFO Dual Clock PMI Port Definitions	83
Table 4.22. FIFO Dual Clock Port PMI Attribute Definitions	83
Table 4.23. Shift Register PMI Port Definitions.....	86
Table 4.24. Shift Register PMI Attribute Definitions.....	86
Table 6.1. File List	90
Table A.1. Device and Tool Tested.....	95
Table A.2. EBR-Based Single-Port Memory Utilization	95
Table A.3. EBR-Based Pseudo Dual-Port Memory Utilization	95
Table A.4. EBR-Based True Dual-Port Memory Utilization	95
Table A.5. EBR-Based Read-Only Memory Utilization	95
Table A.6. Distributed Single-Port Memory Utilization	95
Table A.7. Distributed Pseudo Dual-Port Memory Utilization	96
Table A.8. Distributed Read Only Memory Utilization.....	96
Table A.9. Single Clock First In First Out Utilization	96
Table A.10. Dual Clock First In First Out Utilization	96
Table A.11. Shift Register Utilization	96

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
EBR	Embedded Block Random Access Memory
FIFO	First In First Out
FPGA	Field-Programmable Gate Array
IP	Intellectual Property
LUT	Lookup-Table
PMI	Parameterized Module Instantiation
RTL	Register Transfer Level

1. Introduction

This technical note discusses memory usage for the FPGA devices supported by Lattice Radiant™ Software. It is intended to be used by design engineers as a guide to integrating memory blocks for all device families in Lattice Radiant Software.

In addition, behavioral inference for custom memory supported and the Synthesis tool is expected to infer the required components based on synthesis directive.

Designers can utilize the memory primitives using two different methods described below.

- Using Module/IP Block Wizard – The Module/IP Block Wizard user interface allows you to specify the required memory type and size. Module/IP Block Wizard takes this specification and instantiates a synthesizable RTL (register transfer level) code and sets the appropriate parameters based on the user interface setting.
- Using PMI (Parameterized Module Instantiation) – PMI allows experienced users to skip the graphical user interface and utilize the configurable memory primitives on-the-fly from the Lattice Radiant Software project navigator. You set the parameters and the control signals needed in Verilog.

Table 1.1. Memory Module Support

Module Name	iCE40 UltraPlus™	Lattice FPGA Devices Built on the Lattice Nexus™ Platform	PMI Support
Single Port Memory (RAM_DQ)	✓	✓	✓
Pseudo Dual Port Memory (RAM_DP)	✓	✓	✓
True Dual Port Memory (RAM_DP True)	✓	✓	✓
Read Only Memory (ROM)	✓	✓	✓
Pseudo Dual-Port Distributed Memory (Distributed DPRAM)	—	✓	✓*
Single-Port Distributed Memory (Distributed SPRAM)	—	✓	✓*
Read Only Distributed Memory (Distributed ROM)	—	✓	✓*
First In First Out Single Clock (FIFO)	✓	✓	✓
First In First Out Dual Clock (FIFO_DC)	✓	✓	✓
Shift Register (Shift_Reg)	✓	✓	✓*
Single Port Large RAM (Large RAM_DQ)	—	✓	—
Pseudo Dual Port Large RAM (Large RAM_DP)	—	✓	—
True Dual Port Large RAM (Large RAM_DP True)	—	✓	—
Large Read Only Memory (Large ROM)	—	✓	—

***Note:** Available only on Lattice FPGA devices built on the Lattice Nexus platform.

2. Memory Modules

The following sections discuss the different memory modules available from the IP Catalog, the size of memory that each module can support, and other special options for the module. Module/IP Block Wizard automatically allows you to create memories larger than the width and depth supported for each memory primitive.

- **Output Register**
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 Module/IP Block Wizard while customizing the module.
- **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.
- **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, consequently, corruption of memory contents. During a read cycle, a similar issue can occur that involves the correct contents not being read if the address is not correctly registered in the memory.

2.1. Single Port RAM (RAM_DQ) – EBR Based

Lattice FPGAs support all the features of Single Port Memory Module or RAM_DQ. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module, as shown in [Figure 2.1](#).

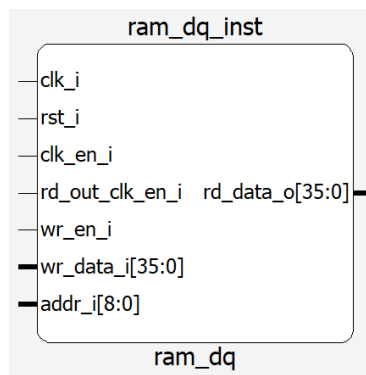


Figure 2.1. Single-Port Memory Module Generated by Module/IP Block Wizard

The various ports and their definitions for Single-Port Memory are listed in [Table 2.1](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.1. 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	Clock Enable
rd_out_clk_en_i	Input	1	Read Output Register Clock Enable
wr_en_i	Input	1	Write Enable
wr_data_i	Input	Data Width	Data Input
addr_i	Input	Address Width	Address Bus
ben_i	Input	Byte Size Width	Byte Enable port
rd_data_o	Output	Data Width	Data Output

The various attributes available for the Single-Port Memory (RAM_DQ) are listed in [Table 2.2](#). Some of these attributes are user selectable through the Module/IP Block Wizard interface.

Table 2.2. EBR-Based Single-Port Memory Attribute Definitions

Attributes	Description	Values	Default Value
Configuration Attributes			
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 ClockEn	Clock Enable for the output clock (this option requires enabling output register)	True, False	False
Reset Assertion	Selection for Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Enable Byte Enable*	Enables the Byte Enable function for the write port	True, False	False
Initialization Attributes			
Memory Initialization	Allows you to initialize memories to all 1s, 0s, or provide custom initialization through a memory file.	none, all 0s, all 1s, memory file	none
Memory File	When Memory File is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different memory file formats.)	binary, hex	hex

***Note:** Byte-Enable can only be used for write ports >= 16 bits.

The Single Port RAM timing waveforms are shown in Figure 2.2 and Figure 2.3.

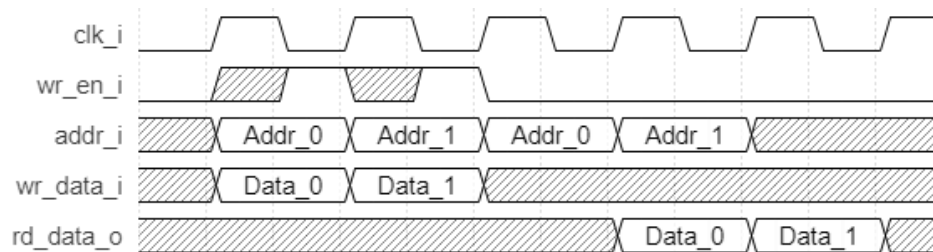


Figure 2.2. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, Without Output Registers

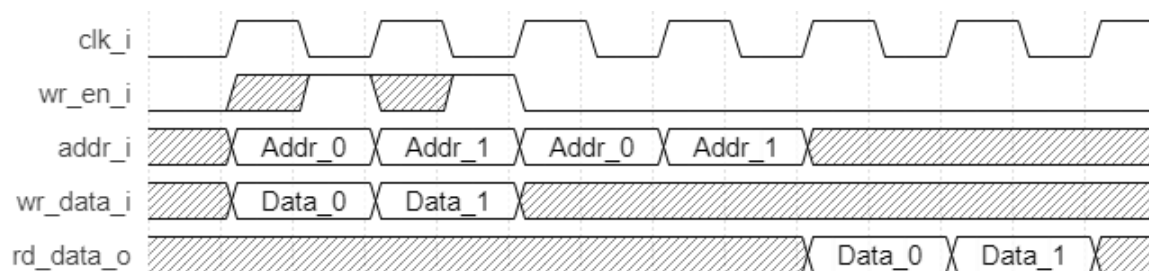


Figure 2.3. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, With Output Registers

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

Lattice FPGAs support all the features of Pseudo-Dual Port Memory Module or RAM_DP. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in Figure 2.4.

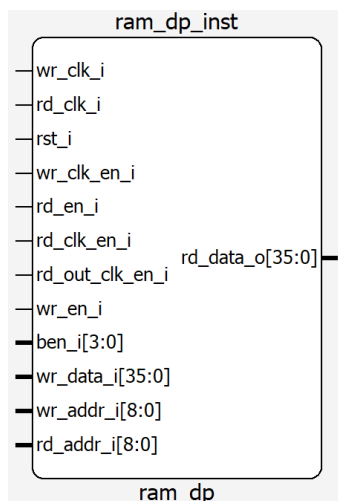


Figure 2.4. Pseudo Dual-Port Memory Module Generated by Module/IP Block Wizard

The various ports and their definitions for Pseudo Dual-Port memory are listed in Table 2.3. The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.3. EBR-Based Pseudo 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_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
wr_data_i	Input	Write Port Data Width	Write Data
wr_addr_i	Input	Write Port Address Width	Write Address
rd_en_i	Input	1	Read Enable
rd_addr_i	Input	Read Address Width	Read Address
ben_i	Input	Byte Size Width	Byte Enable port
rd_data_o	Output	Read Port Data Width	Read Data
one_err_det_o	Output	1	ECC 1-bit error detect
two_err_det_o	Output	1	ECC 2-bit error detect

The various attributes available for the Pseudo Dual-Port Memory (RAM_DP) are listed in [Table 2.4](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.4. EBR-Based Pseudo Dual-Port Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Write Port Address Depth	Write Port Address depth	2 - 65536	512
Write Port Data Width ^{1,2}	Write Port Data word width	1 - 256	36
Read Port Address Depth	Read Port Address depth	2 - 65536	512
Read Port Data Width ^{1,2}	Read Port Data word width	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
Reset Assertion	Selection for the Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Enable Byte Enable ^{3, 4, 5}	Enables the Byte Enable function for the write port	True, False	False
Enable ECC ^{5, 6}	Enable the ECC function for the EBR	True, False	False
Initialization Attributes			
Memory Initialization ⁷	Allows you to initialize their memories to all 1s, 0s, or providing custom initialization through a memory file.	none, all 0s, all 1s, Memory file	none
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex

Notes:

- For mixed width configurations, the total bit size must be equal (that is, $WDEPTH \times WDATA = RDEPTH \times RDATA$)
- For mixed width configuration, the ratio between max DATA and min DATA must be power of 2. (that is. if $WDATA > RDATA$, then $2^n = (WDATA/RDATA)$, where n must be a positive integer and $2^n \leq 32$ [Lattice Nexus devices], $2^n \leq 8$ [iCE40 UltraPlus device]).
- Byte-Enable with mixed-width: the ratio between max DATA and min DATA must be 2 or 4 [Lattice Nexus devices], 2 [iCE40 UltraPlus device]).
- Byte-Enable can only be used for write ports ≥ 16 bits.
- ECC and Byte-Enable cannot be used together
- ECC is available only if $WDATA_WIDTH = RDATA_WIDTH$
- Initialization files should be provided with respect to $WADDR_DEPTH \times WDATA_WIDTH$.

You have the option to enable the output registers for Pseudo-Dual Port RAM (RAM_DP). The internal timing waveforms for Pseudo-Dual Port RAM (RAM_DP) with these options are shown in Figure 2.5 and Figure 2.6. A mixed width timing with no output registers are also provided in Figure 2.7.

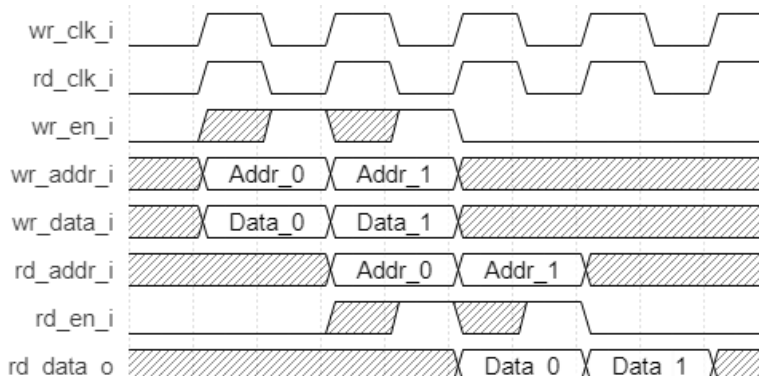


Figure 2.5. PSEUDO DUAL PORT RAM Timing Diagram, Without Output Registers

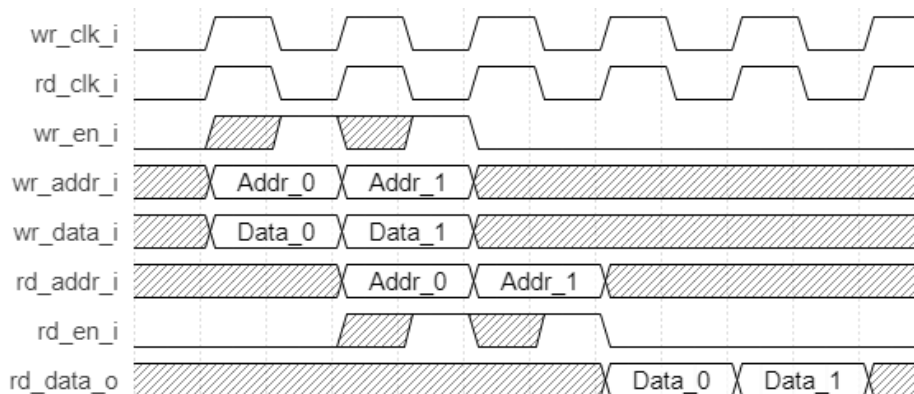


Figure 2.6. PSEUDO DUAL PORT RAM Timing Diagram, With Output Registers

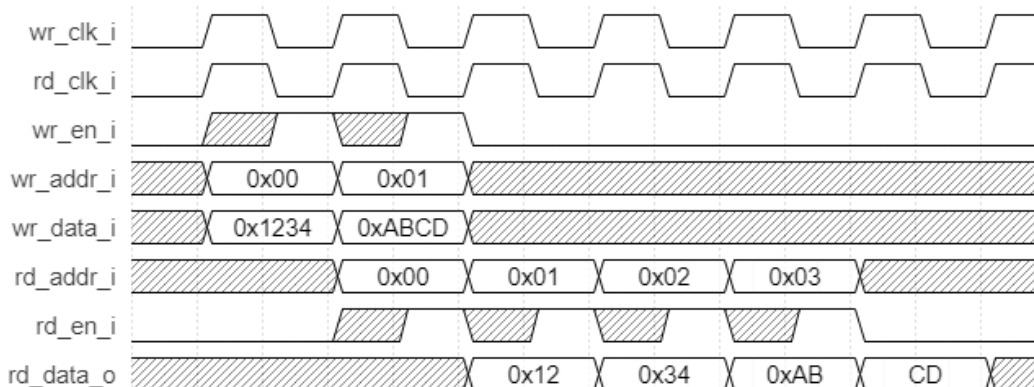


Figure 2.7. PSEUDO DUAL PORT RAM Timing Diagram, Mixed-Width Without Output Registers

2.3. True Dual-Port RAM (RAM_DP True) – EBR Based

Lattice FPGA devices built on the Lattice Nexus platform support all the features of True-Dual Port Memory Module or RAM_DP True. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.8](#).

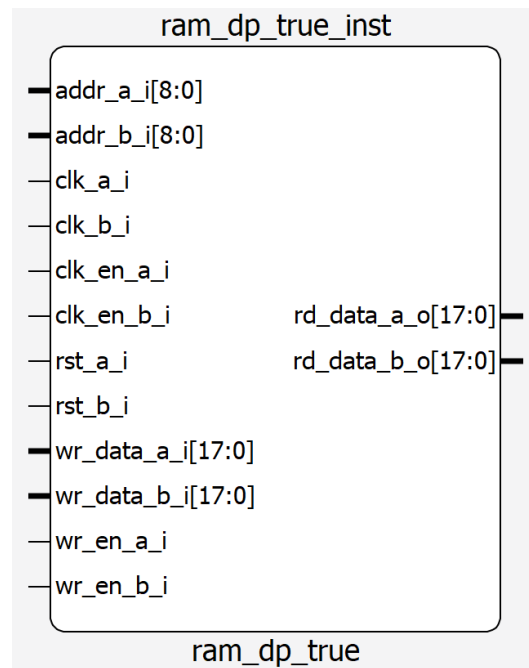


Figure 2.8. True Dual-Port Memory Module Generated by Module/IP Block Wizard

The various ports and their definitions for True Dual-Port memory are listed in [Table 2.5](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.5. EBR-Based True Dual-Port Memory Port Definitions

Port Name	Direction	Width	Description
addr_a_i	Input	Port A Address Width	Port A address
addr_b_i	Input	Port B Address Width	Port B address
wr_data_a_i	Input	Port A Data Width	Port A write data
wr_data_b_i	Input	Port B Data Width	Port B write data
clk_a_i	Input	1	Port A Clock
clk_b_i	Input	1	Port B Clock
clk_en_a_i	Input	1	Port A Clock Enable
clk_en_b_i	Input	1	Port B Clock Enable
wr_en_a_i	Input	1	Port A Write Enable
wr_en_b_i	Input	1	Port B Write Enable
rst_a_i	Input	1	Port A Reset
rst_b_i	Input	1	Port B Reset
ben_a_i	Input	Port A Byte Enable Width	Port A Byte Enable port
ben_b_i	Input	Port B Byte Enable Width	Port B Byte Enable port
rd_data_a_o	Output	Port A Data Width	Port A Output Data
rd_data_b_o	Output	Port B Data Width	Port B Output Data

The various attributes available for the True Dual-Port Memory (RAM_DP True) are listed in [Table 2.6](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.6. EBR-Based True Dual-Port Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Port A Address Depth	Port A Address Depth	2 – 65536	512
Port A Data Width	Port A Data Width	1 – 512	18
Port B Address Depth	Port B Address depth. ^{1, 2}	2 – 65536	512
Port B Data Width	Port B Data word width. ^{1, 2}	1 – 512	18
Enable Output Register (Port A)	Data Out port A (Q) can be registered or not using this selection.	True, False	True
Enable Output Register (Port B)	Data Out port B (Q) can be registered or not using this selection.	True, False	True
Reset Assertion (Port A)	Selection for the Port A Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Reset Assertion (Port B)	Selection for the Port B Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Reset Deassertion (Port A)	Selection for the Port A Reset Release to be Synchronous or Asynchronous to the Clock	async, sync	sync
Reset Deassertion (Port B)	Selection for the Port A Reset Release to be Synchronous or Asynchronous to the Clock	async, sync	sync
Enable Byte Enable (Port A) ^{3, 4}	Enables Byte Enable function for port A's write port.	True, False	False
Enable Byte Enable (Port B) ^{3, 4}	Enables Byte Enable function for port B's write port.	True, False	False
Initialization Attributes			
Memory Initialization	Allows you to initialize their memories to all 1s, 0s, or providing custom initialization through a memory file.	none, all 0s, all 1s, Memory file	none
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex

Notes:

- For mixed width configurations total bit size must be equal (that is, $ADEPTH \times ADATA = BDEPTH \times BDATA$)
- For mixed width configuration the ratio between max DATA and min DATA must be power of 2. (that is, if $ADATA > BDATA$, then $2^n = (ADATA/BDATA)$, where n must be a positive integer and $2^n \leq 16$.)
- Byte-enable for mixed width configuration is applicable only when $max\ DATA = 2 \times min\ DATA$ (that is, if $ADATA > BDATA$, then $FACTOR = 2 = (ADATA / BDATA)$).
- Initialization files should be provided with respect to $ADDR_DEPTH_A \times DATA_WIDTH_A$.

You have the option to enable the output registers for True Dual Port RAM (RAM_DP_TRUE). The internal timing waveforms for True Dual Port RAM (RAM_DP_TRUE) with these options are shown in [Figure 2.9](#) and [Figure 2.10](#). A mixed width timing with no output registers are also provided in [Figure 2.11](#).

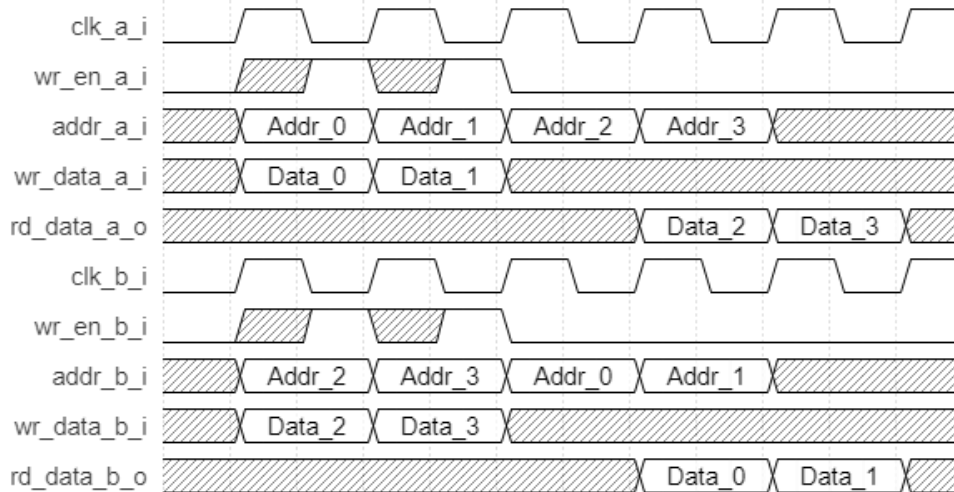


Figure 2.9. TRUE DUAL PORT RAM Timing Diagram, Without Output Registers

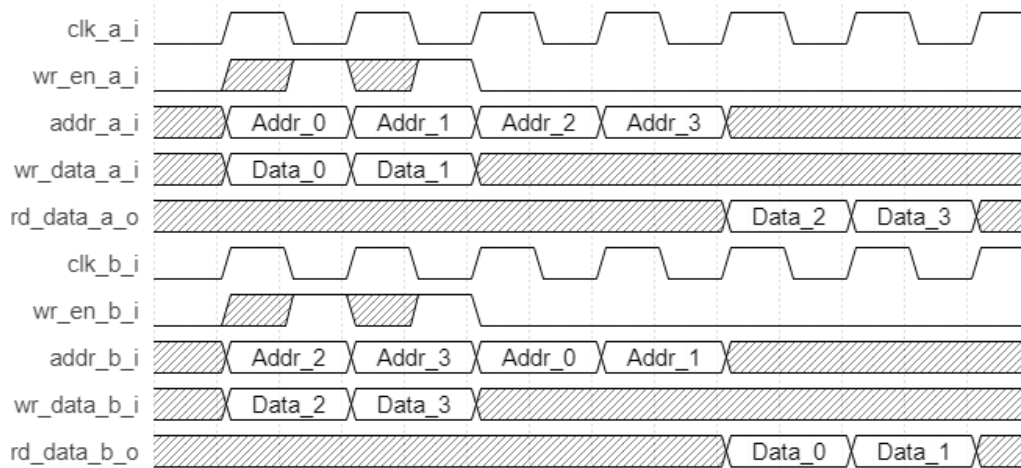


Figure 2.10. TRUE DUAL PORT RAM Timing Diagram, With Output Registers

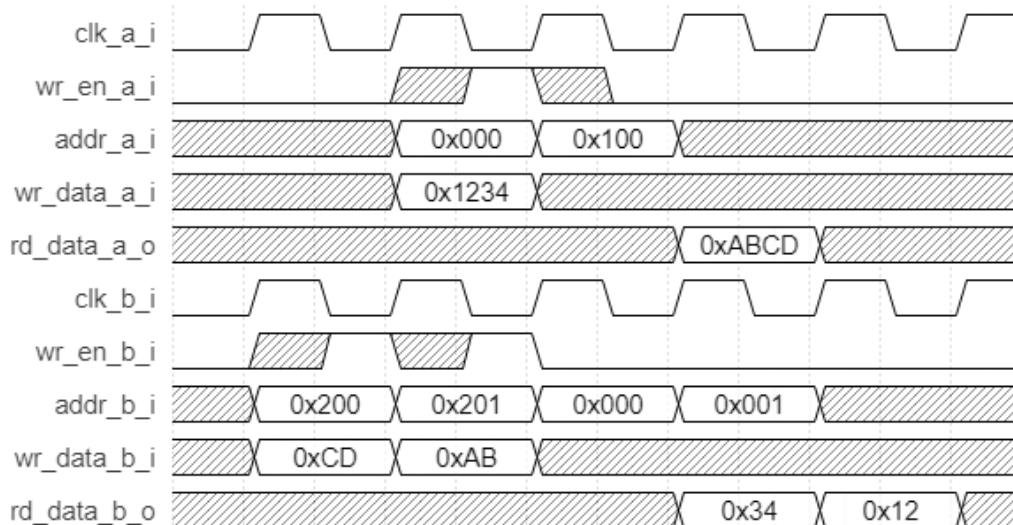


Figure 2.11. TRUE DUAL PORT RAM Timing Diagram, Mixed Width Without Output Registers

2.4. Read Only Memory (ROM) – EBR Based

Lattice FPGAs support all the features of Read Only Memory Module or ROM. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.12](#).

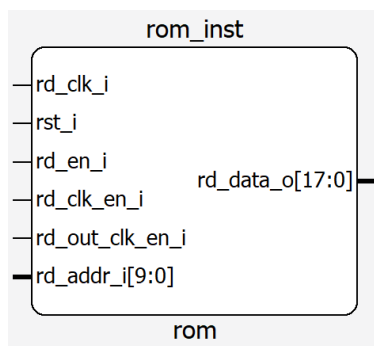


Figure 2.12. Read Only Memory Module Generated by Module/IP Block Wizard

The various ports and their definitions for Read Only Memory are listed in [Table 2.7](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.7. EBR-Based Read Only Memory Port Definitions

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

The various attributes available for the Read Only Memory (ROM) are listed in [Table 2.8](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.8. EBR-Based Read Only Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Read Port Address Depth	Read Port Address Depth	2 - 65536	512
Read Port Data Width	Read Port Data Width	1 - 512	36
Enable Output Register	Data Out (Q) can be registered or not using this selection.	True, False	True
Enable Output ClockEn	Clock Enable for the output clock of (this option requires enabling output register)	True, False	False
Reset Assertion	Selection for the Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Initialization Attributes			
Memory Initialization	Allows you to initialize the memory by providing a custom initialization through a memory file.	Memory file	Memory file
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex

The Read Only Memory timing waveforms are shown in [Figure 2.13](#) and [Figure 2.14](#).



Figure 2.13. ROM Timing Diagram, Without Output Registers



Figure 2.14. ROM Timing Diagram, With Output Registers

2.5. Pseudo Dual-Port RAM (Distributed_DPRAM) – LUT Based

Lattice FPGA devices built on the Lattice Nexus platform support all the features of Pseudo-Dual Port Memory Module or Distributed_DPRAM. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in Figure 2.15.

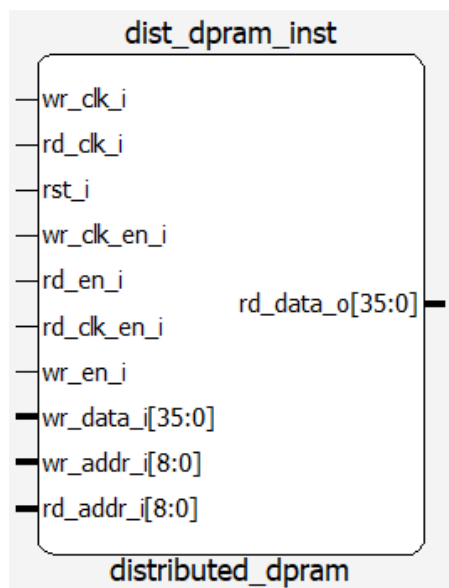


Figure 2.15. Distributed Pseudo Dual Port Module Generated by Module/IP Block Wizard

The various ports and their definitions for Distributed Pseudo Dual-Port memory are listed in Table 2.9. The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.9. LUT-Based Pseudo 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_clk_en_i	Input	1	Read Clock Enable
wr_en_i	Input	1	Write Enable
wr_data_i	Input	Write Port Data Width	Write Data
wr_addr_i	Input	Write Port Address Width	Write Address
rd_en_i	Input	1	Read Enable
rd_addr_i	Input	Read Address Width	Read Address
rd_data_o	Output	Read Port Data Width	Read Data

The various attributes available for the Distributed Pseudo Dual-Port Memory (Distributed_DPRAM) are listed in [Table 2.10](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.10. LUT-Based Pseudo Dual-Port Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Write Port Address Depth	Write Port Address depth	2 - 32768	512
Write Port Data Width	Write Port Data word width	1 - 512	36
Read Port Address Depth	Read Port Address depth ¹	2 - 32768	512
Read Port Data Width	Read Port Data word width ²	1 - 512	36
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
Initialization Attributes			
Memory Initialization	Allows you to initialize their memories to all 1s, 0s, or providing custom initialization through a memory file.	none, all 0s, all 1s, Memory file	none
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex

Notes:

1. Read Port Address Depth != Write Port Address Depth is not supported with distributed memory.
2. Read Port Data Width != Write Port Data Width is not supported with distributed memory.

You have the option to enable the output registers for Pseudo-Dual Port RAM (RAM_DP). The internal timing waveforms for Pseudo-Dual Port RAM (RAM_DP) with these options are shown in [Figure 2.16](#) and [Figure 2.17](#).

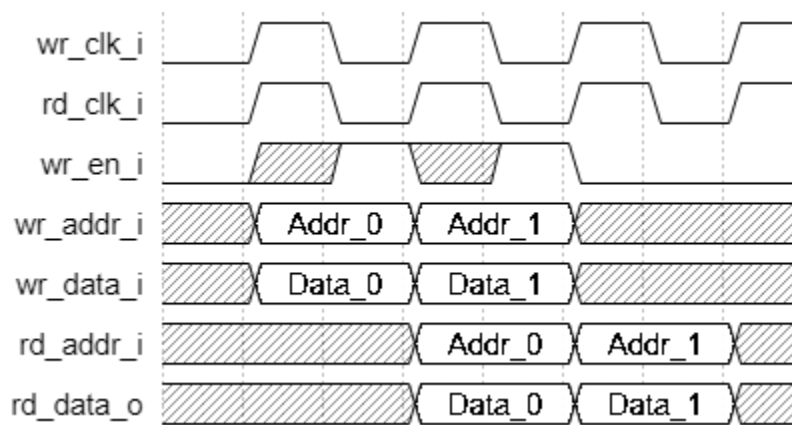


Figure 2.16. Distributed PSEUDO DUAL PORT RAM Timing Diagram, without Output Registers

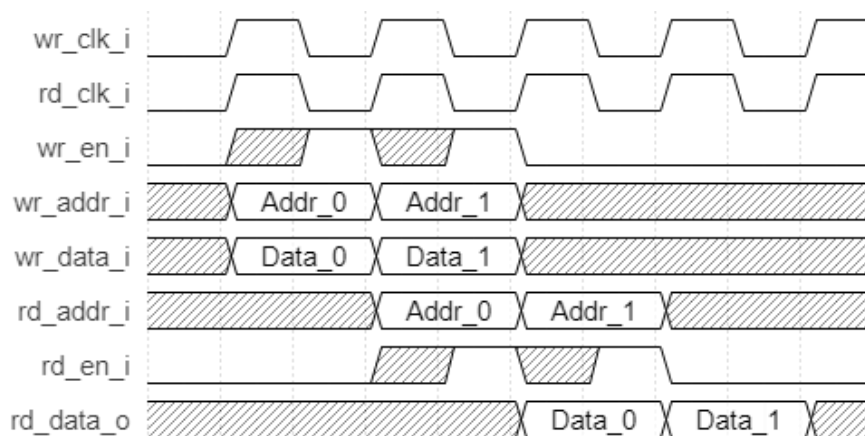


Figure 2.17. Distributed PSEUDO DUAL PORT RAM Timing Diagram, With Output Registers

2.6. Single Port RAM (Distributed_SPRAM) – LUT Based

Lattice FPGA devices built on the Lattice Nexus platform support all the features of Single Port Memory Module or Distributed_SPRAM. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.18](#).

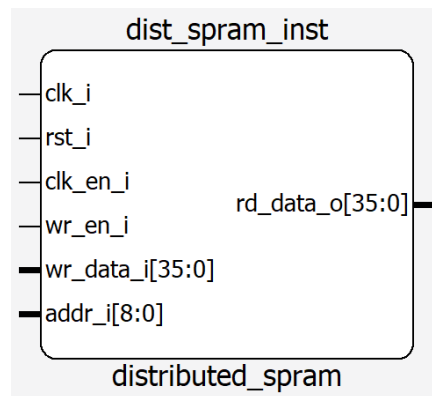


Figure 2.18. Distributed Single Port Module Generated by Module/IP Block Wizard

The various ports and their definitions for Distributed Single-Port Memory are listed in [Table 2.11](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.11. LUT-Based Single-Port Memory Port Definitions

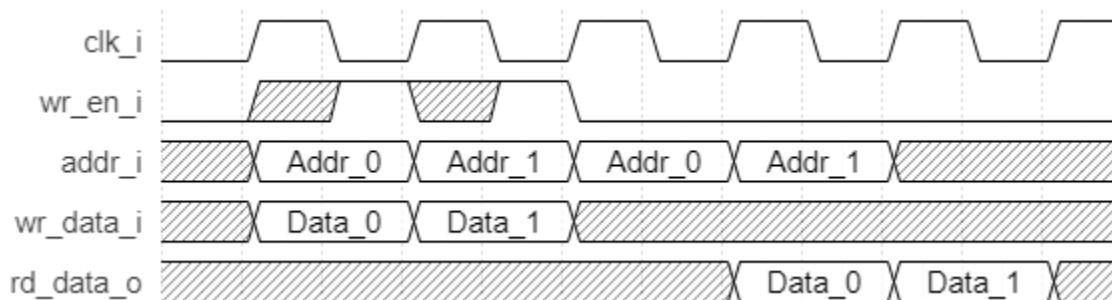
Port Name	Direction	Width	Description
clk_i	Input	1	Clock
rst_i	Input	1	Reset
clk_en_i	Input	1	(Input) Clock Enable
wr_en_i	Input	1	Write Enable
wr_data_i	Input	Data Width	Data Input
addr_i	Input	Address Width	Address Bus
rd_data_o	Output	Data Width	Data Output

The various attributes available for the Distributed Single-Port Memory (Distributed_SPRAM) are listed in [Table 2.12](#). Some of these attributes are user selectable through the Module/IP Block Wizard interface.

Table 2.12. LUT-Based Single-Port Memory Attribute Definitions

Attribute	Description	Values	Default Value
Configuration Attributes			
Address Depth	Address depth of the Read and Write port	2 - 8192	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 Assertion	Selection for Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Initialization Attributes			
Memory Initialization	Allows you to initialize memories to all 1s, 0s, or provide custom initialization through a memory file.	none, all 0s, all 1s, memory file	none
Memory File	When Memory File is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different memory file formats.)	binary, hex	hex

The Distributed Single Port RAM timing waveforms are shown in [Figure 2.19](#) and [Figure 2.20](#).


Figure 2.19. Distributed Single Port RAM Timing Waveform in Normal (NORMAL) Mode, Without Output Registers

Figure 2.20. Distributed Single Port RAM Timing Waveform in Normal (NORMAL) Mode, With Output Registers

2.7. Read Only Memory (Distributed ROM) – LUT Based

Lattice FPGA devices built on the Lattice Nexus platform support all the features of Read Only Memory Module or ROM. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.21](#).

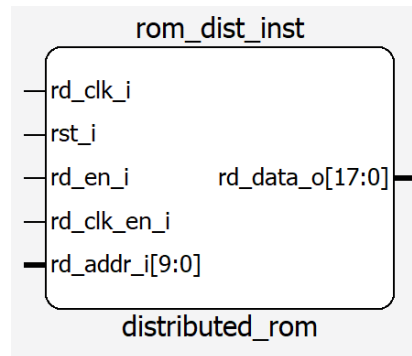


Figure 2.21. Distributed Read Only Memory Module Generated by Module/IP Block Wizard

The various ports and their definitions for Distributed Read Only Memory are listed in [Table 2.13](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.13. LUT-Based Read Only Memory Port Definitions

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

The various attributes available for the Distributed Read Only Memory (Distributed_ROM) are listed in [Table 2.14](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.14. LUT-Based Read Only Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Read Port Address Depth	Read Port Address Depth	2 - 65536	1024
Read Port Data Width	Read Port Data Width	1 - 512	18
Enable Output Register	Data Out (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
Initialization Attributes			
Memory Initialization	Allows you to initialize the memory by providing a custom initialization through a memory file.	Memory file	Memory file
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex

The Read Only Memory timing waveforms are shown in [Figure 2.22](#) and [Figure 2.23](#).

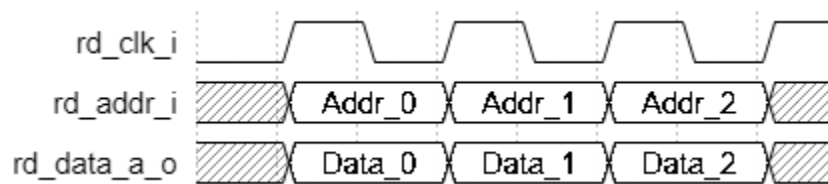


Figure 2.22. Distributed ROM Timing Diagram, Without Output Registers



Figure 2.23. Distributed ROM Timing Diagram, With Output Registers

2.8. First in First Out Single Clock (FIFO)

Lattice FPGAs support all the features of First in First Out Single Clock Memory Module or FIFO. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.12](#).

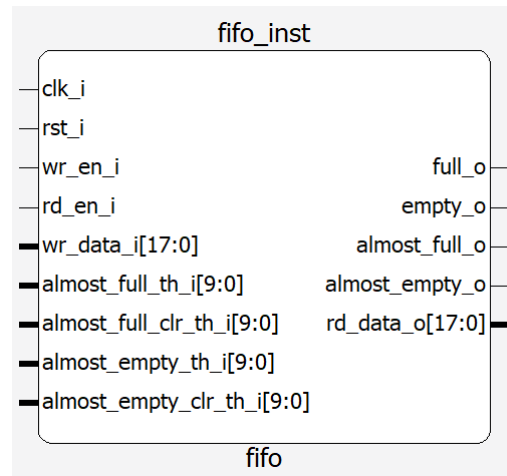


Figure 2.24. Distributed Pseudo Dual Port Module Generated by Module/IP Block Wizard

The various ports and their definitions for First in First Out Single Clock are listed in [Table 2.15](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.15. First in First Out Single Clock Port Definitions

Port Name	Direction	Width	Description
clk_i	Input	1	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	Data Input
almost_full_th_i	Input	Address Width	Almost full threshold (needs single/dual dynamic almost full selected)
almost_full_clr_th_i	Input	Address Width	Almost full clear threshold (needs dual dynamic almost full selected)
almost_empty_th_i	Input	Address Width	Almost empty threshold (needs single/dual dynamic almost empty selected)
almost_empty_clr_th_i	Input	Address Width	Almost empty clear threshold (needs dual dynamic almost empty selected)
rd_data_o	Output	Data Width	Data Output
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 + 1	Data Count – carries the number of data written to the fifo. (requires data count selected).

The various attributes available for the First in First Out Single Clock (FIFO) are listed in. Some of these attributes are user selectable through the Module/IP Block Wizard interface.

Table 2.16. First in First Out Single Clock Attribute Definitions

Attribute	Description	Values	Default Value
Configuration Attributes			
Address Depth ^{1,2,3}	Address depth of the Read and Write port	2 - 65536 ^{1,2,3}	1024
Data Width	Data word width of the Read and Write port	1 - 256	18
Controller Implementation ¹	Chooses how the FIFO controller is implemented. For LIFCL/LFD2NX users, you can opt for an area-optimized or feature-rich option. For iCE40UP users, only feature-rich is available and cannot be changed.	Area-Optimized (HW), Feature-Rich (LUT)	Area-Optimized (HW) for LIFCL/LFD2NX devices. Feature-Rich (LUT) for iCE40UP devices.
Use HIGH Speed Implementation ²	When using a hardware based controller. This option can be checked for an aggressive FIFO routing resulting significantly faster fmax, but can consume a large amount of EBR resources	True, False	False
Implementation Type ³	Chooses how the FIFO memory is implemented.	EBR, LUT	EBR
Enable Output Register	Data Out port (Q) can be registered or not using this selection	True, False	True
Reset Assertion	Selection for Reset to be Synchronous or Asynchronous to the Clock	async, sync	async
Enable Almost Full Flag	Enables or disables the functionality of the almost full flag	enable, disable	enable
Almost Full Assertion	Checks how the almost full flag is implemented. Either dynamic or static, with or without clearing. (requires Enable Almost full flag enabled).	Static – Single Threshold, Static – Dual Threshold, Dynamic – Single Threshold, Dynamic – Dual Threshold.	Static – Single Threshold for LIFCL/LFD2NX devices. Static – Dual Threshold for iCE40UP devices.
Full Assert Level	The current address when written, flags the almost full flag. (Requires Static – Single / Dual Threshold).	0 < Full Assert Level < Address Depth	1023
Full Deassert Level	The current address when read, clears the almost full flag. (Requires Static – Dual Threshold).	0 < Full Deassert Level < Full Assert Level	1020
Enable Almost Empty Flag	Enables or disables the functionality of the almost empty flag	enable, disable	enable
Almost Empty Assertion	Checks how the almost empty flag is implemented. Either dynamic or static, with or without clearing. (requires Enable Almost empty flag enabled).	Static – Single Threshold, Static – Dual Threshold, Dynamic – Single Threshold, Dynamic – Dual Threshold.	Static – Single Threshold for LIFCL/LFD2NX devices. Static – Dual Threshold for iCE40UP devices.
Empty Assert Level	The current address when read, flags the almost empty flag. (Requires Static – Single / Dual Threshold).	0 < Empty Assert Level < Address Depth	1
Empty Deassert Level	The current address, when written, clears the almost empty flag. (Requires Static – Dual Threshold).	Empty Assert Level < Empty Deassert Level < Address Depth	4
Enable Data Count	Enables counting the number of data written to the FIFO	enable, disable	disable

Notes:

1. For Feature-Rich (LUT) based implementations, the ADDRESS_DEPTH must be a power of 2 (that is, 2, 4, ... etc.), for Area-Optimized (HW) implementation there is no limitation on the ADDRESS_DEPTH.
2. For HIGH Speed Implementations, maximum ADDRESS_DEPTH is 16363 only.
3. For LUT implementations, maximum ADDRESS_DEPTH is 8192 only.

Differences between Feature-Rich (LUT) and Area-Optimized (HW) based implementations:

1. Address Depth Limits
 - In LUT based controller, the FIFO controller is implemented on the fabric and due to this nature LUT based FIFO controllers are limited to 2^n values. In contrast with the HW based controller, the complexities on the counter and comparator has been offloaded to custom hardware and therefore is not limited by the optimized 2^n values.
 - The drawback is that the HW based controller can consume much more EBR resources than the LUT based controller. To maximize resource and optimize maximum speed, opt for $(2^n - 1)$ values such as 511, 1023, and 2047 when implementing FIFO design in HW-based controller. 2^n configurations can also be implemented, but we caution users that they might see 25 – 100% more EBR resources consumed when compared to the LUT based controller.
2. Flag and Data Counter Limitations
 - The HW based controller contains the very basic requirements for a FIFO. Full, Empty, Single-Assert Almost Full and Almost Empty are provided. While those functions are suitable for most applications. LUT based controllers provides several other features implemented in fabric such as data counters and dual-assert almost full / almost empty flags and even dynamic flag control to provide users more flexibility in their designs.
3. Output Register Implementation
 - In the HW based controller, the output register is implemented by adding a pre-read in the FIFO in order to put the data in the pipeline. This is important because it can affect the de-assertion of the almost full flag, when a pre-read occurs one data is removed from stack and pushed into the pipeline essentially freeing up one slot in the memory space. Because of this, the almost full flag can de-assert one read earlier than expected, however this does not mean that the data would be lost. The free memory space CAN be utilized and written upon, and you can read the data at the output side without worrying of any lost data in the sequence.

Timing diagrams for FIFO Single clock are shown in [Figure 2.25](#) and [Figure 2.26](#), for configurations without register and configuration with registers, respectively. (FIFO Configuration uses ADDRESS_DEPTH = 4).

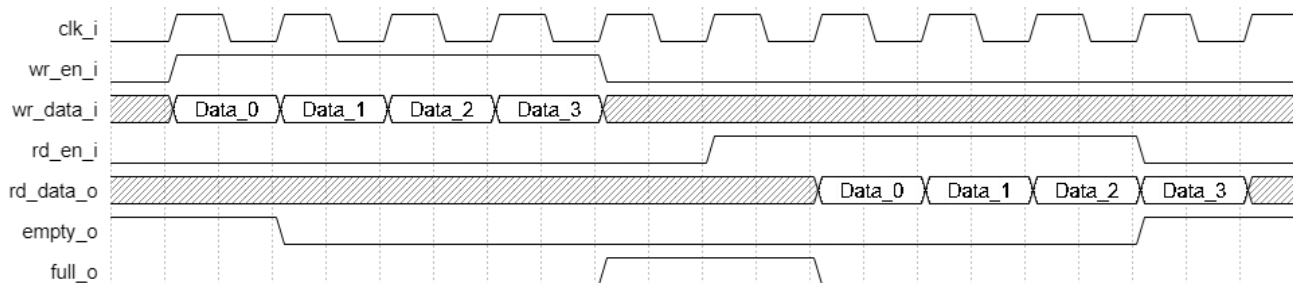


Figure 2.25. FIFO Single Clock Without Registers

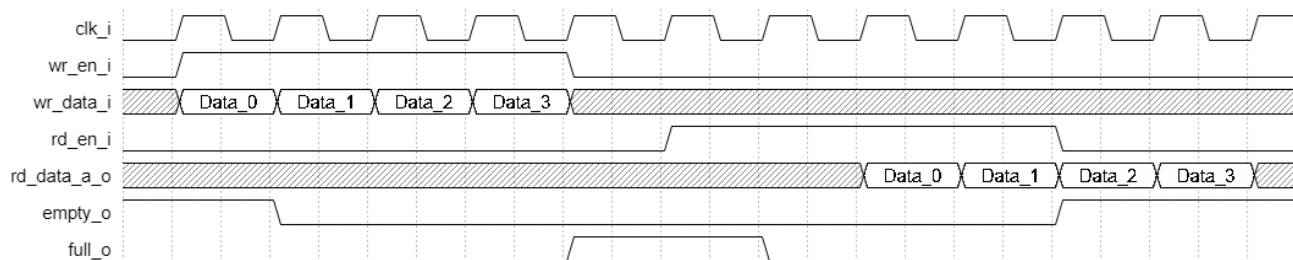


Figure 2.26. FIFO Single Clock With Registers

2.9. First in First Out Dual Clock (FIFO_DC)

Lattice FPGAs support all the features of First in First Out Dual Clock Memory Module or FIFO_DC. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.27](#).

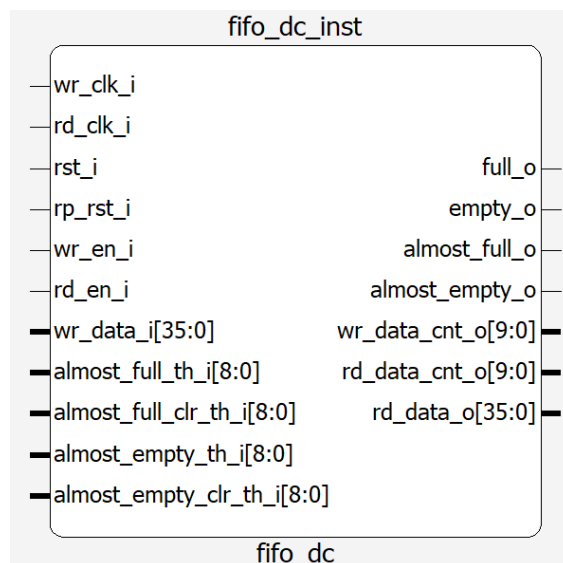


Figure 2.27. FIFO Dual Clock Module Generated by Module/IP Block Wizard

The various ports and their definitions for First in First Out Dual Clock are listed in [Table 2.17](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.17. First in First Out Dual Clock 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	Fifo Empty Reset
rp_rst_i*	Input	1	Fifo Full Reset
wr_en_i	Input	1	Write Enable
rd_en_i	Input	1	Read Enable
wr_data_i	Input	Write Data Width	Data Input
almost_full_th_i	Input	Write Address Width	Almost full threshold (needs single/dual dynamic almost full selected)
almost_full_clr_th_i	Input	Write Address Width	Almost full clear threshold (needs dual dynamic almost full selected)
almost_empty_th_i	Input	Read Address Width	Almost empty threshold (needs single/dual dynamic almost empty selected)
almost_empty_clr_th_i	Input	Read Address Width	Almost empty clear threshold (needs dual dynamic almost empty selected)
rd_data_o	Output	Read Data Width	Data Output
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

Port Name	Direction	Width	Description
wr_data_cnt_o	Output	Write Address Width + 1	Write Data Count – carries the number of data written to the fifo. (requires data count selected).
rd_data_cnt_o	Output	Read Address Width + 1	Read Data Count – carries the number of data read from the fifo. (requires data count selected).

***Note:** rp_rst_i and its accompanying pmi signal RPRreset, is a read-clock synchronized signal. rp_rst or read pointer reset is designed to reset the dual clock FIFO IP to full, and re-send the previous set of data, typical for packet transmissions. For the proper use of this signal, follow these items in sequence:

1. Stop writing to the IP
2. Assert rp_rst_i
3. Wait for the full flag to assert
4. Release rp_rst_i

The various attributes available for the First in First Out Dual Clock (FIFO_DC) are listed in [Table 2.18](#). Some of these attributes are user selectable through the Module/IP Block Wizard interface.

Table 2.18. First in First Out Dual Clock Attribute Definitions

Attribute	Description	Values	Default Value
Configuration Attributes			
Write Port Address Depth ^{1,2,3,4}	Write Port Address depth	2 - 65536	512
Write Port Data Width ¹	Write Port Data word width	1 - 256	18
Read Port Address Depth ^{1,2,3,4}	Read Port Address depth	2 - 65536	512
Read Port Data Width ¹	Read Port Data word width	1 - 256	18
Controller Implementation ²	Chooses how the FIFO controller is implemented. For LIFCL/LFD2NX users, you can opt for an area-optimized or feature-rich option. For iCE40UP users, only feature-rich is available and cannot be changed.	Area-Optimized (HW), Feature-Rich (LUT)	Area-Optimized (HW) for LIFCL/LFD2NX devices. Feature-Rich (LUT) for iCE40UP devices.
Use HIGH Speed Implementation ³	When using a hardware based controller. This option can be checked for an aggressive FIFO routing resulting significantly faster fmax, but can consume a large amount of EBR resources	True, False	False
Implementation ⁴	Chooses how the FIFO_DC memory is implemented.	EBR, LUT	EBR
Enable Output Register	Data Out port (Q) can be registered or not using this selection	True, False	True
Reset Assertion	Selection for Reset to be Synchronous or Asynchronous to the Clock	async, sync	async
Enable Almost Full Flag	Enables or disables the functionality of the almost full flag	enable, disable	enable

Attribute	Description	Values	Default Value
Almost Full Assertion	Checks how the almost full flag is implemented. Either dynamic or static, with or without clearing. (requires Enable Almost full flag enabled).	Static – Single Threshold, Static – Dual Threshold, Dynamic – Single Threshold, Dynamic – Dual Threshold.	Static – Single Threshold for LIFCL/LFD2NX devices. Static – Dual Threshold for iCE40UP devices.
Full Assert Level	The current address when written, flags the almost full flag. (Requires Static – Single / Dual Threshold).	$0 < \text{Full Assert Level} < \text{Write Address Depth}$	511
Full Deassert Level	The current address when read, clears the almost full flag. (Requires Static – Dual Threshold).	$0 < \text{Full Deassert Level} < \text{Full Assert Level}$	508
Enable Almost Empty Flag	Enables or disables the functionality of the almost empty flag	enable, disable	enable
Almost Empty Assertion	Checks how the almost empty flag is implemented. Either dynamic or static, with or without clearing. (requires Enable Almost empty flag enabled).	Static – Single Threshold, Static – Dual Threshold, Dynamic – Single Threshold, Dynamic – Dual Threshold.	Static – Single Threshold for LIFCL/LFD2NX devices. Static – Dual Threshold for iCE40UP devices.
Empty Assert Level	The current address when read, flags the almost empty flag. (Requires Static – Single / Dual Threshold).	$0 < \text{Empty Assert Level} < \text{Read Address Depth}$	1
Empty Deassert Level	The current address when written, clears the almost empty flag. (Requires Static – Dual Threshold).	$\text{Empty Assert Level} < \text{Empty Deassert Level} < \text{Read Address Depth}$	4
Enable Data Count (Write)	Enables counting the amount of data written to the FIFO_DC	enable, disable	disable
Enable Data Count (Read)	Enables counting the amount of data read from the FIFO_DC	enable, disable	disable

Notes:

- For mixed width configurations total bit size must be equal (that is, $\text{WDEPTH} * \text{WDATA} = \text{RDEPTH} * \text{RDATA}$). For mixed width configuration the ratio between maximum DATA and minimum DATA must be power of 2. (that is, if $\text{WDATA} > \text{RDATA}$, then $2^n = (\text{WDATA}/\text{RDATA})$, where n must be a positive integer and $2^n \leq 32$ [Lattice Nexus devices], $2^n \leq 8$ [iCE40UP device]).
- For Feature-Rich (LUT) based implementations, the ADDRESS_DEPTH must be a power of 2 (that is, 2, 4, ... etc.), for Area-Optimized (HW) implementation there is no limitation on the ADDRESS_DEPTH.
- For HIGH Speed Implementations, maximum ADDRESS_DEPTH is 16363 only.
- Mixed width implementations cannot be used with LUT memory, maximum ADDRESS_DEPTH is 8192 only.

Differences between Feature-Rich (LUT) and Area-Optimized (HW) based implementations:

1. Latency Differences

- In the LUT based controller, the FIFO controller is implemented on the fabric, and due to this nature a synchronization logic is required to translate the write-clock domain transactions to the read-clock and vice-versa in order to properly maintain the integrity of the data being passed. This introduces a latency to the system and affects the flags from either side of the domain. A full flag operating in the write-domain would need to wait 3 cycles before de-assertion because of a read in the read-domain, however no latency is required for flag assertion since a write would operate in same write-clock domain. In HW based controllers, a special circuit is utilized to mitigate clock crossing issues and removes this latency altogether, as seen on [Figure 2.31](#).

2. Address Depth Limits

- In LUT based controller, the FIFO limited to 2^n values. In contrast with the HW based controller, the complexities on the counter and comparator has been offloaded to custom hardware and therefore is not limited by the optimized 2^n values.
- The drawback is that the HW based controller can consume much more EBR resources than the LUT based controller. To maximize resource and optimize maximum speed, opt for $(2^n - 1)$ values such as 511, 1023, and 2047 when implementing FIFO design in HW-based controller. 2^n configurations can also be implemented, but we caution users that they might see 25 – 100% more EBR resources consumed when compared to the LUT based controller.

3. Flag and Data Counter Limitations

- The HW based controller contains the very basic requirements for a FIFO. Full, Empty, Single-Assert Almost Full and Almost Empty are provided. While those functions are suitable for most applications. LUT based controllers provides several other features implemented in fabric such as data counters and dual-assert almost full / almost empty flags and even dynamic flag control to provide users more flexibility in their designs.

4. Output Register Implementation

- In the HW based controller, the output register is implemented by adding a pre-read in the FIFO in order to put the data in the pipeline. This is important because it can affect the de-assertion of the almost full flag, when a pre-read occurs one data is removed from stack and pushed into the pipeline essentially freeing up one slot in the memory space. Because of this, the almost full flag can de-assert one read earlier than expected, however this does not mean that the data would be lost. The free memory space CAN be utilized and written upon, and you can read the data at the output side without worrying of any lost data in the sequence.
- In mixed width application the pre-read counts as one (1) read word. This means the effect of the pre-read scales with the write-to-read ratio. If $W > R$, then the effects of the pre-read on the flag might be miniscule, but for $W < R$, the pre-read can prematurely de-assert the almost-full flag much much earlier.

Timing diagrams for FIFO Dual clock are shown in Figure 2.28 and Figure 2.29, for configurations without register and with registers, respectively. An additional timing diagram is provided for Figure 2.30 for mixed-width configuration without registers, and Figure 2.31 and Figure 2.32 shows the Hard Controller waveform. (FIFO_DC uses WADDR_DEPTH = 4)

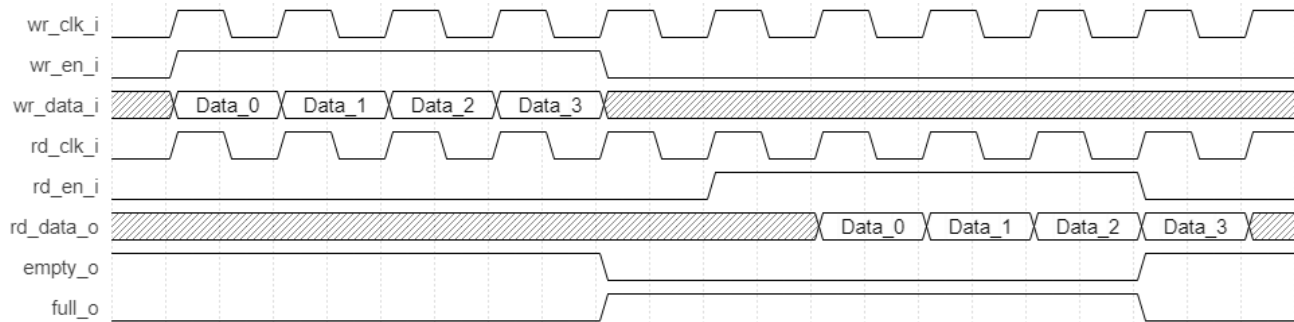


Figure 2.28. FIFO Dual Clock Module Without registers

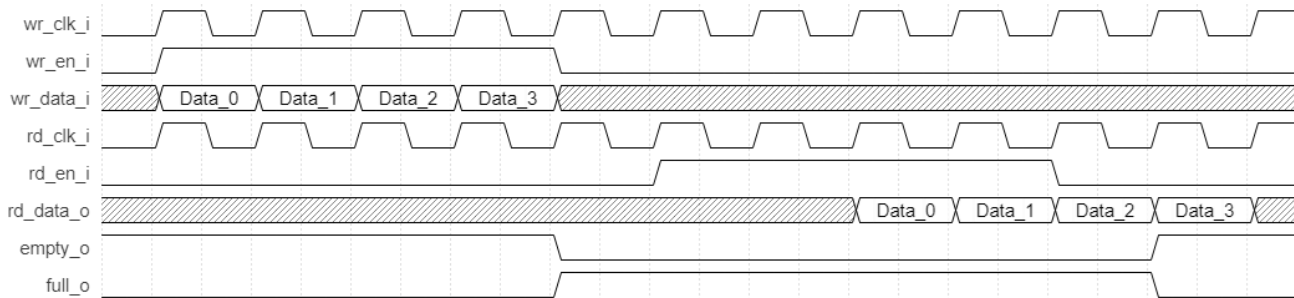


Figure 2.29. FIFO Dual Clock Module With Registers

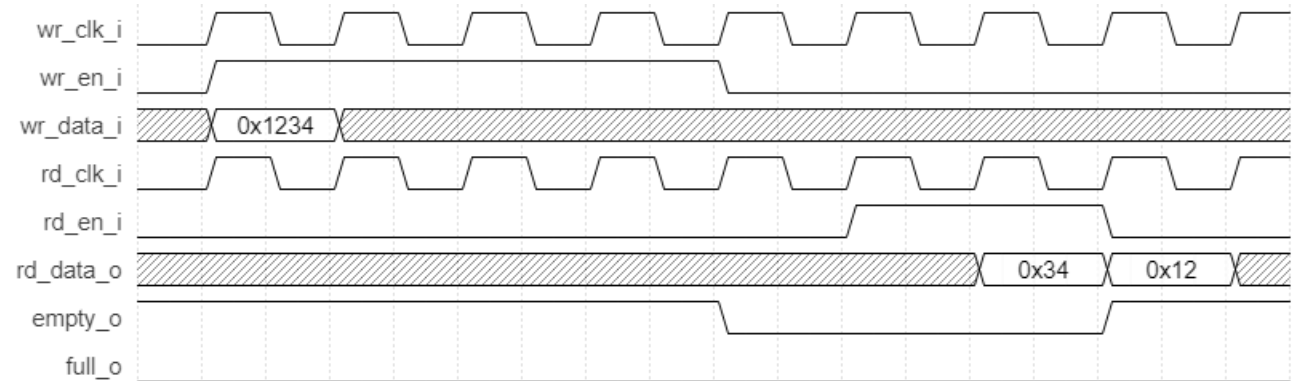


Figure 2.30. FIFO Dual Clock Module, Mixed-Width Without registers

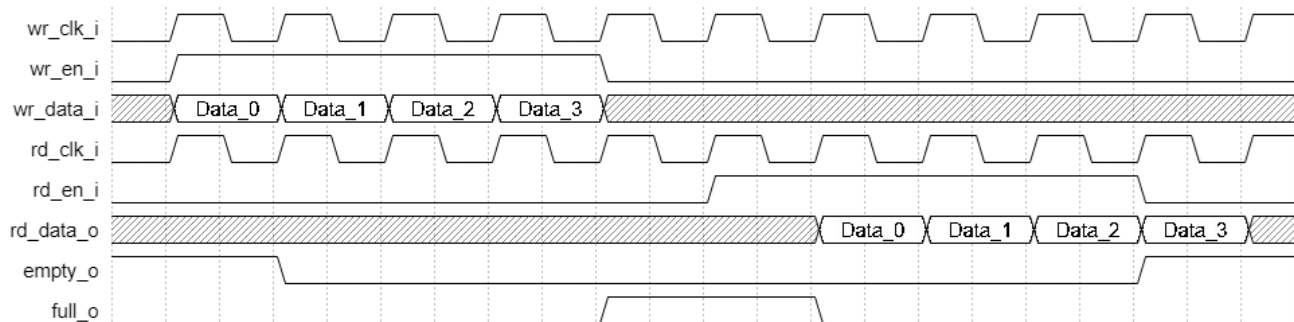


Figure 2.31. Hard Controller FIFO Dual Clock Module Without registers

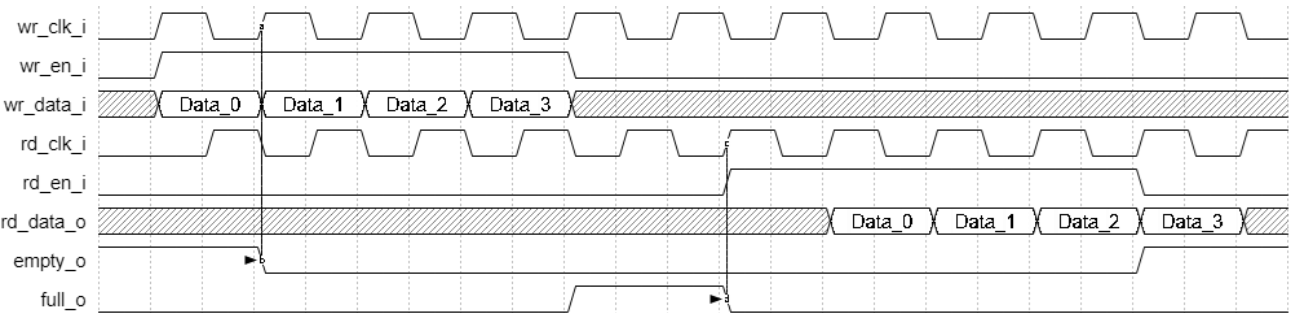


Figure 2.32. Hard Controller FIFO Dual Clock Module Without registers operating in asynchronous clocks.

2.10. Shift Register (Shift_Register)

Lattice FPGAs support all the features of Shift Register or Shift_Register. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.33](#).

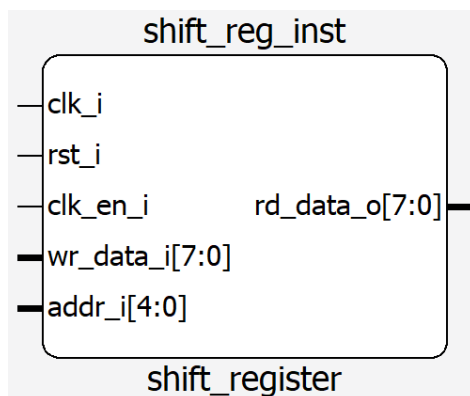


Figure 2.33. Shift Register Module Generated by Module/IP Block Wizard

The various ports and their definitions for Shift Register are listed in [Table 2.19](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.19. Shift Register Port Definitions

Port Name	Direction	Width	Description
clk_i	Input	1	Clock
rst_i	Input	1	Reset
clk_en_i	Input	1	Clock Enable
wr_data_i	Input	Data Width	Data Input
addr_i	Input	Max Shift Width	Current amount of shift from the current address. (Requires shift register type to lossless, or variable).
rd_data_o	Output	Data Width	Data Output

The various attributes available for the Shift Register (Shift_Register) are listed in [Table 2.20](#). Some of these attributes are user selectable through the Module/IP Block Wizard interface.

Table 2.20. Shift Register Attribute Definitions

Attribute	Description	Values	Default Value
Configuration Attributes			
Max Shift ¹	The maximum address shifts allowed	2 - 8192	16
Data Width ¹	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
Shift Register Type	Chooses whether, the shift register is operating at: (1) fixed delta from the currently written address, or (2) variable delta from the currently written address	fixed, variable	fixed
Implementation ¹	Chooses how the shift register memory is implemented.	EBR, LUT	EBR

***Note:** For EBR implementations, this IP requires a minimum size of 30-bits to be properly implemented.

Timing diagrams for Shift Register are shown in [Figure 2.34](#) and [Figure 2.35](#), for non-registered and registered configurations respectively. A timing diagram is also provided for variable shift configuration without registers see [Figure 2.36](#) and a LUT based shift register on [Figure 2.37](#) [Lattice Nexus devices] and [Figure 2.38](#) [Thunder Plus] (Shift Register uses MAX_DEPTH = 4 for fixed configuration and MAX_DEPTH = 8 for variable configuration).

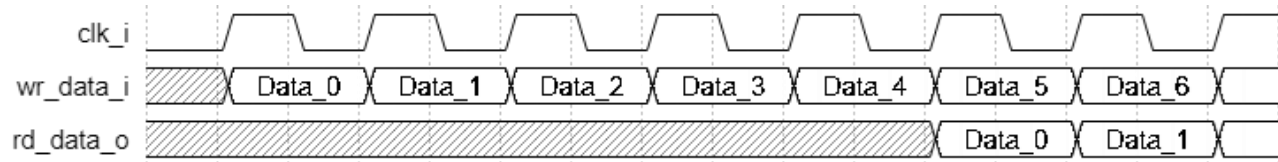


Figure 2.34. Shift Register Without Output Register



Figure 2.35. Shift Register With Output Register

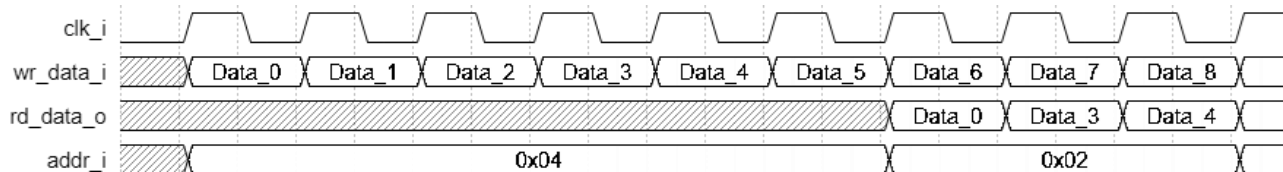


Figure 2.36. Shift Register Variable Configuration and Without Output Register

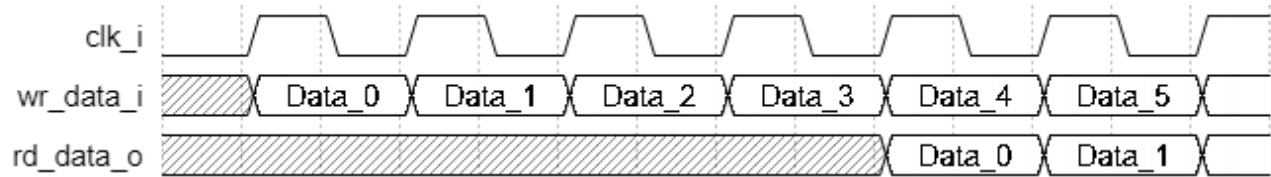


Figure 2.37. Shift Register Without Output Register (LUT Memory, Lattice Nexus Devices)

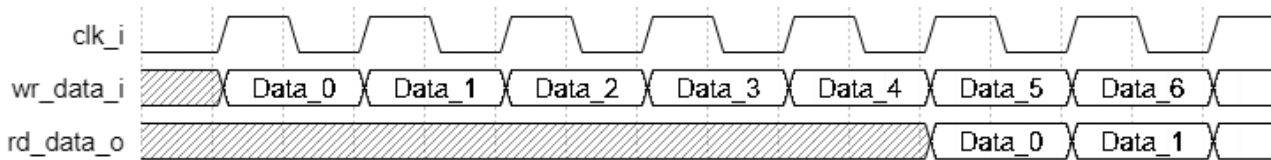


Figure 2.38. Shift Register Without Output Register (LUT Memory, iCE40 UltraPlus)

2.11. Single Port Large RAM (Large_RAM_SP)

Lattice FPGA devices built on the Lattice Nexus platform support all the features of Single Port Large RAM Module or Large_RAM_SP. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

The Single Port Large RAM Module/IP Block Wizard is shown in [Figure 2.39](#)

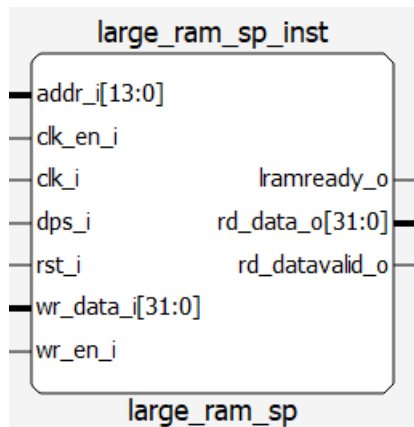


Figure 2.39. Single-Port Large RAM Module Generated by Module/IP Block Wizard

The various ports and their definitions for Single-Port Memory are listed in [Table 2.21](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.21. Large-RAM based Single-Port Memory Port Definitions

Port Name	Direction	Width	Description
clk_i	Input	1	Clock
rst_i	Input	1	Reset
dps_i	Input	1	Dynamic Power Switching – when LOW means the LRAM is in normal operation mode, when HIGH means the LRAM is in low power mode, and write/read functions are stopped.
clk_en_i	Input	1	Clock Enable
rd_out_clk_en_i	Input	1	Read Output Register Clock Enable
wr_en_i	Input	1	Write Enable
wr_data_i	Input	Data Width	Data Input
addr_i	Input	Address Width	Address Bus
ben_i	Input	Byte Size Width	Byte Enable port / Unaligned Access shared port
lramready_o	Output	1	When high means that the LRAM is ready for operation
rd_datavalid_o	Output	1	When high means the current output values of rd_data_o and ecc_errdet_o are valid.
rd_data_o	Output	Data Width	Data Output
errdet_o	Output	1	When HIGH means an error has occurred in the LRAM
ecc_errdet_o	Output	2	ECC output result: MSB – Two error detect LST – One error detect

The various attributes available for the Single-Port Memory (Large_RAM_SP) are listed in [Table 2.22](#). Some of these attributes are user selectable through the Module/IP Block Wizard interface.

Table 2.22. Large RAM-Based Single-Port Memory Attribute Definitions

Attributes	Description	Values	Default Value
Configuration			
Address Depth	Address depth of the Read and Write port	2 – 131072	16384
Data Width	Data word width of the Read and Write port	1 – 64	32
Enable Output Register	Data Out port (rd_data_o) 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
Reset Assertion	Selection for Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Reset De-Assertion	Selection for Reset De-Assertion to be Synchronous or Asynchronous to the Clock. (This option requires an asynchronous reset assert)	async, sync	sync
Enable Byte Enable ^{1, 2, 3}	Enables the Byte Enable function for the write port	True, False	False
Initialization			
Memory Initialization	Allows you to initialize memories to all 1s, 0s, or provide custom initialization through a memory file.	none, all 0s, all 1s, memory file	none
Memory File	When Memory File is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different memory file formats.)	binary, hex	hex
Miscellaneous Options			
Write/Read Related			
Enable ECC ^{2, 4}	Enables ECC functionality of the IP	True, False	False
Select Write Mode	Sets the behavior of the output port rd_data_o based on the current write transaction.	“normal”, “write-through”, “read-before-write”	“normal”
Enable Unaligned Read ^{3, 4}	Enables the unaligned read functionality of the Soft IP which shift the data out based on the current port input at the time of the read command	True, False	False
Others			
Enable Preserve Array	Allows the LRAM to preserve the array when the IP is set to low power mode	True, False	True
Enable GSR	Enables the GSR to reset this IP	True, False	True

Notes:

1. Byte-Enable can only be used for write ports ≥ 16 bits.
2. Byte-Enable CANNOT be used with ECC.
3. Byte-Enable port ben_i is shared with Unaligned Read functionality. The port functions as byte-enable if wr_en_i is HIGH, and as unaligned read when wr_en_i is LOW.
4. ECC and Unaligned Read can only be enabled when DATA_WIDTH = 32.

2.11.1. Unaligned Access Feature

The Large RAM Module supports RISC-V compressed instruction chunks of data and shift them. If RISC-V needs to read the upper 16-bit of data in some address, it is very helpful to add support for shifting the upper 16 bits of output into the lower 16 bits, padding the upper bits with to 0. Unaligned Read enables this feature; Table 2.23 shows possible combinations for Unaligned Read shifting. The unaligned read pins are shared with the `ben_i` ports for Single Port Large RAM. Given this, you should be careful in changing the value of these signals. The port functions as byte-enabled during write-access and unaligned read during read-access.

Table 2.23. Unaligned Read shift function (Single Port LRAM)

<code>ben_i[1:0]</code>	<code>ben_i[2] = 0</code>	<code>ben_i[2] = 1</code>
00	$DOx = x[31:0]$ (no shift)	$DOx = x[31:0]$ (no shift)
01	$DOx = \{8'b0, x[31:8]\}$	$DOx = \{x[23:0], 8'b0\}$
10	$DOx = \{16'b0, x[31:16]\}$	$DOx = \{x[15:0], 16'b0\}$
11	$DOx = \{24'b0, x[31:24]\}$	$DOx = \{x[7:0], 24'b0\}$

DOx refers to `rd_data_o` output port of the Large RAM.

Unaligned Read is only supported if `DATA_WIDTH = 32`.

Timing diagrams for Large RAM Single Port are shown in Figure 2.40 and Figure 2.41, for non-registered and registered configurations respectively. Additional timing diagrams are provided for write-through, Figure 2.42, and read-before-write transactions, Figure 2.43.

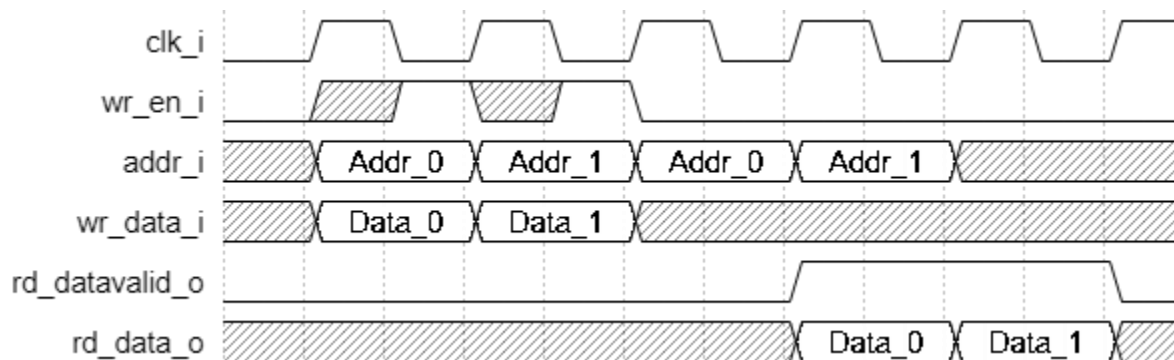


Figure 2.40. Single Port LRAM Timing Waveform in NORMAL Mode, Without Output Registers

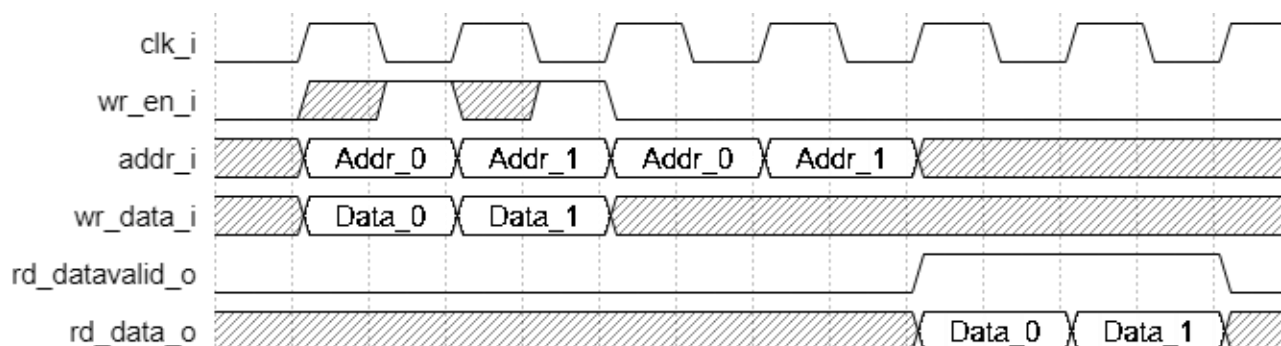


Figure 2.41. Single Port RAM Timing Waveform in NORMAL Mode, With Output Registers



Figure 2.42. Single Port LRAM Timing Waveform in Write-Through Mode, Without Output Registers

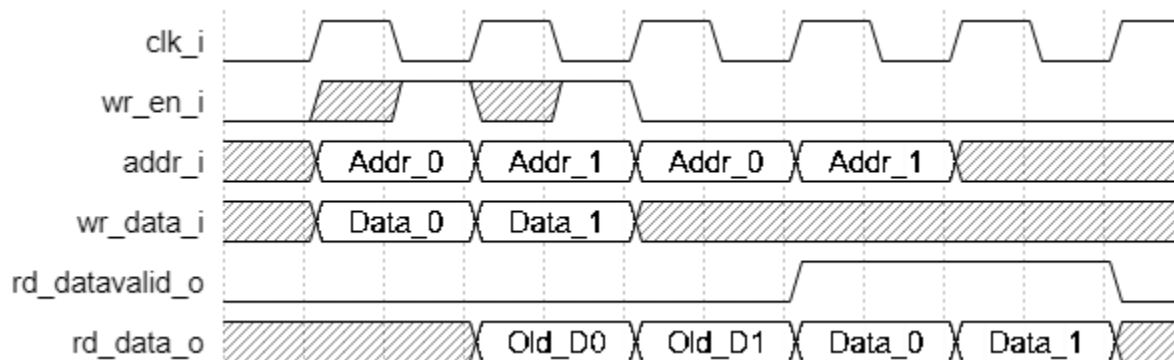


Figure 2.43. Single Port RAM Timing Waveform in Read-Before-Write Mode, Without Output Registers

2.12. Pseudo Dual Port Large RAM (Large_RAM_DP)

Lattice FPGA devices built on the Lattice Nexus platform support all the features of Pseudo-Dual Port Large RAM Module or Large RAM_DP. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in Figure 2.44.

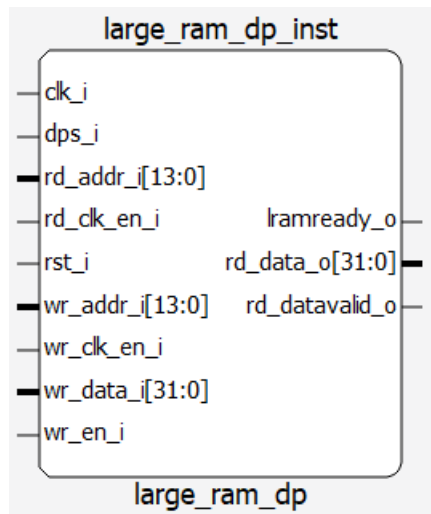


Figure 2.44. Pseudo Dual-Port Large RAM Module Generated by Module/IP Block Wizard

The various ports and their definitions for Pseudo Dual-Port memory are listed in Table 2.24. The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.24. Large RAM-Based Pseudo Dual-Port Memory Port Definitions

Port Name	Direction	Width	Description
clk_i	Input	1	Clock
rst_i	Input	1	Reset
dps_i	Input	1	Dynamic Power Switching – when HIGH means the LRAM is in normal operation mode, when LOW means the LRAM is in low power mode, and write/read functions are stopped.
wr_clk_en_i	Input	1	Write Clock 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
wr_data_i	Input	Write Port Data Width	Write Data
wr_addr_i	Input	Write Port Address Width	Write Address
rd_en_i	Input	1	Read Enable
rd_addr_i	Input	Read Address Width	Read Address
ben_i	Input	Byte Size Width	Byte Enable port
unaligned_i	Input	4	Unaligned Access Port
lramready_o	Output	1	When high means that the LRAM is ready for operation
rd_datavalid_o	Output	1	When high means the current output values of rd_data_o and ecc_errdet_o are valid.
rd_data_o	Output	Read Port Data Width	Read Data
errdet_o	Output	1	When HIGH means an error has occurred in the LRAM
ecc_errdet_o	Output	2	ECC output result: MSB – Two error detect LST – One error detect

The various attributes available for the Pseudo Dual-Port Memory (RAM_DP) are listed in [Table 2.25](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.25. Large RAM-Based Pseudo Dual-Port Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Write Port Address Depth ^{1, 2}	Write Port Address depth	2 – 131072	16384
Write Port Data Width ^{1, 2}	Write Port Data word width	1 – 64	32
Read Port Address Depth ^{1, 2}	Read Port Address depth	2 – 131072	16384
Read Port Data Width ^{1, 2}	Read Port Data word width	1 – 64	32
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
Reset Assertion	Selection for the Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Enable Byte Enable ^{3, 4}	Enables the Byte Enable function for the write port	True, False	False
Initialization Attributes			
Memory Initialization ⁵	Allows you to initialize their memories to all 1s, 0s, or providing custom initialization through a memory file.	none, all 0s, all 1s, Memory file	none
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex
Miscellaneous Options			
Write/Read Related			
Enable ECC ^{4, 6}	Enables ECC functionality of the IP	True, False	False
Enable Unaligned Read ⁶	Enables the unaligned read functionality of the Soft IP which shift the data out based on the current port input at the time of the read command	True, False	False
Others			
Enable Preserve Array	Allows the LRAM to preserve the array when the IP is set to low power mode	True, False	True
Enable GSR	Enables the GSR to reset this IP	True, False	True

Notes:

1. Total Number of bits between write and read ports must match. ($WADDR_DEPTH * WDATA_WIDTH = RADDR_DEPTH * RDATA_WIDTH$)
2. For mixed-width implementations, the allowable factor is 2 or 4. (that is, $WDATA_WIDTH = 2 * RDATA_WIDTH$, $4 * WDATA_WIDTH = RDATA_WIDTH$)
3. Byte-Enable can only be used for write port ≥ 16 bits.
4. Byte-Enable CANNOT be used with ECC.
5. Memory file is provided with respect to write port ($WADDR_DEPTH \times WDATA_WIDTH$)
6. ECC and/or Unaligned Read can only be used when $WDATA_WIDTH = RDATA_WIDTH = 32$.

2.12.1. Unaligned Access Feature

The Large RAM Module supports RISC-V compressed instruction chunks of data and shift them. If RISC-V needs to read the upper 16-bit of data in some address, it is very helpful to add support for shifting the upper 16 bits of output into the lower 16 bits, padding the upper bits with 0. Unaligned Read enables this feature; Table 2.26 shows possible combinations for Unaligned Read shifting.

Table 2.26. Unaligned Read shift function (Pseudo Dual Port LRAM)

<i>unaligned_i[1:0]</i>	<i>unaligned_i[2] = 0</i>	<i>unaligned_i[2] = 1</i>
00	DOx=x[31:0] (no shift)	DOx=x[31:0] (no shift)
01	DOx={8'b0,x[31:8]}	DOx={x[23:0],8'b0}
10	DOx={16'b0,x[31:16]}	DOx={x[15:0],16'b0}
11	DOx={24'b0,x[31:24]}	DOx={x[7:0],24'b0}

DOx refers to rd_data_o output port of the Large RAM.

Unaligned Read is only supported if WDATA_WIDTH = RDATA_WIDTH = 32.

You have the option to enable the output registers for Pseudo-Dual Port Large RAM (Large RAM_DP). The internal timing waveforms for Pseudo-Dual Port Large RAM (Large RAM_DP) with these options are shown in Figure 2.45 and Figure 2.46.

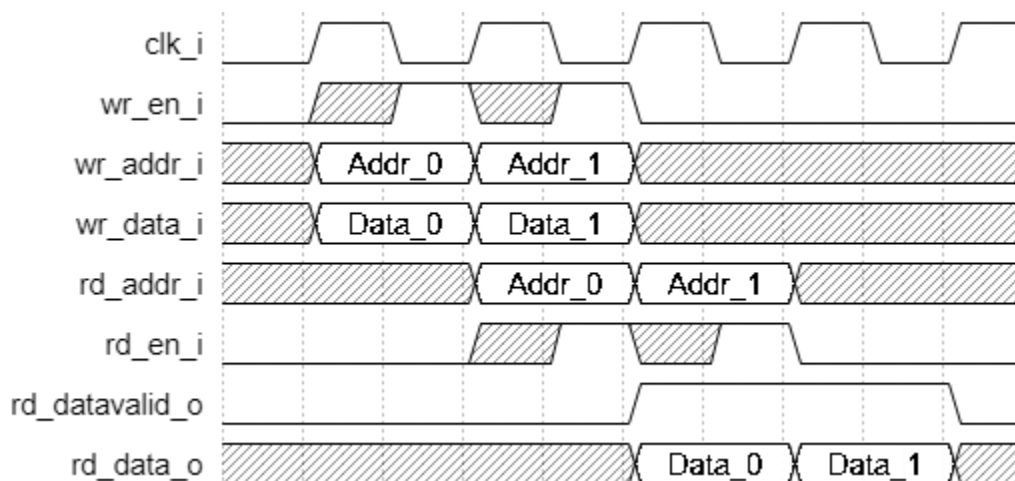


Figure 2.45. PSEUDO DUAL PORT Large RAM Timing Diagram, Without Output Registers

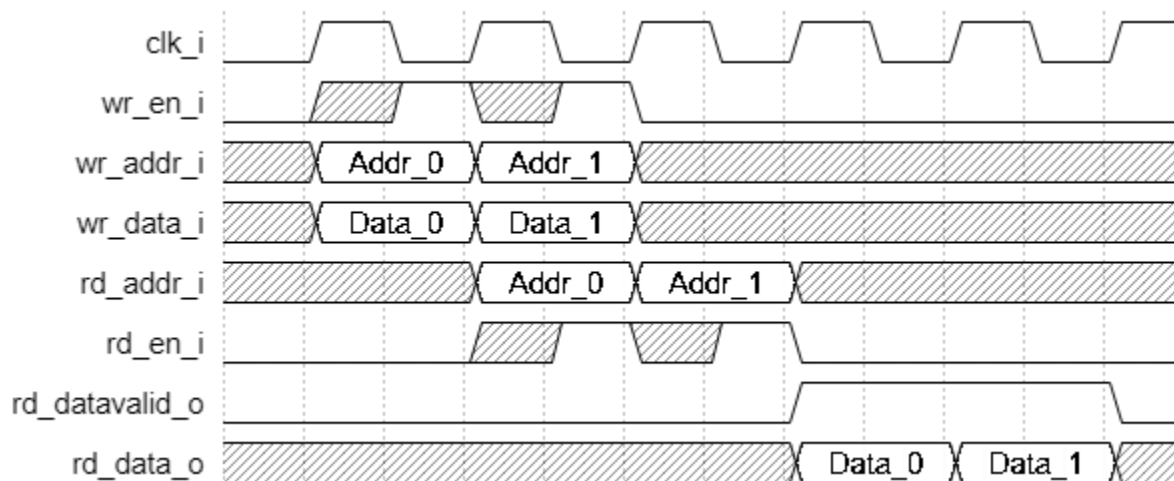


Figure 2.46. PSEUDO DUAL PORT Large RAM Timing Diagram, With Output Registers

2.13. True Dual-Port Large RAM (Large_RAM_DP True)

Lattice FPGA devices built on the Lattice Nexus platform support all the features of True-Dual Port Large RAM or Large RAM_DP_True. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.47](#).

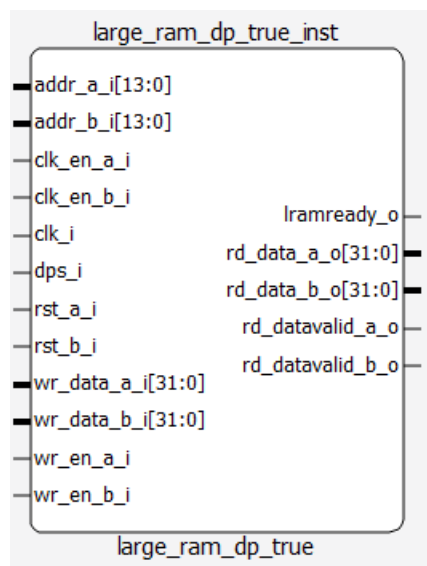


Figure 2.47. True Dual-Port Large RAM Module Generated by Module/IP Block Wizard

The various ports and their definitions for True Dual-Port Large RAM are listed in [Table 2.27](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.27. Large RAM-Based True Dual-Port Memory Port Definitions

Port Name	Direction	Width	Description
addr_a_i	Input	Port A Address Width	Port A address
addr_b_i	Input	Port B Address Width	Port B address
wr_data_a_i	Input	Port A Data Width	Port A write data
wr_data_b_i	Input	Port B Data Width	Port B write data
clk_i	Input	1	Clock
dps_i	Input	1	Dynamic Power Switching – when HIGH means the LRAM is in normal operation mode, when LOW means the LRAM is in low power mode, and write/read functions are stopped.
clk_en_a_i	Input	1	Port A Clock Enable
clk_en_b_i	Input	1	Port B Clock Enable
wr_en_a_i	Input	1	Port A Write Enable
wr_en_b_i	Input	1	Port B Write Enable
rst_a_i	Input	1	Port A Reset
rst_b_i	Input	1	Port B Reset
ben_a_i	Input	Port A Byte Enable Width	Port A Byte Enable port / Unaligned Access
ben_b_i	Input	Port B Byte Enable Width	Port B Byte Enable port / Unaligned Access
lramready_o	Output	1	When high means that the LRAM is ready for operation
rd_datavalid_a_o	Output	1	When high means the current output values of rd_data_a_o and ecc_errdet_a_o are valid.
rd_datavalid_b_o	Output	1	When high means the current output values of rd_data_b_o and ecc_errdet_b_o are valid.

Port Name	Direction	Width	Description
rd_data_a_o	Output	Port A Data Width	Port A Output Data
rd_data_b_o	Output	Port B Data Width	Port B Output Data
ecc_errdet_a_o	Output	2	Port A ECC output result: MSB – Two error detect LST – One error detect
ecc_errdet_b_o	Output	2	Port B ECC output result: MSB – Two error detect LST – One error detect
errdet_o	Output	1	When HIGH means an error has occurred in the LRAM

The various attributes available for the True Dual-Port Large RAM are listed in [Table 2.28](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.28. Large RAM-Based True Dual-Port Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Port A Address Depth ^{1, 2}	Port A Address Depth	2 – 131072	16384
Port A Data Width ^{1, 2}	Port A Data Width	1 – 64	32
Port B Address Depth ^{1, 2}	Port B Address depth. ^{1, 2}	2 – 131072	16384
Port B Data Width ^{1, 2}	Port B Data word width. ^{1, 2}	1 – 64	32
Enable Output Register (Port A)	Data Out port A (Q) can be registered or not using this selection.	True, False	True
Enable Output Register (Port B)	Data Out port B (Q) can be registered or not using this selection.	True, False	True
Reset Assertion (Port A)	Selection for the Port A Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Reset Assertion (Port B)	Selection for the Port B Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Reset Deassertion (Port A)	Selection for the Port A Reset Release to be Synchronous or Asynchronous to the Clock	async, sync	sync
Reset Deassertion (Port B)	Selection for the Port A Reset Release to be Synchronous or Asynchronous to the Clock	async, sync	sync
Enable Byte Enable (Port A) ^{3, 4}	Enables Byte Enable function for port A's write port.	True, False	False
Enable Byte Enable (Port B) ^{3, 4}	Enables Byte Enable function for port B's write port.	True, False	False
Initialization Attributes			
Memory Initialization ⁵	Allows you to initialize their memories to all 1s, 0s, or providing custom initialization through a memory file.	none, all 0s, all 1s, Memory file	none
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex

Miscellaneous Options			
Write/Read Related			
Enable ECC ^{4, 6}	Enables ECC functionality of the IP	True, False	False
Enable Write-Through A	Enable Write-Through mode for Port A	True, False	False
Enable Write-Through B	Enable Write-Through mode for Port B	True, False	False
Enable Unaligned Read ⁶	Enables the unaligned read functionality of the Soft IP which shift the data out based on the current port input at the time of the read command	True, False	False
Others			
Enable Preserve Array	Allows the LRAM to preserve the array when the IP is set to low power mode	True, False	True
Enable GSR	Enables the GSR to reset this IP	True, False	True

Notes:

1. For mixed width configurations total bit size must be equal (that is, ADEPTH*ADATA = BDEPTH*BDATA)
2. For mixed width configuration the ratio between ADATA and BDATA must be 2 or 4.
3. Byte-enable can only be enabled when ADATA >= 16 for port A, and/or BDATA >= 16 port B.
4. Byte-enable and ECC cannot be used simultaneously
5. Initialization files should be provided with respect to ADDR_DEPTH_A x DATA_WIDTH_A.
6. ECC and Unaligned Read can only be enabled when ADATA = BDATA = 32

2.13.1. Unaligned Access Feature

The Large RAM Module supports RISC-V compressed instruction chunks of data and shift them. If RISC-V needs to read the upper 16-bit of data in some address, it is very helpful to add support for shifting the upper 16 bits of output into the lower 16 bits, padding the upper bits with 0. Unaligned Read enables this feature; Table 2.29 shows possible combinations for Unaligned Read shifting. The unaligned read pins are shared with the `ben_[x]_i` ports for True Dual Port Large RAM. Given this, you should be careful in changing the value of these signals. The port functions as byte-enabled during write-access and unaligned read during read-access.

Table 2.29. Unaligned Read shift function (True Dual Port LRAM)

<code>ben_[x]_i[1:0]</code>	<code>ben_[x]_i[2] = 0</code>	<code>ben_[x]_i[2] = 1</code>
00	DOx=x[31:0] (no shift)	DOx=x[31:0] (no shift)
01	DOx={8'b0,x[31:8]}	DOx={x[23:0],8'b0}
10	DOx={16'b0,x[31:16]}	DOx={x[15:0],16'b0}
11	DOx={24'b0,x[31:24]}	DOx={x[7:0],24'b0}

DO[x] refers to `rd_data_[x]_o` output port of the Large RAM. Where [x] can be port a or b.

Unaligned Read is only supported if `DATA_WIDTH_A = DATA_WIDTH_B = 32`.

You have the option to enable the output registers for True Dual Port Large RAM. The internal timing waveforms for True Dual Port Large RAM with these options are shown in Figure 2.48 and Figure 2.49. A sample waveform form for write-through without registered output is also provided in Figure 2.50.

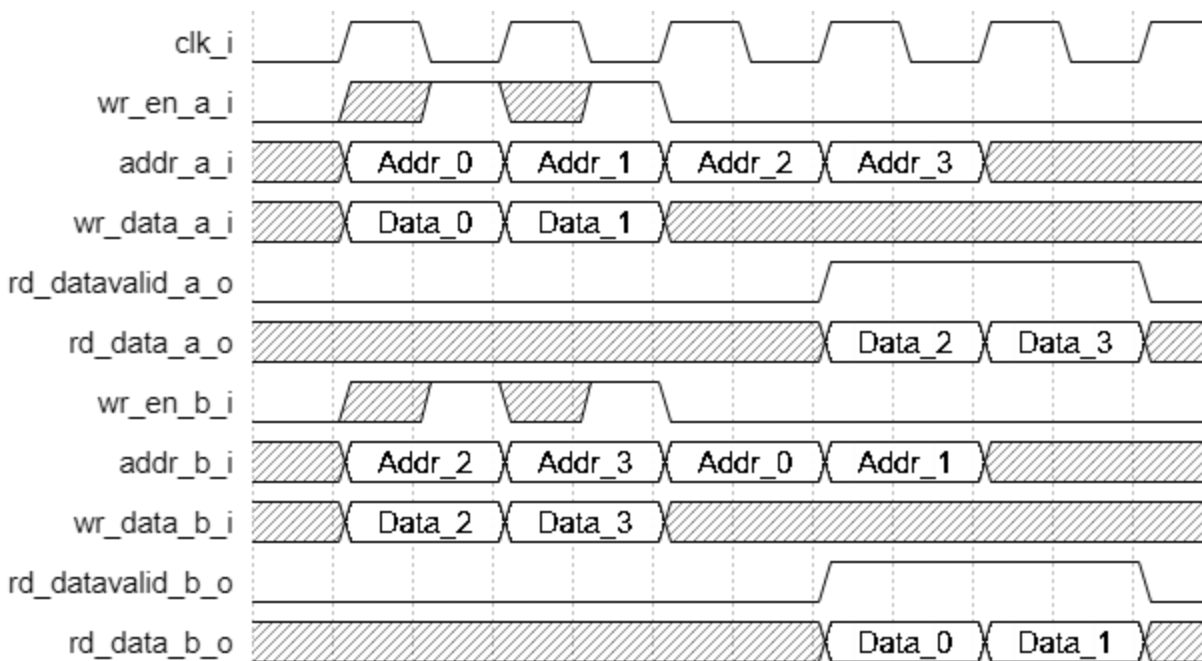


Figure 2.48. Single Port LRAM Timing Waveform in NORMAL Mode, Without Output Registers

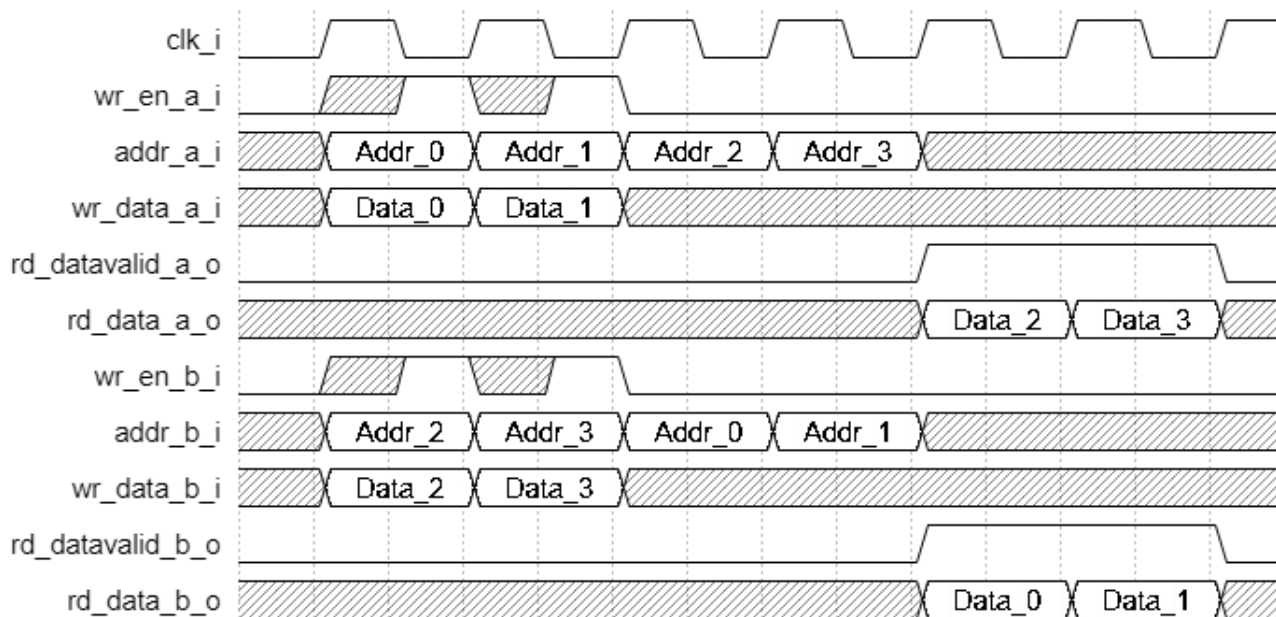


Figure 2.49. Single Port RAM Timing Waveform in NORMAL Mode, With Output Registers

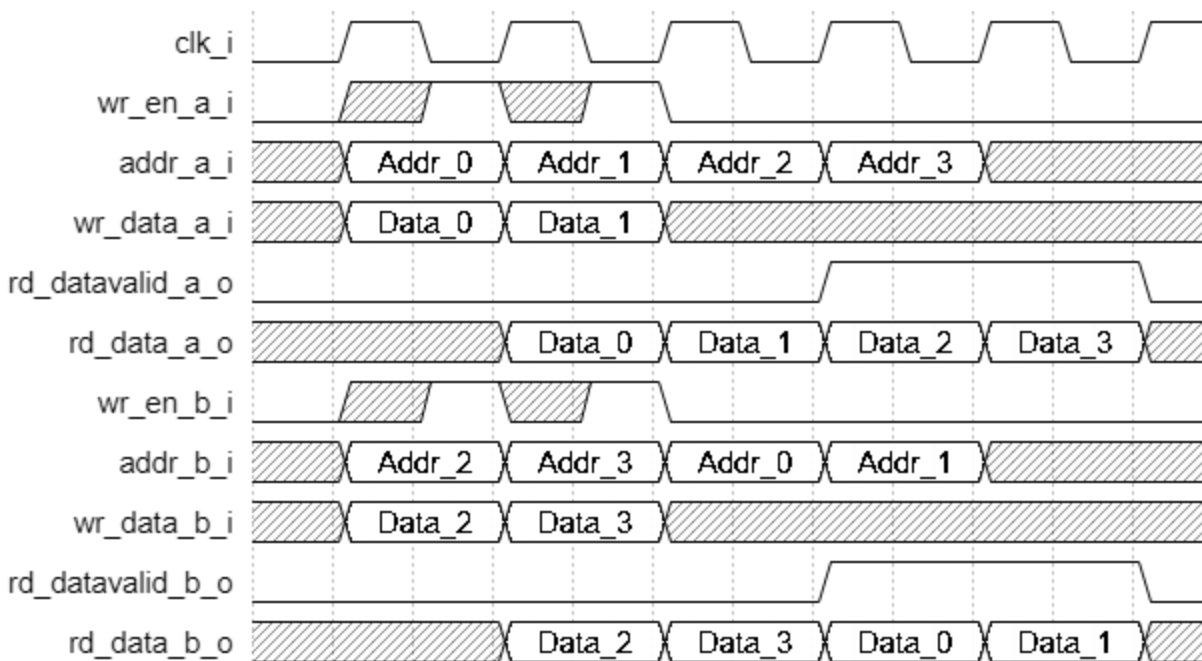


Figure 2.50. Single Port LRAM Timing Waveform in Write-Through Mode, Without Output Registers

2.14. Large Read-Only Memory (Large_ROM)

Lattice FPGA devices built on the Lattice Nexus platform support all the features of Large Read Only Memory Module. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirement.

Module/IP Block Wizard generates the memory module shown in [Figure 2.51](#).

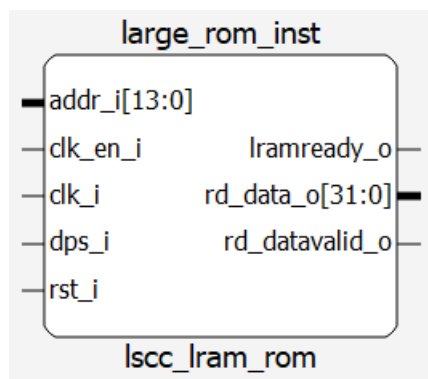


Figure 2.51. Large Read Only Memory Module Generated by Module/IP Block Wizard

The various ports and their definitions for Read Only Memory are listed in [Table 2.30](#). The table lists the corresponding ports for the module generated by Module/IP Block Wizard.

Table 2.30. Large Read Only Memory Port Definitions

Port Name	Direction	Width	Description
rd_clk_i	Input	1	Read Clock input
rst_i	Input	1	Output Reset
dps_i	Input	1	Dynamic Power Switching – when HIGH means the LRAM is in normal operation mode, when LOW means the LRAM is in low power mode, and read functions are stopped.
rd_clk_en_i	Input	1	Read Clock Enable
rd_out_clk_en_i	Input	1	Read Out Clock Enable
rd_en_i	Input	1	Read Enable
rd_addr_i	Input	Read Address Width	Read Address
unaligned_i	Input	4	Unaligned Access Port
lramready_o	Output	1	When high means that the LRAM is ready for operation
rd_datavalid_o	Output	1	When high means the current output values of rd_data_o and ecc_errdet_o are valid.
rd_data_o	Output	Read Data Width	Read Data
errdet_o	Output	1	When HIGH means an error has occurred in the LRAM
ecc_errdet_o	Output	2	ECC output result: MSB – Two error detect LST – One error detect

The various attributes available for the Read Only Memory (ROM) are listed in [Table 2.31](#). Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.31. Large Read Only Memory Attribute Definitions

Attribute	Description	Values	Default Value
General Attributes			
Read Port Address Depth	Read Port Address Depth	2 - 131072	16384
Read Port Data Width	Read Port Data Width	1 - 64	32
Enable Output Register	Data Out (Q) can be registered or not using this selection.	True, False	True
Enable Output ClockEn	Clock Enable for the output clock of (this option requires enabling output register)	True, False	False
Reset Assertion	Selection for the Reset to be Synchronous or Asynchronous to the Clock	async, sync	sync
Initialization Attributes			
Memory Initialization	Allows you to initialize the memory by providing a custom initialization through a memory file.	Memory file	Memory file
Memory File	When Memory file is selected, you can browse to the memory file for custom initialization of RAM.	—	none
Memory File Format	This option allows you to select if the memory file is formatted as Binary or Hex. (See the Initialization File Formats section for details on the different formats.)	binary, hex	hex
Miscellaneous Options			
Write/Read Related			
Enable ECC*	Enables ECC functionality of the IP	True, False	False
Enable Unaligned Read*	Enables the unaligned read functionality of the Soft IP which shift the data out based on the current port input at the time of the read command	True, False	False
Others			
Enable Preserve Array	Allows the LRAM to preserve the array when the IP is set to low power mode	True, False	True
Enable GSR	Enables the GSR to reset this IP	True, False	True

* ECC and Unaligned Read can only be enabled when RDATA_WIDTH = 32

2.14.1. Unaligned Access Feature

The Large RAM Module supports RISC-V compressed instruction chunks of data and shift them. If RISC-V needs to read the upper 16-bit of data in some address, it is very helpful to add support for shifting the upper 16 bits of output into the lower 16 bits, padding the upper bits with to 0. Unaligned Read enables this feature; [Table 2.32](#) shows possible combinations for Unaligned Read shifting.

Table 2.32. Unaligned Read Shift Function (LROM)

<i>unaligned_i[1:0]</i>	<i>unaligned_i [2] = 0</i>	<i>unaligned_i [2] = 1</i>
00	DOx=x[31:0] (no shift)	DOx=x[31:0] (no shift)
01	DOx={8'b0,x[31:8]}	DOx={x[23:0],8'b0}
10	DOx={16'b0,x[31:16]}	DOx={x[15:0],16'b0}
11	DOx={24'b0,x[31:24]}	DOx={x[7:0],24'b0}

DOx refers to rd_data_o output port of the Large RAM.

Unaligned Read is only supported if RDATA_WIDTH = 32.

The Large Read Only Memory timing waveforms are shown in [Figure 2.52](#) and [Figure 2.53](#).



Figure 2.52. Large ROM Timing Diagram, Without Output Registers

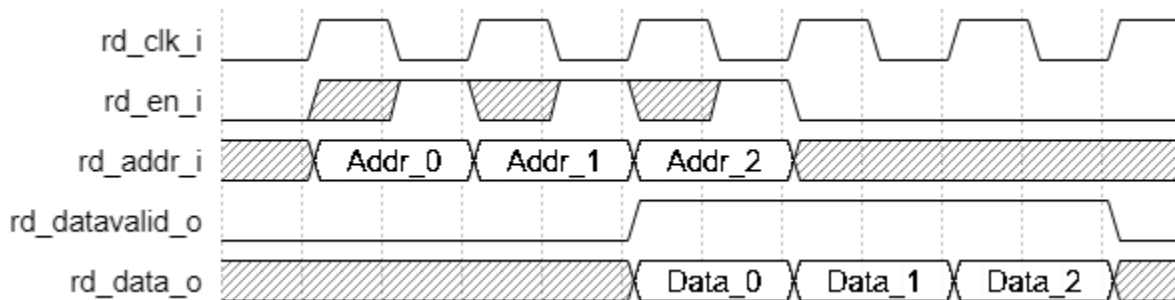


Figure 2.53. Large ROM Timing Diagram, With Output Registers

3. IP Generation

Module/IP Block Wizard in Lattice Radiant Software allows you to generate, create, or open modules for the target device. From the Lattice Radiant Software, select the IP Catalog tab as shown in [Figure 3.1](#).

The left pane of the IP Catalog window displays the module tree. You can utilize Module/IP on Local to 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 (lookup tables) as required.

The memory modules are categorized as Memory_Modules with EBR_Components as sub-category. The available memory modules in the Lattice Radiant Software IP catalog are shown below

The right pane of the window shows the description of the selected module and provides links to the documentation.

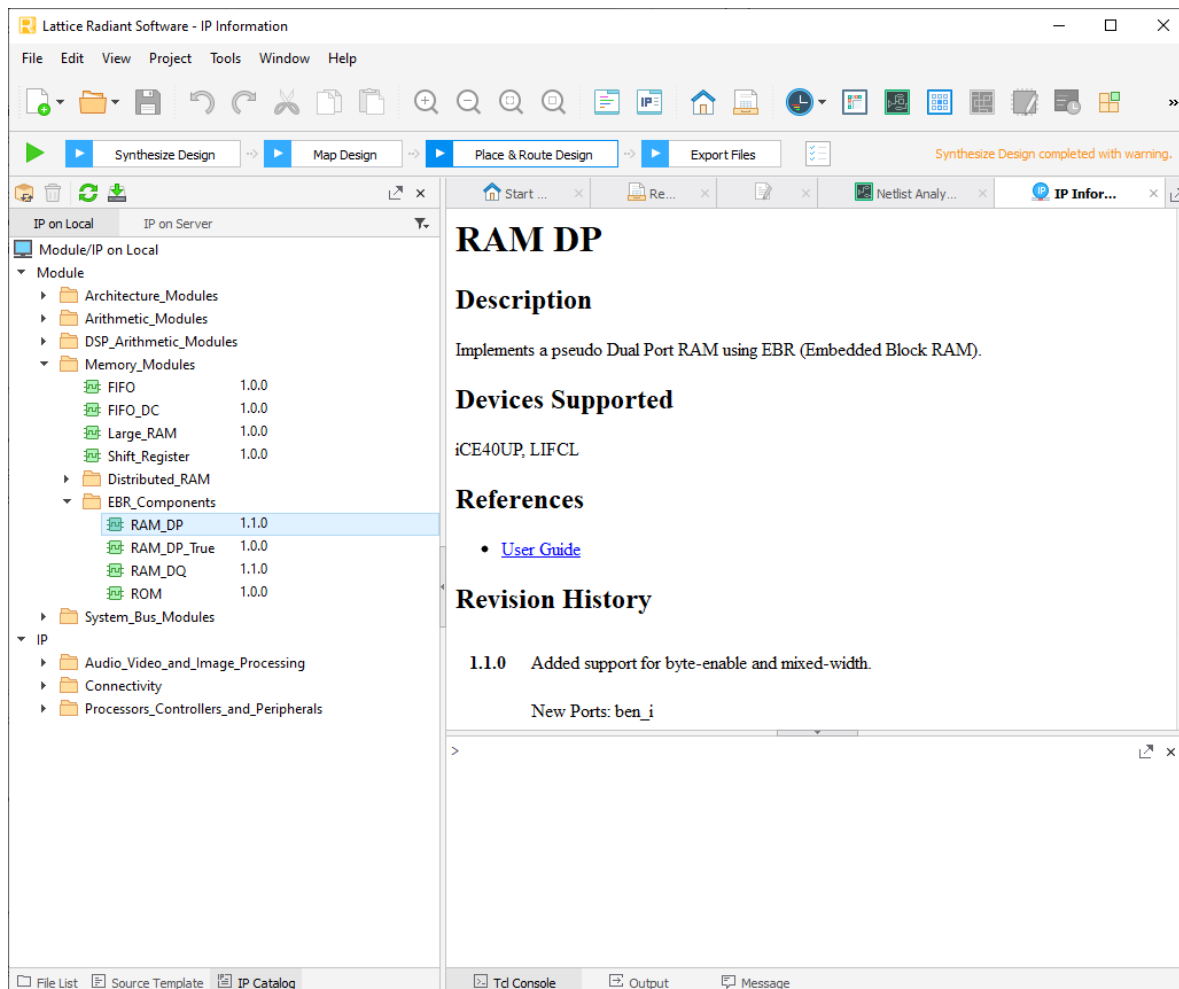


Figure 3.1. MemoryModules Under Module/IP on Local in the Lattice Radiant Software

The following section shows 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 of size 512 x 18:

1. Double-click RAM_DP under Memory_Modules > EBR_Components.
2. This opens the Module/IP Block Wizard. In the **Name & Location** page, fill out the information of the module to generate as shown in [Figure 3.2](#). Click **Next**.

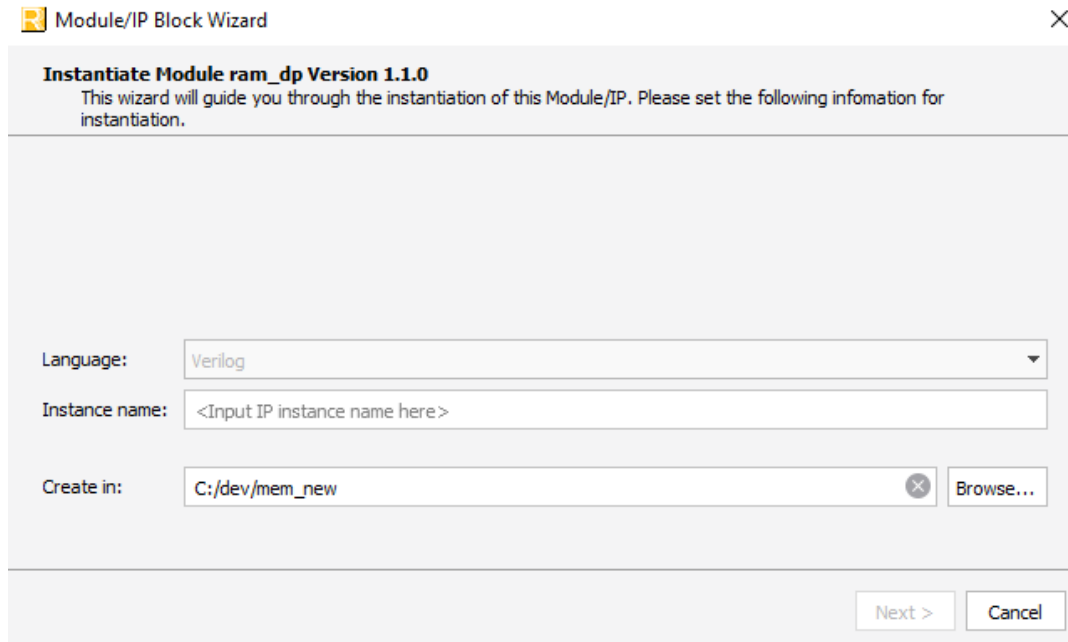


Figure 3.2. Example: Generating Pseudo Dual Port RAM (RAM_DP) Using Module/IP Block Wizard

3. In the **IP Configuration** page, customize the Dual Port RAM by selecting options as shown in Figure 3.3.

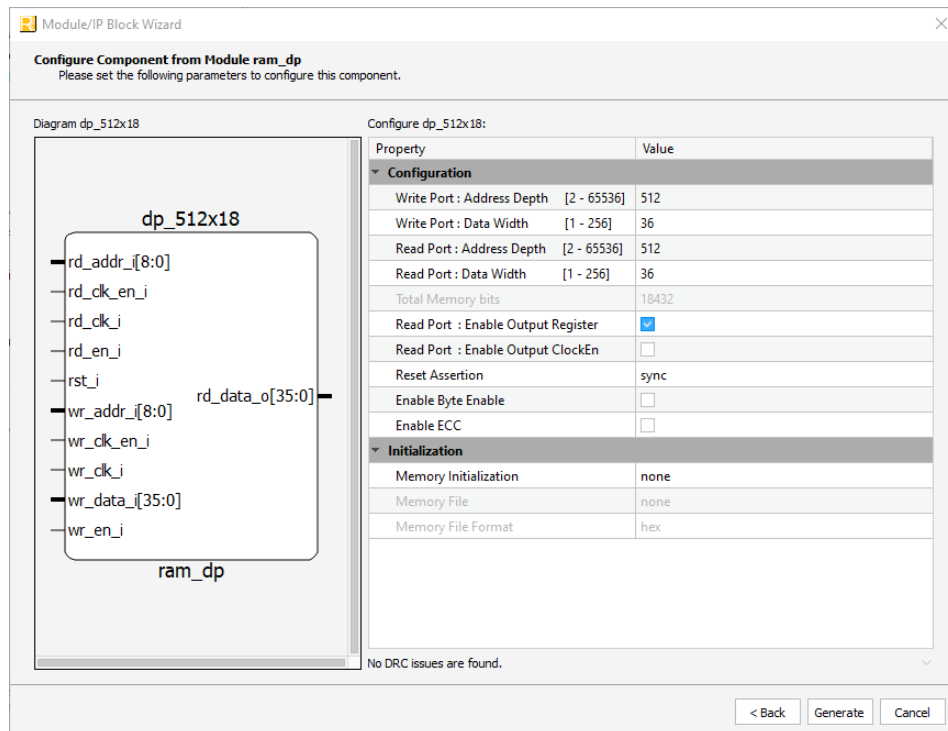


Figure 3.3. Example: Generating Pseudo Dual Port RAM (RAM_DP) Module Customization

4. When all the options are set, click **Generate**.

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

4. PMI Support

The following sections discuss the different PMI modules which can be utilized for entering custom parameters for the single-port and pseudo dual-port RAM. If parameters are left unspecified during module instantiation, default values are used. Some parameters here are unused, and are added to maintain backward compatibility with older devices.

4.1. pmi_ram_dq

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dq (Single Port Memory Module).

Table 4.1. Single Port PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	Data	Bus	(pmi_data_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Table 4.2. Single Port PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_addr_depth	Address depth of the read and write port	2—<Max that can fit in the device>	512
pmi_addr_width	Address width of the read and write port	1—<Max that can fit in the device>	9
pmi_data_width	Data word width of the Read and Write port	1–512	18
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	“reg”, “noreg”	“reg”
pmi_gsr	Sets if the global set/reset is enabled.	“enabled”, “disable”	“disable”
pmi_resetmode	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	“async”, “sync”	“sync”
pmi_optimization ¹	Describes the synthesis directive if optimizing for area or speed	“speed”, “area”	“speed”
pmi_init_file	When Memory file is selected, you should set this parameter to the memory file.	<file_path>	“none”
pmi_init_file_format	This option allows you to select if the memory file is formatted as Binary, Hex. (See the Initializing Memory section for details on different formats.)	“binary”, “hex”	“binary”
pmi_write_mode	Defines the write mode of the module.	“normal”	“normal”
pmi_family ²	This option selects the product family used.	“iCE40UP”, “LIFCL”, “LFD2NX”, “LFCPNX” “common”	“common”
module_type ¹	Refers to the type of pmi module.	“pmi_ram_dq”	“pmi_ram_dq”

Notes:

1. Unused
2. Use pmi_family = “common” if initialization is needed. pmi_family = “common”; requires 30-bits to be properly implemented (pmi_addr_depth x pmi_data_width >= 30).

Verilog pmi_ram_dq Definition

```
module pmi_ram_dq
#(parameter pmi_addr_depth = 512,
  parameter pmi_addr_width = 9
  parameter pmi_data_width = 18,
  parameter pmi_regmode = "reg",
  parameter pmi_gsr = "disable",
  parameter pmi_resetmode = "sync",
  parameter pmi_optimization = "speed",
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_write_mode = "normal",
  parameter pmi_family = "common",
  parameter module_type = "pmi_ram_dq")

  (input [(pmi_data_width-1):0] Data,
    input [(pmi_addr_width-1):0] Address,
    input Clock,
    input ClockEn,
    input WE,
    input Reset,
    output [(pmi_data_width-1):0];

endmodule // pmi_ram_dq
```

VHDL pmi_ram_dq Definition

```
component pmi_ram_dq is
  generic (
    pmi_addr_depth : integer := 512;
    pmi_addr_width : integer := 9;
    pmi_data_width : integer := 18;
    pmi_regmode : string := "reg";
    pmi_gsr : string := "disable";
    pmi_resetmode : string := "sync";
    pmi_optimization : string := "speed";
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_write_mode : string := "normal";
    pmi_family : string := "EC";
    module_type : string := "pmi_ram_dq"
  );
  port (
    Data : in std_logic_vector((pmi_data_width-1) downto 0);
    Address : in std_logic_vector((pmi_addr_width-1) downto 0);
    Clock: in std_logic;
    ClockEn: in std_logic;
    WE: in std_logic;
    Reset: in std_logic;
    Q : out std_logic_vector((pmi_data_width-1) downto 0)
  );
end component pmi_ram_dq;
```

4.2. pmi_ram_dq_be

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dq_be (Single Port Memory Module with Byte-Enable).

Table 4.3. Single Port with Byte-Enable PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	Data	Bus	(pmi_data_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WE	Bit	N/A
I	ByteEn	Bus	(pmi_data_width/ pmi_byte_size-1):0
O	Q	Bus	(pmi_data_width - 1):0

Table 4.4. Single Port with Byte-Enable PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_addr_depth	Address depth of the read and write port	2-<Max that can fit in the device>	512
pmi_addr_width	Address width of the read and write port	1-<Max that can fit in the device>	9
pmi_data_width	Data word width of the Read and Write port	1-512	18
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_gsr	Sets if the global set/reset is enabled.	"enabled", "disable"	"disable"
pmi_resetmode	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	"async", "sync"	"sync"
pmi_optimization*	Describes the synthesis directive if optimizing for area or speed	"speed", "area"	"speed"
pmi_init_file ¹	When Memory file is selected, you should set this parameter to the memory file.	<file_path>	"none"
pmi_init_file_format*	This option allows you to select if the memory file is formatted as Binary, Hex. (See the Initializing Memory section for details on different formats.)	"binary", "hex"	"binary"
pmi_write_mode	Defines the write mode of the module.	"normal"	"normal"
pmi_family ²	This option selects the product family used.	"iCE40UP", "LIFCL", "LFD2NX", "LFCPNX"	"common"
pmi_byte_size	Byte size. For LIFCL devices value should be 9 if pmi_data_width is divisible by 9, else 8. Also use 8 for iCE40UP devices.	8, 9	9
module_type ¹	Refers to the type of pmi module.	"pmi_ram_dq_be"	"pmi_ram_dq_be"

Notes:

1. Unused. WARNING: Do not change the value of pmi_init_file, this PMI does not support initialization.
2. Use pmi_family = "<device>"

Verilog pmi_ram_dq_be Definition

```
module pmi_ram_dq_be
#(parameter pmi_addr_depth = 512,
  parameter pmi_addr_width = 9
  parameter pmi_data_width = 18,
  parameter pmi_regmode = "reg",
  parameter pmi_gsr = "disable",
  parameter pmi_resetmode = "sync",
  parameter pmi_optimization = "speed",
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_write_mode = "normal",
  parameter pmi_family = "common",
  parameter pmi_byte_size = 9,
  parameter module_type = "pmi_ram_dq")

  (input [(pmi_data_width-1):0] Data,
    input [(pmi_addr_width-1):0] Address,
    input Clock,
    input ClockEn,
    input WE,
    input Reset,
    input [(pmi_data_width/pmi_byte_size-1):0] ByteEn,
    output [(pmi_data_width-1):0];

endmodule // pmi_ram_dq_be
```

VHDL pmi_ram_dq_be Definition

```
component pmi_ram_dq_be is
  generic (
    pmi_addr_depth : integer := 512;
    pmi_addr_width : integer := 9;
    pmi_data_width : integer := 18;
    pmi_regmode : string := "reg";
    pmi_gsr : string := "disable";
    pmi_resetmode : string := "sync";
    pmi_optimization : string := "speed";
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_write_mode : string := "normal";
    pmi_family : string := "common";
    pmi_byte_size : integer := 9;
    module_type : string := "pmi_ram_dq"
  );
  port (
    Data : in std_logic_vector((pmi_data_width-1) downto 0);
    Address : in std_logic_vector((pmi_addr_width-1) downto 0);
    Clock: in std_logic;
    ClockEn: in std_logic;
    WE: in std_logic;
    Reset: in std_logic;
    ByteEn : in std_logic_vector((pmi_data_width/pmi_byte_size -1) downto 0);
    Q : out std_logic_vector((pmi_data_width-1) downto 0)
  );
end component pmi_ram_dq_be;
```

4.3. pmi_ram_dp

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp (Pseudo Dual Port Memory Module).

Table 4.5. Pseudo Dual Port PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_wr_addr_width - 1):0
I	RdAddress	Bus	(pmi_rd_addr_width - 1):0
I	Data	Bus	(pmi_wr_data_width - 1):0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_rd_data_width - 1):0

Table 4.6. Pseudo Dual Port PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_wr_addr_depth	Write Port Address depth.	2-<Max that can fit in the device>	512
pmi_wr_addr_width	Write Port Address width. Equal to $\log_2(\text{pmi_wr_addr_depth})$	1-<Max that can fit in the device>	9
pmi_wr_data_width	Write Port Data word width.	1-512	18
pmi_rd_addr_depth	Read Port Address depth.	2-<Max that can fit in the device>	512
pmi_rd_addr_width	Read Port Address width. Equal to $\log_2(\text{pmi_rd_addr_depth})$	1-<Max that can fit in the device>	9
pmi_rd_data_width	Read Port Data word width.	1-512	18
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_gsr	Sets if the global set/reset is enabled.	"enabled", "disable"	"disable"
pmi_resetmode ¹	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	"async", "sync"	"sync"
pmi_optimization	Describes the synthesis directive if optimizing for area or speed	"speed", "area"	"speed"
pmi_init_file	When Memory file is selected, user should set this parameter to the memory file.	<file_path>	"none"
pmi_init_file_format	This option allows users to select if the memory file is formatted as Binary, Hex. (Check Sec.5 for details on different formats)	"binary", "hex"	"binary"
pmi_family ²	Defines the FPGA family being used in the module	"iCE40UP", "LIFCL", "LFD2NX", "LFCPNX", "common"	"common"
module_type ¹	Refers to the type of pmi module.	"pmi_ram_dp"	"pmi_ram_dp"

Notes:

1. Unused.
2. Use pmi_family = "common" if initialization is needed. pmi_family = "common", requires 30-bits to be properly implemented ($\text{pmi_wr_addr_depth} \times \text{pmi_wr_data_width} \geq 30$), use pmi_family = "<device>" if mixed-width is needed. Mixed-width + initialization is NOT supported.

Verilog pmi_ram_dp Definition

```
module pmi_ram_dp
#(parameter pmi_wr_addr_depth = 512,
  parameter pmi_wr_addr_width = 9,
  parameter pmi_wr_data_width = 18,
  parameter pmi_rd_addr_depth = 512,
  parameter pmi_rd_addr_width = 9,
  parameter pmi_rd_data_width = 18,
  parameter pmi_regmode = "reg",
  parameter pmi_gsr = "disable",
  parameter pmi_resetmode = "sync",
  parameter pmi_optimization = "speed",
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_family = "common",
  parameter module_type = "pmi_ram_dp")

  (input [(pmi_wr_data_width-1):0] Data,
   input [(pmi_wr_addr_width-1):0] WrAddress,
   input [(pmi_rd_addr_width-1):0] RdAddress,
   input  WrClock,
   input  RdClock,
   input  WrClockEn,
   input  RdClockEn,
   input  WE,
   input  Reset,
   output [(pmi_rd_data_width-1):0] Q);

endmodule // pmi_ram_dp
```

VHDL pmi_ram_dp Definition

```
component pmi_ram_dp is
  generic (
    pmi_wr_addr_depth : integer := 512;
    pmi_wr_addr_width : integer := 9;
    pmi_wr_data_width : integer := 18;
    pmi_rd_addr_depth : integer := 512;
    pmi_rd_addr_width : integer := 9;
    pmi_rd_data_width : integer := 18;
    pmi_regmode : string := "reg";
    pmi_gsr : string := "disable";
    pmi_resetmode : string := "sync";
    pmi_optimization : string := "speed";
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_family : string := "common";
    module_type : string := "pmi_ram_dp"
  );
  port (
    Data : in std_logic_vector((pmi_wr_data_width-1) downto 0);
    WrAddress : in std_logic_vector((pmi_wr_addr_width-1) downto 0);
    RdAddress : in std_logic_vector((pmi_rd_addr_width-1) downto 0);
    WrClock: in std_logic;
    RdClock: in std_logic;
    WrClockEn: in std_logic;
    RdClockEn: in std_logic;
    WE: in std_logic;
    Reset: in std_logic;
    Q : out std_logic_vector((pmi_rd_data_width-1) downto 0)
  );
end component pmi_ram_dp;
```


4.4. pmi_ram_dp_be

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp_be (Pseudo Dual Port Memory Module).

Table 4.7. Pseudo Dual Port with Byte Enable PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_wr_addr_width - 1):0
I	RdAddress	Bus	(pmi_rd_addr_width - 1):0
I	Data	Bus	(pmi_wr_data_width - 1):0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
I	ByteEn	Bus	(pmi_data_width/ pmi_byte_size-1):0
O	Q	Bus	(pmi_rd_data_width - 1):0

Table 4.8. Pseudo Dual Port with Byte Enable PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_wr_addr_depth	Write Port Address depth.	2–<Max that can fit in the device>	512
pmi_wr_addr_width	Write Port Address width. Equal to $\log_2(\text{pmi_wr_addr_depth})$	1–<Max that can fit in the device>	9
pmi_wr_data_width	Write Port Data word width.	1–512	18
pmi_rd_addr_depth	Read Port Address depth.	2–<Max that can fit in the device>	512
pmi_rd_addr_width	Read Port Address width. Equal to $\log_2(\text{pmi_rd_addr_depth})$	1–<Max that can fit in the device>	9
pmi_rd_data_width	Read Port Data word width.	1–512	18
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_gsr	Sets if the global set/reset is enabled.	"enabled", "disable"	"disable"
pmi_resetmode ¹	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	"async", "sync"	"sync"
pmi_optimization	Describes the synthesis directive if optimizing for area or speed	"speed", "area"	"speed"
pmi_init_file ¹	When Memory file is selected, user should set this parameter to the memory file.	<file_path>	"none"
pmi_init_file_format ¹	This option allows users to select if the memory file is formatted as Binary, Hex. (Check Sec.5 for details on different formats)	"binary", "hex"	"binary"
pmi_family ²	Defines the FPGA family being used in the module	"ice40UP", "LIFCL", "LFD2NX", "LFCPNX"	"common"
pmi_byte_size	Byte size. For LIFCL devices value should be 9 if pmi_data_width is divisible by 9, else 8. Also use 8 for iCE40UP devices.	8, 9	9
module_type ¹	Refers to the type of pmi module.	"pmi_ram_dp_be"	"pmi_ram_dp_be"

Notes:

1. Unused. WARNING: Do not change the value of pmi_init_file, this PMI does not support initialization.
2. Use pmi_family = "<device>", mixed-width byte-enable only applies to W/R ≤ 4 for Lattice Nexus devices and W/R ≤ 2 for iCE40 UltraPlus.

Verilog pmi_ram_dp_be Definition

```
module pmi_ram_dp_be
#(parameter pmi_wr_addr_depth = 512,
  parameter pmi_wr_addr_width = 9,
  parameter pmi_wr_data_width = 18,
  parameter pmi_rd_addr_depth = 512,
  parameter pmi_rd_addr_width = 9,
  parameter pmi_rd_data_width = 18,
  parameter pmi_regmode = "reg",
  parameter pmi_gsr = "disable",
  parameter pmi_resetmode = "sync",
  parameter pmi_optimization = "speed",
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_family = "common",
  parameter pmi_byte_size = 9,
  parameter module_type = "pmi_ram_dp")

  (input [(pmi_wr_data_width-1):0] Data,
   input [(pmi_wr_addr_width-1):0] WrAddress,
   input [(pmi_rd_addr_width-1):0] RdAddress,
   input  WrClock,
   input  RdClock,
   input  WrClockEn,
   input  RdClockEn,
   input  WE,
   input  Reset,
   input [(pmi_wr_data_width/pmi_byte_size)-1:0] ByteEn,
   output [(pmi_rd_data_width-1):0] Q);

endmodule // pmi_ram_dp_be
```

VHDL pmi_ram_dp_be Definition

```
component pmi_ram_dp_be is
  generic (
    pmi_wr_addr_depth : integer := 512;
    pmi_wr_addr_width : integer := 9;
    pmi_wr_data_width : integer := 18;
    pmi_rd_addr_depth : integer := 512;
    pmi_rd_addr_width : integer := 9;
    pmi_rd_data_width : integer := 18;
    pmi_regmode : string := "reg";
    pmi_gsr : string := "disable";
    pmi_resetmode : string := "sync";
    pmi_optimization : string := "speed";
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_family : string := "common";
    pmi_byte_size : integer := 9;
    module_type : string := "pmi_ram_dp_be"
  );
  port (
    Data : in std_logic_vector((pmi_wr_data_width-1) downto 0);
    WrAddress : in std_logic_vector((pmi_wr_addr_width-1) downto 0);
    RdAddress : in std_logic_vector((pmi_rd_addr_width-1) downto 0);
    WrClock: in std_logic;
    RdClock: in std_logic;
    WrClockEn: in std_logic;
    RdClockEn: in std_logic;
    WE: in std_logic;
    Reset: in std_logic;
    ByteEn : in std_logic_vector((pmi_wr_data_width/pmi_byte_size -1) downto 0);
    Q : out std_logic_vector((pmi_rd_data_width-1) downto 0)
  );
end component pmi_ram_dp_be;
```

4.5. pmi_rom

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_rom (Read Only Memory Module).

Table 4.9. Read Only Memory PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	OutClock	Bus	N/A
I	OutClockEn	Bus	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Table 4.10. Read Only Memory PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_addr_depth	Address depth.	2-<Max that can fit in the device>	512
pmi_addr_width	Address width. Equal to $\log_2(\text{pmi_wr_addr_depth})$	1-<Max that can fit in the device>	9
pmi_data_width	Data word width.	1-512	18
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_gsr	Sets if the global set/reset is enabled.	"enabled", "disable"	"disable"
pmi_resetmode	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	"async", "sync"	"sync"
pmi_optimization ¹	Describes the synthesis directive if optimizing for area or speed	"speed", "area"	"speed"
pmi_init_file	When Memory file is selected, user should set this parameter to the memory file.	<file_path>	"none"
pmi_init_file_format	This option allows users to select if the memory file is formatted as Binary, Hex. (Check Sec.5 for details on different formats)	"binary", "hex"	"binary"
pmi_family ²	Defines the FPGA family being used in the module	"iCE40UP", "LIFCL", "LFD2NX", "LFCPNX"	"common"
module_type ¹	Refers to the type of pmi module.	"pmi_rom"	"pmi_rom"

Notes:

1. Unused
2. Use only pmi_family = "common". This pmi requires 30-bits to be properly implemented (pmi_addr_depth x pmi_data_width >= 30)

Verilog pmi_rom Definition

```
module pmi_rom # (  
    parameter pmi_addr_depth = 512,  
    parameter pmi_addr_width = 9  
    parameter pmi_data_width = 8,  
    parameter pmi_regmode = "reg",  
    parameter pmi_gsr = "disable",  
    parameter pmi_resetmode = "sync",  
    parameter pmi_optimization = "speed",  
    parameter pmi_init_file = "none",  
    parameter pmi_init_file_format = "binary",  
    parameter pmi_family = "common",  
    parameter module_type = "pmi_rom"  
) (input [(pmi_addr_width-1):0] Address,  
    input OutClock,  
    input OutClockEn,  
    input Reset,  
    output [(pmi_data_width-1):0] Q)  
;  
  
endmodule // pmi_rom
```

VHDL pmi_rom Definition

```
component pmi_rom is  
    generic (  
        pmi_addr_depth : integer := 512;  
        pmi_addr_width : integer := 9;  
        pmi_data_width : integer := 8;  
        pmi_regmode : string := "reg";  
        pmi_gsr : string := "disable";  
        pmi_resetmode : string := "sync";  
        pmi_optimization : string := "speed";  
        pmi_init_file : string := "none";  
        pmi_init_file_format : string := "binary";  
        pmi_family : string := "common";  
        module_type : string := "pmi_rom"  
    );  
    port (  
        Address : in std_logic_vector((pmi_addr_width-1) downto 0);  
        OutClock: in std_logic;  
        OutClockEn: in std_logic;  
        Reset: in std_logic;  
        Q : out std_logic_vector((pmi_data_width-1) downto 0)  
    );  
end component pmi_rom;
```

4.6. pmi_ram_dp_true

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp (Pseudo Dual Port Memory Module).

Table 4.11. True Dual Port PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	AddressA	Bus	(pmi_addr_width_a - 1):0
I	AddressB	Bus	(pmi_addr_width_b - 1):0
I	DataInA	Bus	(pmi_data_width_a - 1):0
I	DataInB	Bus	(pmi_data_width_b - 1):0
I	ClockA	Bit	N/A
I	ClockB	Bit	N/A
I	ClockEnA	Bit	N/A
I	ClockEnB	Bit	N/A
I	WrA	Bit	N/A
I	WrB	Bit	N/A
I	ResetA	Bit	N/A
I	ResetB	Bit	N/A
O	QA	Bus	(pmi_data_width_a - 1):0
O	QB	Bus	(pmi_data_width_b - 1):0

Table 4.12. True Dual Port PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_addr_depth_a	Port A Address depth.	2-<Max that can fit in the device>	512
pmi_addr_width_a	Port A Address width. Equal to $\log_2(\text{pmi_addr_depth_a})$.	1-<Max that can fit in the device>	9
pmi_data_width_a	Port A Data word width.	1-512	18
pmi_addr_depth_b	Port B Address depth.	2-<Max that can fit in the device>	512
pmi_addr_width_b	Port B Address width. Equal to $\log_2(\text{pmi_addr_depth_b})$.	1-<Max that can fit in the device>	9
pmi_data_width_b	Port B Data word width.	1-512	18
pmi_regmode_a	Data Out for port A (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_regmode_b	Data Out for port B (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_gsr	Sets if the global set/reset is enabled.	"enabled", "disable"	"disable"
pmi_resetmode	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	"async", "sync"	"sync"
pmi_optimization ¹	Describes the synthesis directive if optimizing for area or speed	"speed", "area"	"speed"
pmi_init_file	When Memory file is selected, user should set this parameter to the memory file.	<file_path>	"none"
pmi_init_file_format	This option allows users to select if the memory file is formatted as Binary, Hex. (Check Sec.5 for details on different formats)	"binary", "hex"	"binary"
pmi_write_mode_a	Defines the write mode of the module's port A.	"normal"	"normal"
pmi_write_mode_b	Defines the write mode of the module's port B.	"normal"	"normal"
pmi_family ²	Defines the FPGA family being used in the module	"LIFCL", "LFD2NX", "LFPCNX"	"common"
module_type ¹	Refers to the type of pmi module.	"pmi_ram_dp_true"	"pmi_ram_dp_true"

Notes:

1. Unused
2. Use pmi_family = "common" if initialization is needed. pmi_family = "common", requires 30-bits to be properly implemented ($\text{pmi_addr_depth_a} \times \text{pmi_data_width_a} \geq 30$), use pmi_family = "<device>" if mixed-width is needed. Mixed-width + initialization is NOT supported.

Verilog pmi_ram_dp_true Definition

```
module pmi_ram_dp_true
#(parameter pmi_addr_depth_a = 512,
  parameter pmi_addr_width_a = 9,
  parameter pmi_data_width_a = 18,
  parameter pmi_addr_depth_b = 512,
  parameter pmi_addr_width_b = 9,
  parameter pmi_data_width_b = 18,
  parameter pmi_regmode_a = "reg",
  parameter pmi_regmode_b = "reg",
  parameter pmi_gsr = "disable",
  parameter pmi_resetmode = "sync",
  parameter pmi_optimization = "speed",
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_write_mode_a = "normal",
  parameter pmi_write_mode_b = "normal",
  parameter pmi_family = "common",
  parameter module_type = "pmi_ram_dp_true")

  (input [(pmi_data_width_a-1):0] DataInA,
    input [(pmi_data_width_b-1):0] DataInB,
    input [(pmi_addr_width_a-1):0] AddressA,
    input [(pmi_addr_width_b-1):0] AddressB,
    input ClockA,
    input ClockB,
    input ClockEnA,
    input ClockEnB,
    input WrA,
    input WrB,
    input ResetA,
    input ResetB,
    output [(pmi_data_width_a-1):0] QA,
    output [(pmi_data_width_b-1):0] QB);

endmodule // pmi_ram_dp_true
```


VHDL pmi_ram_dp_true Definition

```
component pmi_ram_dp_true is
  generic (
    pmi_addr_depth_a : integer := 512;
    pmi_addr_width_a : integer := 9;
    pmi_data_width_a : integer := 18;
    pmi_addr_depth_b : integer := 512;
    pmi_addr_width_b : integer := 9;
    pmi_data_width_b : integer := 18;
    pmi_regmode_a : string := "reg";
    pmi_regmode_b : string := "reg";
    pmi_gsr : string := "disable";
    pmi_resetmode : string := "sync";
    pmi_optimization : string := "speed";
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_write_mode_a : string := "normal";
    pmi_write_mode_b : string := "normal";
    pmi_family : string := "common";
    module_type : string := "pmi_ram_dp_true"
  );
  port (
    DataInA : in std_logic_vector((pmi_data_width_a-1) downto 0);
    DataInB : in std_logic_vector((pmi_data_width_b-1) downto 0);
    AddressA : in std_logic_vector((pmi_addr_width_a-1) downto 0);
    AddressB : in std_logic_vector((pmi_addr_width_b-1) downto 0);
    ClockA: in std_logic;
    ClockB: in std_logic;
    ClockEnA: in std_logic;
    ClockEnB: in std_logic;
    WrA: in std_logic;
    WrB: in std_logic;
    ResetA: in std_logic;
    ResetB: in std_logic;
    QA : out std_logic_vector((pmi_data_width_a-1) downto 0);
    QB : out std_logic_vector((pmi_data_width_b-1) downto 0)
  );
end component pmi_ram_dp_true;
```

4.7. pmi_distributed_spram

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_distributed_spram (Single Port Memory Module – Distributed Memory). This IP is only available on LIFCL and LFD2NX devices.

Table 4.13. Single Port – Distributed Memory PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	Data	Bus	(pmi_data_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Table 4.14. Single Port – Distributed Memory PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_addr_depth	Address depth of the read and write port	2–<Max that can fit in the device>	32
pmi_addr_width	Address width of the read and write port	1–<Max that can fit in the device>	5
pmi_data_width	Data word width of the Read and Write port	1–512	8
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	“reg”, “noreg”	“reg”
pmi_init_file	When Memory file is selected, user should set this parameter to the memory file.	<file_path>	“none”
pmi_init_file_format	This option allows users to select if the memory file is formatted as Binary, Hex. (Check Sec.5 for details on different formats)	“binary”, “hex”	“binary”
pmi_family	This option selects the product family used. Defaults to common.	“LIFCL”, “LFD2NX”, “LFCLPNX”	“common”
module_type	Refers to the type of pmi module.	“pmi_distributed_spram”	“pmi_distributed_spram”

Verilog pmi_distributed_spram Definition

```
module pmi_distributed_spram
#(parameter pmi_addr_depth = 32,
  parameter pmi_addr_width = 5,
  parameter pmi_data_width = 8,
  parameter pmi_regmode = "reg",
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_family = "common",
  parameter module_type = "pmi_distributed_spram")

(
  input [(pmi_addr_width-1):0] Address,
  input [(pmi_data_width-1):0] Data,
  input Clock,
  input ClockEn,
  input WE,
  input Reset,
  output [(pmi_data_width-1):0] Q);

endmodule // pmi_distributed_spram
```

VHDL pmi_distributed_spram Definition

```
component pmi_distributed_spram is
  generic (
    pmi_addr_depth : integer := 32;
    pmi_addr_width : integer := 5;
    pmi_data_width : integer := 8;
    pmi_regmode : string := "reg";
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_family : string := "common";
    module_type : string := "pmi_distributed_spram"
  );
  port (
    Address : in std_logic_vector((pmi_addr_width-1) downto 0);
    Data : in std_logic_vector((pmi_data_width-1) downto 0);
    Clock: in std_logic;
    ClockEn: in std_logic;
    WE: in std_logic;
    Reset: in std_logic;
    Q : out std_logic_vector((pmi_data_width-1) downto 0)
  );
end component pmi_distributed_spram;
```

4.8. pmi_distributed_dpram

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_distributed_dpram (Pseudo Dual Port Memory Module). This IP is only available on LIFCL and LFD2NX devices.

Table 4.15. Distributed Pseudo Dual Port PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_addr_width - 1):0
I	RdAddress	Bus	(pmi_addr_width - 1):0
I	Data	Bus	(pmi_data_width - 1):0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Table 4.16. Distributed Pseudo Dual Port PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_addr_depth	Write Port Address depth.	2-<Max that can fit in the device>	32
pmi_addr_width	Write Port Address width. Equal to $\log_2(\text{pmi_addr_depth})$	1-<Max that can fit in the device>	5
pmi_data_width	Write Port Data word width.	1-512	8
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_init_file	When Memory file is selected, user should set this parameter to the memory file.	<file_path>	"none"
pmi_init_file_format	This option allows users to select if the memory file is formatted as Binary, Hex. (Check Sec.5 for details on different formats)	"binary", "hex"	"binary"
pmi_family	Defines the FPGA family being used in the module	"LIFCL", "LFD2NX", "LFCPNX"	"common"
module_type	Refers to the type of pmi module.	"pmi_ram_dp"	"pmi_ram_dp"

Verilog pmi_distributed_dpram Definition

```
module pmi_distributed_dpram # (  
    parameter pmi_addr_depth = 32,  
    parameter pmi_addr_width = 5,  
    parameter pmi_data_width = 8,  
    parameter pmi_regmode = "reg",  
    parameter pmi_init_file = "none",  
    parameter pmi_init_file_format = "binary",  
    parameter pmi_family = "common",  
    parameter module_type = "pmi_distributed_dpram"  
) (  
    input [(pmi_addr_width-1):0] WrAddress,  
    input [(pmi_data_width-1):0] Data,  
    input WrClock,  
    input WE,  
    input WrClockEn,  
    input [(pmi_addr_width-1):0] RdAddress,  
    input RdClock,  
    input RdClockEn,  
    input Reset,  
    output [(pmi_data_width-1):0] Q  
);
```

VHDL pmi_distributed_dpram Definition

```
component pmi_distributed_dpram is  
    generic (  
        pmi_addr_depth : integer := 32;  
        pmi_addr_width : integer := 5;  
        pmi_data_width : integer := 8;  
        pmi_regmode : string := "reg";  
        pmi_init_file : string := "none";  
        pmi_init_file_format : string := "binary";  
        pmi_family : string := "common";  
        module_type : string := "pmi_distributed_dpram"  
    );  
    port (  
        WrAddress : in std_logic_vector((pmi_addr_width-1) downto 0);  
        Data : in std_logic_vector((pmi_data_width-1) downto 0);  
        WrClock: in std_logic;  
        WE: in std_logic;  
        WrClockEn: in std_logic;  
        RdAddress : in std_logic_vector((pmi_addr_width-1) downto 0);  
        RdClock: in std_logic;  
        RdClockEn: in std_logic;  
        Reset: in std_logic;  
        Q : out std_logic_vector((pmi_data_width-1) downto 0)  
    );  
end component pmi_distributed_dpram;
```

4.9. pmi_distributed_rom

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_rom (Read Only Memory Module). This IP is only available on LIFCL and LFD2NX devices.

Table 4.17. Read Only Memory PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	OutClock	Bus	N/A
I	OutClockEn	Bus	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Table 4.18. Read Only PMI Memory Attribute Definitions

Attributes	Description	Values	Default Value
pmi_addr_depth	Address depth.	2-<Max that can fit in the device>	32
pmi_addr_width	Address width. Equal to $\log_2(\text{pmi_addr_depth})$	1-<Max that can fit in the device>	5
pmi_data_width	Data word width.	1-512	8
pmi_regmode	Data Out port (Q) can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_init_file	When Memory file is selected, user should set this parameter to the memory file.	<file_path>	"none"
pmi_init_file_format	This option allows users to select if the memory file is formatted as Binary, Hex. (Check Sec.5 for details on different formats)	"binary", "hex"	"binary"
pmi_family	Defines the FPGA family being used in the module	"LIFCL", "LFD2NX", "LFCLPNX"	"common"
module_type	Refers to the type of pmi module.	"pmi_distributed_rom"	"pmi_distributed_rom"

Verilog pmi_rom Definition

```
module pmi_distributed_rom
#(parameter pmi_addr_depth = 32,
  parameter pmi_addr_width = 5,
  parameter pmi_data_width = 8,
  parameter pmi_regmode = "reg",
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_family = "common",
  parameter module_type = "pmi_distributed_rom")
(
  input [(pmi_addr_width-1):0] Address,
  input OutClock,
  input OutClockEn,
  input Reset,
  output [(pmi_data_width-1):0] Q);

endmodule
```

VHDL pmi_distributed_dpram Definition

```
component pmi_distributed_rom is
  generic (
    pmi_addr_depth : integer := 32;
    pmi_addr_width : integer := 5;
    pmi_data_width : integer := 8;
    pmi_regmode : string := "reg";
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_family : string := "common";
    module_type : string := "pmi_distributed_rom"
  );
  port (
    Address : in std_logic_vector((pmi_addr_width-1) downto 0);
    OutClock: in std_logic;
    OutClockEn: in std_logic;
    Reset: in std_logic;
    Q : out std_logic_vector((pmi_data_width-1) downto 0)
  );
end component pmi_distributed_rom;;
```

4.10. pmi_fifo

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_fifo (First In First Out Memory Module).

Table 4.19. FIFO Single Clock PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width - 1):0
I	Clock	Bit	N/A
I	WrEn	Bit	N/A
I	RdEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0
O	Empty	Bit	N/A
O	Full	Bit	N/A
O	AlmostEmpty	Bit	N/A
O	AlmostFull	Bit	N/A

Table 4.20. FIFO Single Clock PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_data_width	FIFO data width.	2—<Max that can fit in the device>	8
pmi_data_depth	FIFO address depth.	1—<Max that can fit in the device>	256
pmi_full_flag ¹	Defines the full flag.	2—<Max that can fit in the device>	256
pmi_empty_flag ¹	Defines the empty flag.	1—<Max that can fit in the device>	0
pmi_almost_full_flag	Defines the lower limit of the almost full flag	2—<Max that can fit in the device>	252
pmi_almost_empty_flag	Defines the upper limit of the almost empty flag.	1—<Max that can fit in the device>	4
pmi_regmode	Data Out for FIFO that can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_family	Defines the FPGA family being used in the module	"iCE40UP", "LIFCL", "LFD2NX", "LFCPNX"	"common"
pmi_implementation ²	Refers to the memory used by the FIFO module	"EBR", "LUT", "Hard_IP"	"EBR"
module_type ¹	Refers to the type of pmi module.	"pmi_fifo"	"pmi_fifo"

Notes:

1. Unused. *pmi_full_flag* takes the value of *pmi_data_depth*, *pmi_empty_flag* is tied to 0.
2. *pmi_implementation* uses a soft controller unless set to "Hard_IP". "Hard_IP" and "LUT" are only available on "LFD2NX" and "LIFCL" devices.

Verilog pmi_fifo Definition

```
module pmi_fifo #(
    parameter pmi_data_width = 8,
    parameter pmi_data_depth = 256,
    parameter pmi_full_flag = 256,
    parameter pmi_empty_flag = 0,
    parameter pmi_almost_full_flag = 252,
    parameter pmi_almost_empty_flag = 4,
    parameter pmi_regmode = "reg",
    parameter pmi_family = "common" ,
    parameter module_type = "pmi_fifo",
    parameter pmi_implementation = "EBR"
) (
    input  [pmi_data_width-1:0] Data,
    input  Clock,
    input  WrEn,
    input  RdEn,
    input  Reset,
    output [pmi_data_width-1:0] Q,
    output Empty,
    output Full,
    output AlmostEmpty,
    output AlmostFull);

endmodule // pmi_fifo
```

VHDL pmi_ram_dp Definition

```
component pmi_fifo is
  generic (
    pmi_data_width : integer := 8;
    pmi_data_depth : integer := 256;
    pmi_full_flag : integer := 256;
    pmi_empty_flag : integer := 0;
    pmi_almost_full_flag : integer := 252;
    pmi_almost_empty_flag : integer := 4;
    pmi_regmode : string := "reg";
    pmi_family : string := "common" ;
    module_type : string := "pmi_fifo";
    pmi_implementation : string := "EBR"
  );
  port (
    Data : in std_logic_vector(pmi_data_width-1 downto 0);
    Clock: in std_logic;
    WrEn: in std_logic;
    RdEn: in std_logic;
    Reset: in std_logic;
    Q : out std_logic_vector(pmi_data_width-1 downto 0);
    Empty: out std_logic;
    Full: out std_logic;
    AlmostEmpty: out std_logic;
    AlmostFull: out std_logic
  );
end component pmi_fifo;
```

4.11. pmi_fifo_dc

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp (Pseudo Dual Port Memory Module).

Table 4.21. FIFO Dual Clock PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width_w - 1):0
I	WrClock	Bit	N/A
I	RdClock	Bit	N/A
I	WrEn	Bit	N/A
I	RdEn	Bit	N/A
I	Reset	Bit	N/A
I	RPRreset	Bit	N/A
O	Q	Bus	(pmi_data_width_r - 1):0
O	Empty	Bit	N/A
O	Full	Bit	N/A
O	AlmostEmpty	Bit	N/A
O	AlmostFull	Bit	N/A

Table 4.22. FIFO Dual Clock Port PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_data_width_w	FIFO data write port size.	2—<Max that can fit in the device>	18
pmi_addr_depth_w	FIFO address depth write port size.	1—<Max that can fit in the device>	256
pmi_data_width_r	FIFO data read port size.	2—<Max that can fit in the device>	18
pmi_addr_depth_r	FIFO address depth read port size.	1—<Max that can fit in the device>	256
pmi_full_flag ¹	Defines the full flag.	2—<Max that can fit in the device>	256
pmi_empty_flag ¹	Defines the empty flag.	1—<Max that can fit in the device>	0
pmi_almost_full_flag	Defines the lower limit of the almost full flag	2—<Max that can fit in the device>	252
pmi_almost_empty_flag	Defines the upper limit of the almost empty flag.	1—<Max that can fit in the device>	4
pmi_regmode	Data Out for FIFO, if can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_resetmode	Selection for the Reset to be Synchronous or Asynchronous to the Clock.	"async", "sync"	"async"
pmi_family	Defines the FPGA family being used in the module	"iCE40UP", "LIFCL", "LFD2NX", "LFCPNX"	"common"
pmi_implementation ²	Refers to the memory used by the FIFO module	"EBR", "LUT", "Hard_IP"	"EBR"
module_type	Refers to the type of pmi module.	"pmi_fifo_dc"	"pmi_fifo_dc"

Notes:

1. Unused. *pmi_full_flag* takes the value of *pmi_data_depth*, *pmi_empty_flag* is tied to 0.
2. *pmi_implementation* uses a soft controller unless set to "Hard_IP". "Hard_IP" and "LUT" are only available on "LFD2NX" and "LIFCL" devices.

Verilog pmi_fifo_dc Definition

```
module pmi_fifo_dc #(
    parameter pmi_data_width_w = 18,
    parameter pmi_data_width_r = 18,
    parameter pmi_data_depth_w = 256,
    parameter pmi_data_depth_r = 256,
    parameter pmi_full_flag = 256,
    parameter pmi_empty_flag = 0,
    parameter pmi_almost_full_flag = 252,
    parameter pmi_almost_empty_flag = 4,
    parameter pmi_regmode = "reg",
    parameter pmi_resetmode = "async",
    parameter pmi_family = "common" ,
    parameter module_type = "pmi_fifo_dc",
    parameter pmi_implementation = "EBR"
) (
    input  [pmi_data_width_w-1:0] Data,
    input  WrClock,
    input  RdClock,
    input  WrEn,
    input  RdEn,
    input  Reset,
    input  RPRreset,
    output [pmi_data_width_r-1:0] Q,
    output Empty,
    output Full,
    output AlmostEmpty,
    output AlmostFull);
endmodule // pmi_fifo_dc
```

VHDL pmi_fifo_dc Definition

```
component pmi_fifo_dc is
  generic (
    pmi_data_width_w : integer := 18;
    pmi_data_width_r : integer := 18;
    pmi_data_depth_w : integer := 256;
    pmi_data_depth_r : integer := 256;
    pmi_full_flag : integer := 256;
    pmi_empty_flag : integer := 0;
    pmi_almost_full_flag : integer := 252;
    pmi_almost_empty_flag : integer := 4;
    pmi_regmode : string := "reg";
    pmi_resetmode : string := "async";
    pmi_family : string := "common" ;
    module_type : string := "pmi_fifo_dc";
    pmi_implementation : string := "EBR"

  );
  port (
    Data : in std_logic_vector(pmi_data_width_w-1 downto 0);
    WrClock: in std_logic;
    RdClock: in std_logic;
    WrEn: in std_logic;
    RdEn: in std_logic;
    Reset: in std_logic;
    RPRreset: in std_logic;
    Q : out std_logic_vector(pmi_data_width_r-1 downto 0);
    Empty: out std_logic;
    Full: out std_logic;
    AlmostEmpty: out std_logic;
    AlmostFull: out std_logic
  );
end component pmi_fifo_dc;
```

4.12. pmi_distributed_shift_reg

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_distributed_shift_reg (Shift Register Memory Module). This IP is only available on LIFCL and LFD2NX devices.

Table 4.23. Shift Register PMI Port Definitions

Direction	Port Name	Type	Size (Buses Only)
I	Addr	Bus	(pmi_max_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	Q	Bus	(pmi_data_width - 1):0

Table 4.24. Shift Register PMI Attribute Definitions

Attributes	Description	Values	Default Value
pmi_data_width	Data width for shift register.	2-<Max that can fit in the device>	8
pmi_max_shift	Maximum shift register size.	2-<Max that can fit in the device>	16
pmi_max_width	Maximum shift register width size. Equal to $\log_2(\text{pmi_max_shift} + 1)$.	1-512	5
pmi_num_shift	Maximum amount, which can be shifted.	2-<Max that can fit in the device>	16
pmi_num_width	Maximum shift amount width size. Equal to $\log_2(\text{pmi_num_shift} + 1)$.	1-512	5
pmi_regmode	Data Out can be registered or not using this selection.	"reg", "noreg"	"reg"
pmi_shiftreg_type	Type of shift register.	"fixed", "variable"	"fixed"
pmi_family	Defines the FPGA family being used in the module	"LIFCL", "LFD2NX", "LFCPNX"	"common"
module_type	Refers to the type of pmi module.	"pmi_distributed_shift_reg"	"pmi_distributed_shift_reg"

Verilog pmi_distributed_shift_reg reg Definition

```
module pmi_distributed_shift_reg
#(parameter pmi_data_width = 8,
  parameter pmi_regmode = "reg",
  parameter pmi_shiftreg_type = "fixed",
  parameter pmi_num_shift = 16,
  parameter pmi_num_width = 4,
  parameter pmi_max_shift = 16,
  parameter pmi_max_width = 4,
  parameter pmi_init_file = "none",
  parameter pmi_init_file_format = "binary",
  parameter pmi_family = "common",
  parameter module_type = "pmi_distributed_shift_reg")

(
  input [(pmi_data_width-1):0] Din,
  input [(pmi_max_width-1):0] Addr,
  input Clock,
  input ClockEn,
  input Reset,
  output [(pmi_data_width-1):0] Q);

endmodule // pmi_distributed_shift_reg
```

VHDL pmi_distributed_shift_reg Definition

```
component pmi_distributed_shift_reg is
  generic (
    pmi_data_width : integer := 8;
    pmi_regmode : string := "reg";
    pmi_shiftreg_type : string := "fixed";
    pmi_num_shift : integer := 16;
    pmi_num_width : integer := 4;
    pmi_max_shift : integer := 16;
    pmi_max_width : integer := 4;
    pmi_init_file : string := "none";
    pmi_init_file_format : string := "binary";
    pmi_family : string := "common";
    module_type : string := "pmi_distributed_shift_reg"
  );
  port (
    Din : in std_logic_vector((pmi_data_width-1) downto 0);
    Addr : in std_logic_vector((pmi_max_width-1) downto 0);
    Clock: in std_logic;
    ClockEn: in std_logic;
    Reset: in std_logic;
    Q : out std_logic_vector((pmi_data_width-1) downto 0)
  );
end component pmi_distributed_shift_reg;
```

5. 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.

5.1. Initialization File Formats

The initialization file is an ASCII file, which you can create or edit using any ASCII editor. Module/IP Block Wizard supports the binary and hex memory file formats.

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 involves 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.

5.2. 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 20x32

```
00100000010000000010000001000000
00000001000000010000000100000001
00000010000000010000000100000010
000000110000000110000001100000011
000001000000001000000010000000100
00000101000001010000010100000101
000001100000001100000011000000110
00000111000001110000011100000111
00001000010010000000100001001000
00001001010010010000100101001001
00001010010010100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
00001110001111100000111000111110
00001111001111110000111100111111
00010000000100000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011
```


5.3. Hex File

The hex file is a text file of hexadecimal characters 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 8x16

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

6. Simulation of Memory Modules

Table 6.1. File List

File	Sim	Synthesis	Description
<IP_name>.v	Yes	Yes	Top Level RTL file with the selected configuration. The main IP file
dut_params	Yes	—	Top level parameters of the generated RTL file
dut_inst	Yes	—	Instantiated version of the <IP_name>.v file for simulation use
tb_top.v	Yes	—	Test bench template. You can edit this to match your specific needs.
<IP_name>.cfg	—	—	This file contains the configuration options used to recreate or modify the core in the IP Platform.
<IP_name>.ipx	—	—	The IPX file holds references to all of the elements of an IP or Module after it is generated from the IP Platform user interface. The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP Platform when the IP/Module is being regenerated.

Lattice Radiant Software also allows you to simulate their configured memory modules, so the designer can observe the conditions of the output with a given stimuli. The following section details the individual steps needed to execute the testbench in Active-HDL.

To execute the testbench in Active-HDL

1. Create a new IP instance. Refer to the IP Generation section for detailed instructions. As an example, generate a Pseudo Dual-Port Memory on default settings, with an IP name of dp_512x18. See the [Pseudo Dual-Port RAM \(RAM_DP\) – EBR Based](#) section for more information regarding the IP.

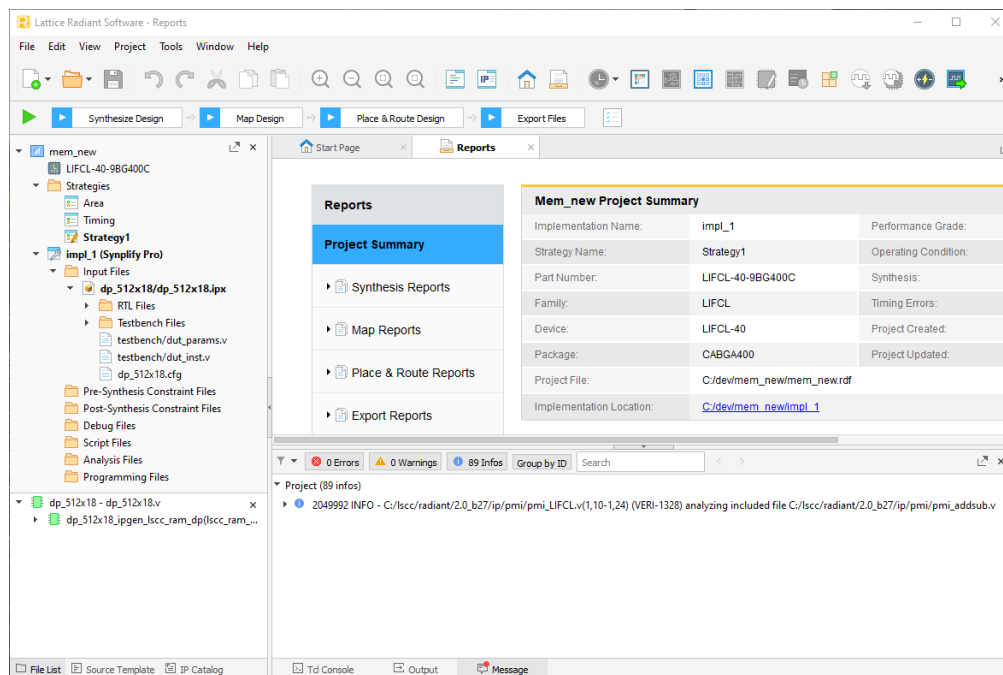


Figure 6.1. Project with Instance of Pseudo Dual Port Memory

- Right-click on **impl_1** under the Strategies folder and click **Add Existing File**. The file selection window opens, showing the folder where the project is saved.

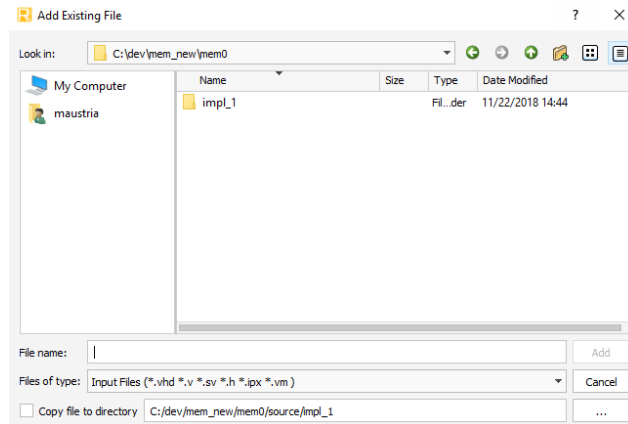


Figure 6.2. File Selection Window

- Click the name of the IP instance **dp_512x18**.
- Navigate to **<IP name>/testbench**. Select the **tb_top.v** file and click **Add**. This adds the top-level testbench to the project. After opening, the interface is shown in Figure 6.3.

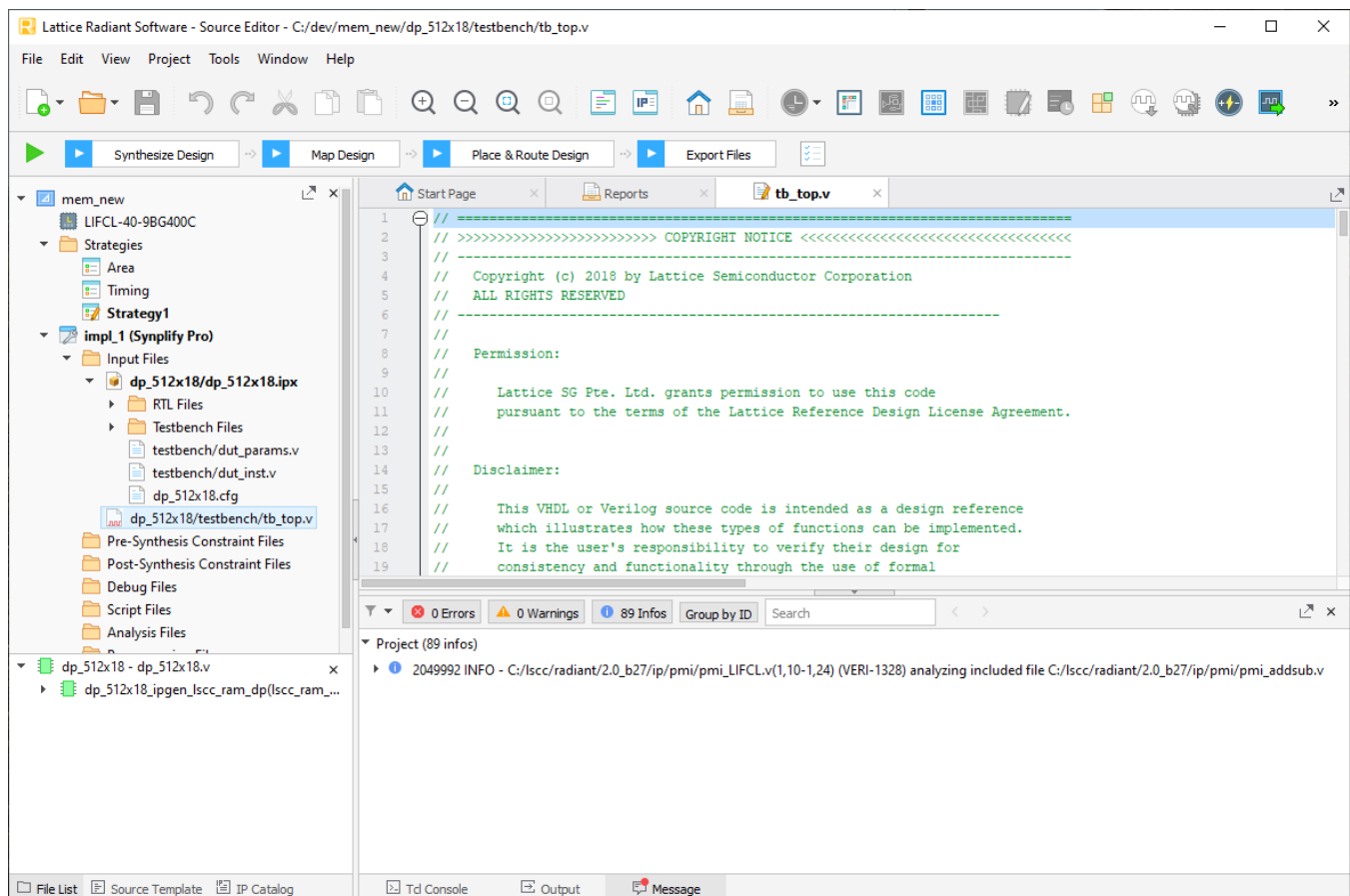


Figure 6.3. Top Level Testbench Instance in Project.

5. Modify the contents of the testbench to add a specific stimuli or check a specific option. For this example, leave it on the default RTL file.
6. Simulate the project. Click on **Tools** and open **Simulation Wizard**. The **Simulation Wizard** window appears.
7. Click **Next**. You are prompted to select the working folder and the test bench name. Enter **test**.

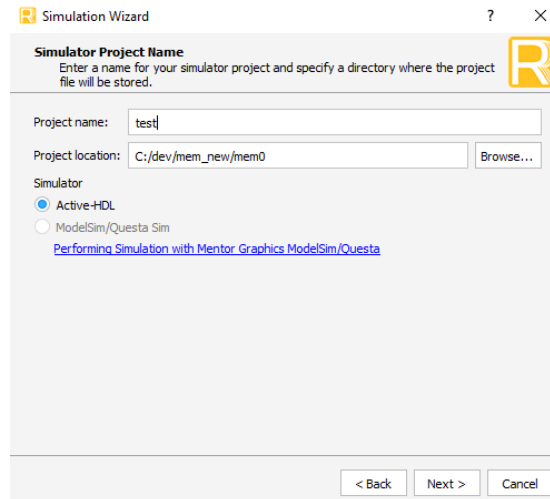


Figure 6.4. Simulation Wizard

8. Click **Next**. In the **Folder Creation** prompt, select **Yes**.
9. In the processing stage, select **RTL**.
10. Click **Next**. The source file list appears. This provides an overview of the list of files to be included for simulation.

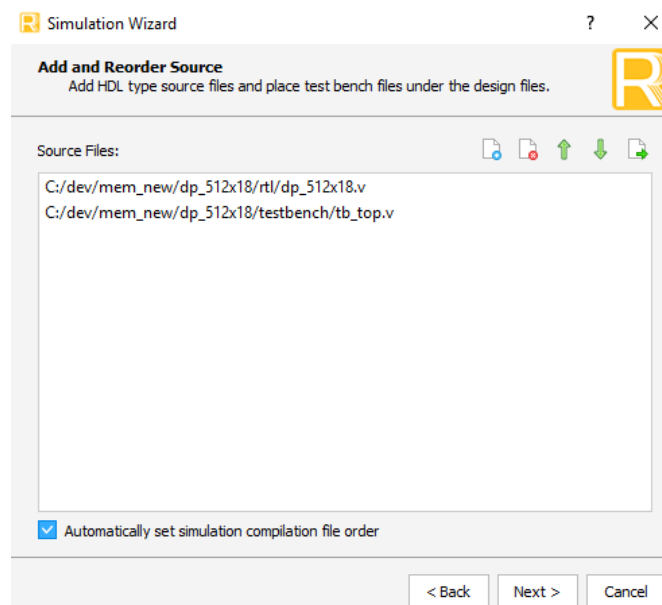


Figure 6.5. Final Simulation Files

11. Add or subtract any other files that may have been missed. Leave the file order for now. Click **Next**.

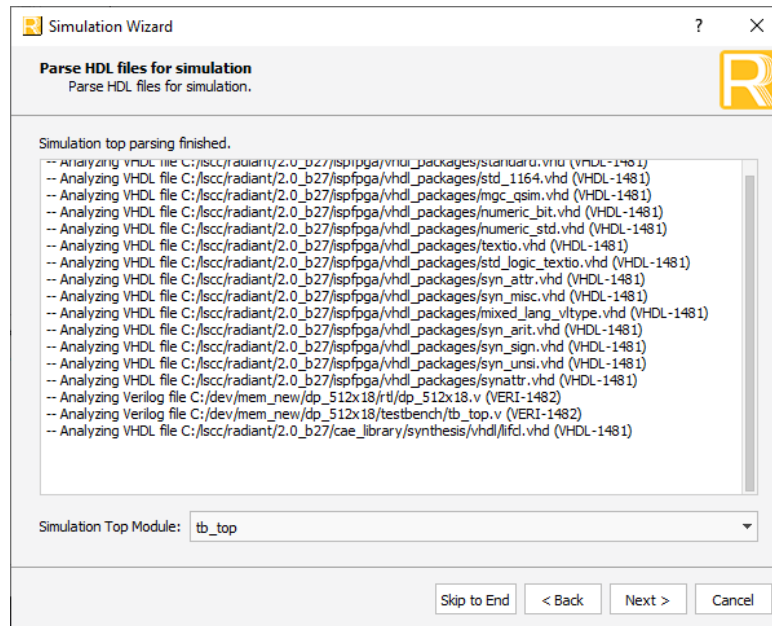


Figure 6.6. File Parsing and Top Module Selection

12. In the next section, the files as well as the other dependencies are parsed to check for final errors. You can select the top simulation module before passing the files to Active-HDL for execution. Click **Next** for a summary view and last minute options.
13. Click **Finish**. This then opens the Active-HDL Lattice Edition software, which automatically compiles and runs the RTL simulation.

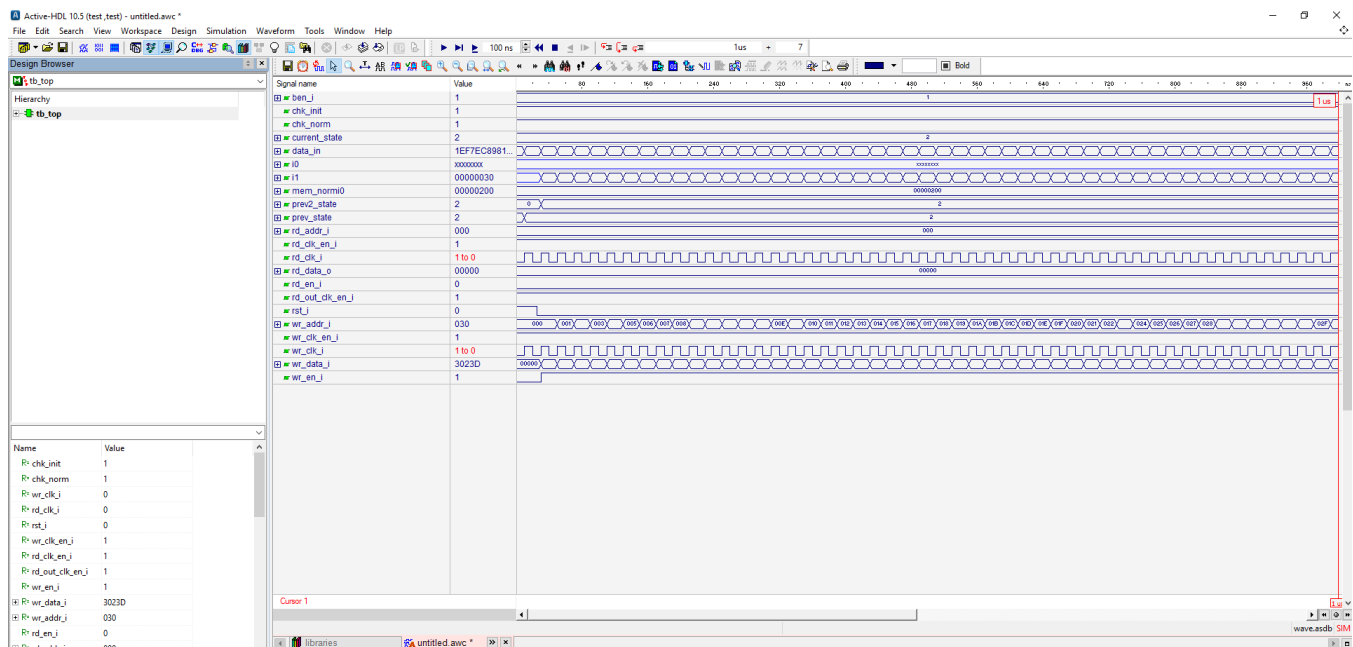


Figure 6.7. Active-HDL Initial Simulation

14. When Active-HDL opens, it compiles the RTL files and runs the simulation up to the first 100 ns.
15. To view the rest of simulation, click the **Play** button on top and zoom out for the overview of the signals.

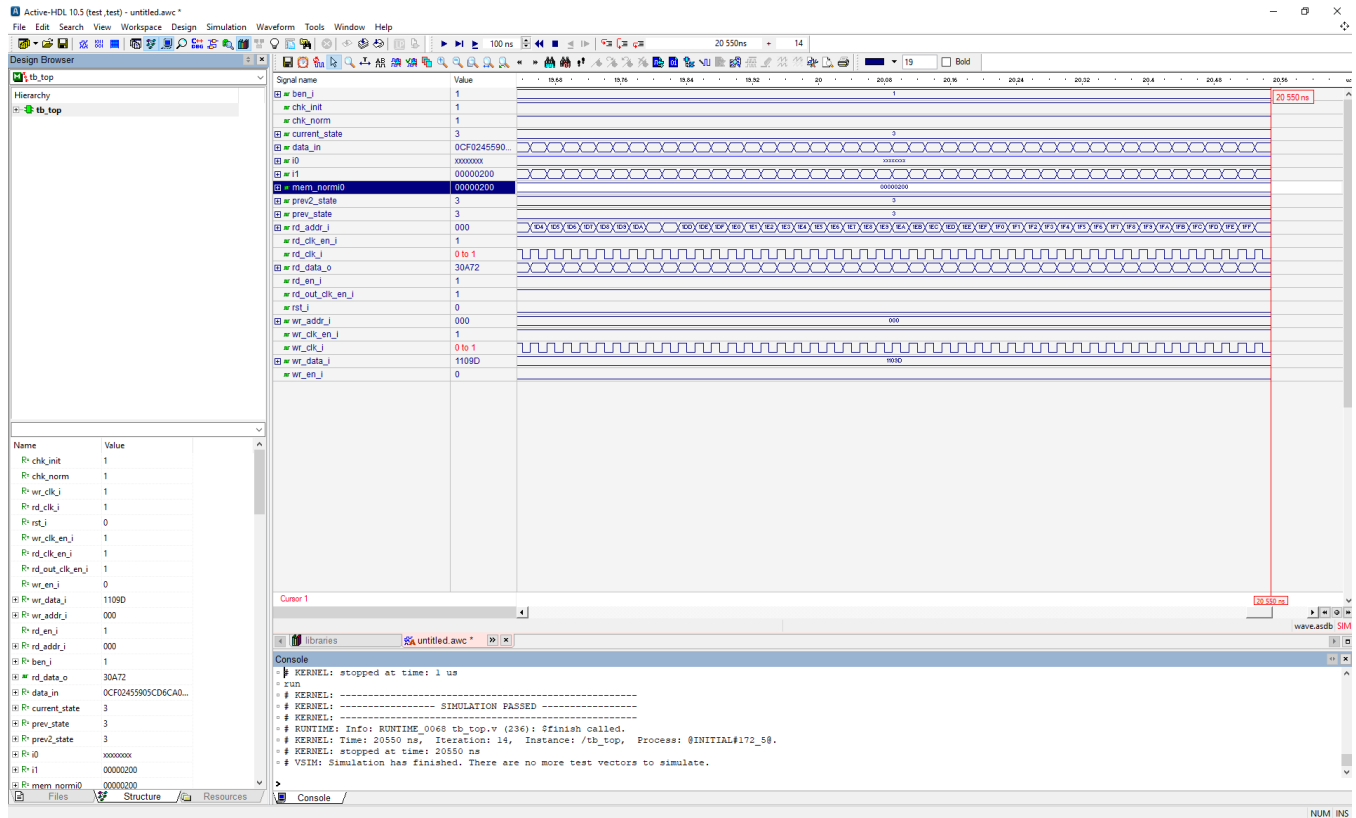


Figure 6.8. Active-HDL Final Simulation

16. At this point, simulation is completed. You can check the individual sections of the signal by zooming in and out of the window. A simulation passed message should also appear.

Appendix A. Resource Utilization

Table A.1. Device and Tool Tested

	Value
Lattice Radiant Software Version	ng2.0 (for Windows)
Device Used	LIFCL-40-9BG400I
Performance Grade	9_High-Performance_1.0V
Synthesis Tool	Synplify Pro (R) O-2018.09LR-SP1-Beta, Build 219R, May 6 2019
	Lattice Synthesis Engine (LSE)

Note: Some bits are clipped to accommodate the current configuration with the selected device.

Table A.2. EBR-Based Single-Port Memory Utilization

Memory Size	REG_EN	Byte-EN	Synthesis Tool	Register	LUTs	EBR
512 × 36	"reg"	No	Synplify Pro	0	3	2
			LSE	0	3	2
16384 × 32	"reg"	Yes	Synplify Pro	0	6	32
			LSE	0	6	32
4096 × 128	"noreg"	No	Synplify Pro	1	68	16
			LSE	1	68	16

Table A.3. EBR-Based Pseudo Dual-Port Memory Utilization

Write CONFIG	Read CONFIG	REG_EN	Byte-EN	Synthesis Tool	Register	LUTs	EBR
512 × 36	512 × 36	"reg"	No	Synplify Pro	1	2	1
				LSE	1	1	1
16384 × 32	4096 × 128	"reg"	Yes	Synplify Pro	14	385	32
				LSE	16	375	32
2048 × 256	16384 × 32	"noreg"	No	Synplify Pro	0	1	32
				LSE	0	1	32

Table A.4. EBR-Based True Dual-Port Memory Utilization

Port A CONFIG	Read CONFIG	REG A	REG B	Byte-EN A	Byte-EN B	Synthesis Tool	Register	LUTs	EBR
512 × 18	512 × 18	"reg"	"reg"	No	No	Synplify Pro	1	2	1
						LSE	1	1	1
16384 × 16	8192 × 32	"reg"	"reg"	Yes	Yes	Synplify Pro	12	276	16
						LSE	15	268	16
8192 × 32	32768 × 8	"noreg"	"noreg"	No	No	Synplify Pro	2	45	16
						LSE	2	45	16

Table A.5. EBR-Based Read-Only Memory Utilization

Memory Size	REG_EN	Synthesis Tool	Register	LUTs	EBR
1024 × 32	"reg"	Synplify Pro	N/A	N/A	N/A
		LSE	32	2	2

Table A.6. Distributed Single-Port Memory Utilization

Memory Size	REG_EN	Synthesis Tool	Register	LUTs	EBR
256 × 16	"reg"	Synplify Pro	32	529	0
		LSE	32	441	0

Table A.7. Distributed Pseudo Dual-Port Memory Utilization

Memory Size	REG_EN	Synthesis Tool	Register	LUTs	EBR
256 × 16	"reg"	Synplify Pro	32	529	0
		LSE	32	440	0

Table A.8. Distributed Read Only Memory Utilization

Memory Size	REG_EN	Synthesis Tool	Register	LUTs	EBR
256 × 32	"reg"	Synplify Pro	64	671	0
		LSE	64	666	0

Table A.9. Single Clock First In First Out Utilization

Memory Size	REG_EN	SHIFT TYPE	Memory Type	Synthesis Tool	Register	LUTs	EBR
1024 × 18	"reg"	Yes	EBR	Synplify Pro	225	191	1
				LSE	178	151	1
256 × 16	"noreg"	No	LUT	Synplify Pro	108	621	0
				LSE	108	519	0

Table A.10. Dual Clock First In First Out Utilization

Memory Size	Read CONFIG	REG_EN	Almost FLAG	Memory Type	Synthesis Tool	Register	LUTs	EBR
512 × 18	512 × 18	"reg"	Yes	EBR	Synplify Pro	84	102	1
					LSE	84	194	1
16384 × 16	8192 × 32	"reg"	Yes	EBR	Synplify Pro	117	291	16
					LSE	117	267	16
256 × 16	256 × 16	"noreg"	No	LUT	Synplify Pro	89	645	0
					LSE	88	571	0

Table A.11. Shift Register Utilization

Memory Size	REG_EN	SHIFT TYPE	Memory Type	Synthesis Tool	Register	LUTs	EBR
16 × 8	"reg"	fixed	EBR	Synplify Pro	27	19	1
				LSE	11	19	1
8192 × 24	"noreg"	variable	EBR	Synplify Pro	53	74	24
				LSE	28	61	24
256 × 16	"noreg"	fixed	LUT	Synplify Pro	34	1093	0
				LSE	34	907	0

Appendix B. Limitations

Following are the known limitations:

- PMI_* initialization can only be supported when using pmi_family = "common".
- PMI_RAM_DP and PMI_RAM_DP_TRUE mixed-width configurations can only be supported when using pmi_family = <device_family>.
- In previous versions of Lattice Radiant software, when using the Hard FIFO controller, you can encounter asynchronous flags, which can cause metastability in your system when not handled appropriately. This issue is rectified in this release of Lattice Radiant software and we encourage you to regenerate your IP to ensure design stability.

Technical Support Assistance

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

Revision History

Document Revision 2.5, Lattice Radiant SW Version 3.0, June 2021

Section	Change Summary
—	Replaced specific product names with <i>Lattice FPGA devices built on the Lattice Nexus platform</i> or <i>Lattice Nexus devices</i> .
Memory Modules	<ul style="list-style-type: none"> Changed Reset Assertion default value to <i>async</i> in Table 2.16. First in First Out Single Clock Attribute Definitions. Corrected headers in Table 2.18. First in First Out Dual Clock Attribute Definitions. Corrected <code>dps_i</code> port description in Table 2.21. Large-RAM based Single-Port Memory Port Definitions.

Document Revision 2.4, Lattice Radiant SW Version 2.3, December 2020

Section	Change Summary
Memory Modules	Updated Hard FIFO flag behavior.
PMI Support	Added "LFD2NX" and "LFCPNX" options.
Appendix B. Limitations	Added item.
—	Adjusted format of table footnotes.

Document Revision 2.3, Lattice Radiant SW Version 2.2, October 2020

Section	Change Summary
Memory Modules	Updated Hard FIFO controller information.

Document Revision 2.2, Lattice Radiant SW Version 2.1, June 2020

Section	Change Summary
Memory Modules	<ul style="list-style-type: none"> Added support for Certus-NX family. Added support for HARD FIFO controller. Added Large RAM modules for Crosslink-NX and Certus-NX.

Document Revision 2.1, Lattice Radiant SW Version 2.0, December 2019

Section	Change Summary
Appendix A. Resource Utilization	Removed maximum frequency information from utilization tables.
Appendix B. Limitations	Removed limitation.
All	Minor formatting changes

Document Revision 2.0, Lattice Radiant SW Version 2.0, October 2019

Section	Change Summary
—	Appended Lattice Radiant Software to document title.
Disclaimers	Added this section.
Memory Modules	Added support for CrossLink-NX family. Initial release for <code>ram_dp_true</code> , <code>distributed_dpam</code> , <code>distributed_spram</code> , <code>distributed_rom</code> , <code>pmi_ram_dp_true</code> , <code>pmi_distributed_dpam</code> , <code>pmi_distributed_spram</code> , <code>pmi_distributed_rom</code> , <code>pmi_distributed_shift_reg</code> (CrossLink-NX exclusive modules)
Appendix A. Resource Utilization	Added this section.
Appendix B. Limitations	Added this section.

Document Revision 1.1, Lattice Radiant SW Version 1.1, January 2019

Section	Change Summary
Memory Modules	<p>Added the following sections:</p> <ul style="list-style-type: none"> • Read Only Memory (ROM) – EBR Based • First In First Out Single Clock (FIFO) • First In First Out Dual Clock (FIFO_DC) • Shift Register (Shift_Register)
PMI Support	<p>Added the following sections:</p> <ul style="list-style-type: none"> • pmi_rom • pmi_ram_dp_be • pmi_ram_dq_be • pmi_fifo • pmi_fifo_dc
Simulation of Memory Modules	Added this section in the document.
Revision History	Updated revision history table to new template.

Document Revision 1.0, Lattice Radiant SW Version 1.0, January 2018

Section	Change Summary
All	Initial release



www.latticesemi.com