Distributed Memory Modules - Lattice Radiant Software

User Guide

FPGA-IPUG-02191-1.0

November 2022

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the 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

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
EBR	Embedded Block Random Access Memory
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 distributed memory usage for the LAV-AT Family of devices supported by Lattice Radiant™ Software. It is intended to be used by design engineers as a guide to integrating distributed memory blocks with their applications.

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

2. Memory Modules

The following sections discuss the different distributed 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 module 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 memory modules also support the Reset signal. The Reset (or RST) signal only resets the input and output registers of the memory module. It does not reset the contents of the memory module.
- Timing
  To correctly write into a memory cell in the memory module, the correct address should be registered by the logic. Hence, it is important to note that while running the trace on the memory module, 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 (Distributed_SPRAM) – LUT Based

Lattice FPGA devices built on the Lattice LAV-AT family support all the features of a 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 requirements.

Module/IP Block Wizard generates the memory module shown in Figure 2.1.

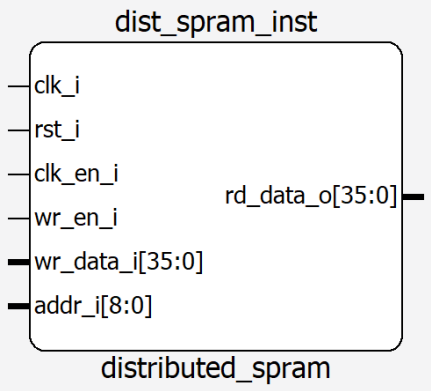

Figure 2.1. Distributed Single Port Memory Generated by Module/IP Block Wizard

The various ports and their definitions for Distributed 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. 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	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 2.2. Some of these attributes are user selectable through the Module/IP Block Wizard interface.

Table 2.2. LUT-Based Single Port Memory Attribute Definitions

Attribute	Description	Values	Default Value
Configuration Attributes			
Address Depth	Address the 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, Initialize to all 0s, Initialize to 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 Memory (Distributed_SPRAM) timing waveforms are shown in Figure 2.2 and Figure 2.3.

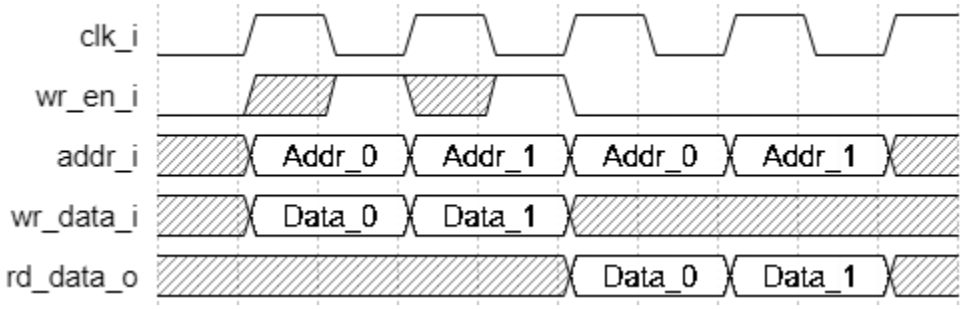

Figure 2.2. Distributed Single Port Memory Timing Waveform, Without Output Registers

Figure 2.3. Distributed Single Port Memory Timing Waveform, With Output Registers

2.2. Pseudo Dual Port RAM (Distributed_DPRAM) – LUT Based

Lattice FPGA devices built on the Lattice LAV-AT family support all the features of the 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 requirements.

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

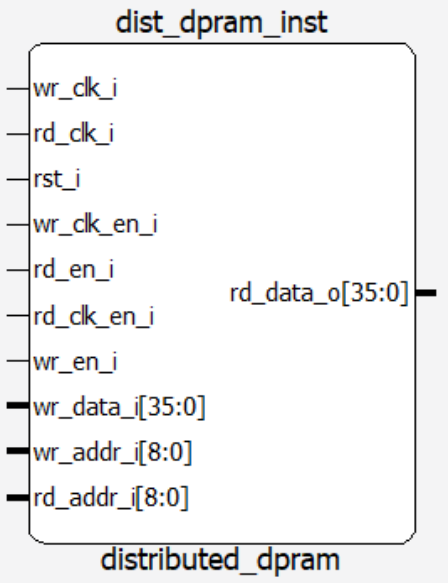

Figure 2.4. Distributed Pseudo Dual Port Memory Generated by Module/IP Block Wizard

The various ports and their definitions for Distributed 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. 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 2.4. Some of these attributes are user-selectable through the Module/IP on Local.

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

Attribute	Description	Values	Default Value
General Attributes			
Write Port Address Depth	Write the Port Address depth	2 - 32768	512
Write Port Data Width	Write Port Data word width	1 - 512	36
Read Port Address Depth	Read the 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 provide custom initialization through a memory file.	none, Initialize to all 0s, Initialize to all 1s, Memory File	none
Memory File	When the 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 Distributed Pseudo Dual Port Memory (Distributed_DPRAM). The internal timing waveforms for Distributed Pseudo Dual Port Memory (Distributed_DPRAM) with these options are shown in Figure 2.5 and Figure 2.6.

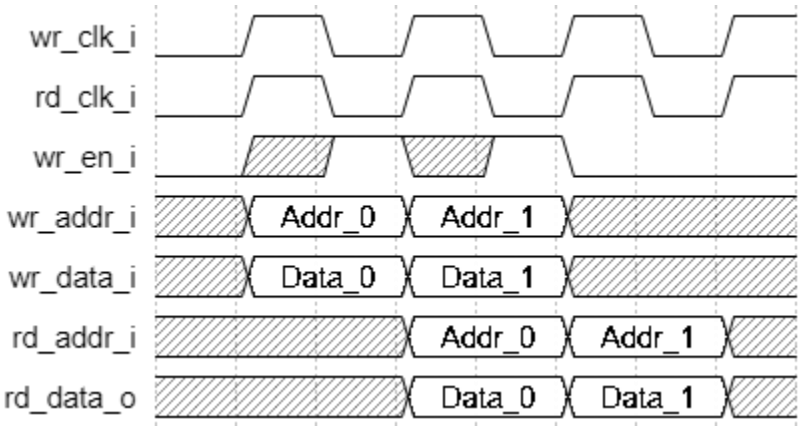

Figure 2.5. Distributed Pseudo Dual Port Memory Timing Diagram, Without Output Registers

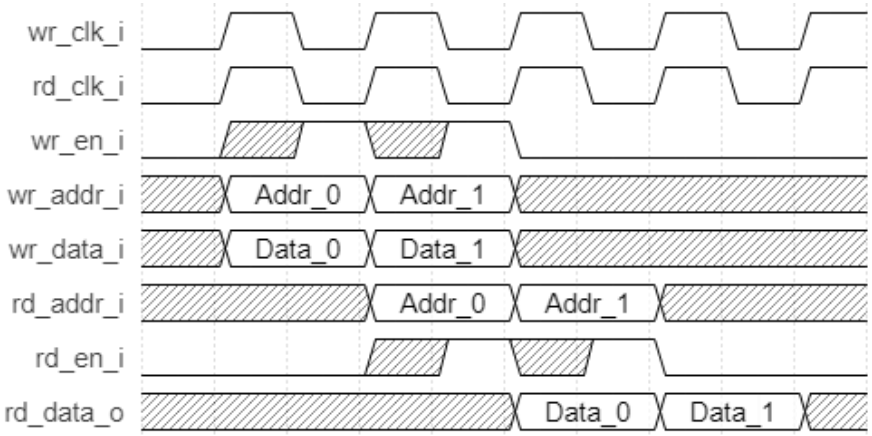

Figure 2.6. Distributed Pseudo Dual Port Memory Timing Diagram, With Output Registers

2.3. Read Only Memory (Distributed_ROM) – LUT Based

Lattice FPGA devices built on the Lattice LAV-AT family support all the features of the Read Only Memory Module or Distributed_ROM. Module/IP Block Wizard allows you to generate the Verilog-HDL for the memory size as per design requirements.

Module/IP Block Wizard generates the memory module shown in Figure 2.7.

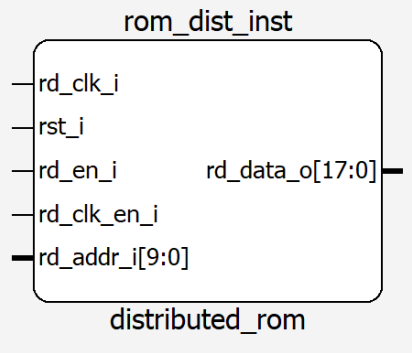

Figure 2.7. Distributed Read Only Memory Generated by Module/IP Block Wizard

The various ports and their definitions for Distributed Read Only 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. 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 2.6. Some of these attributes are user-selectable through the Module/IP on Local.

Table 2.6. 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 the 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 Distributed Read Only Memory (Distributed_ROM) timing waveforms are shown in Figure 2.8 and Figure 2.9.

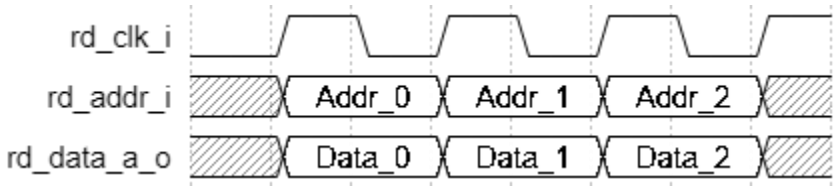

Figure 2.8. Distributed Read Only Memory Timing Diagram, Without Output Registers

Figure 2.9. Distributed Read Only Memory Timing Diagram, With Output Registers

3. IP Generation

This section provides information on how to generate the Distributed Memory Module IP using the Lattice Radiant software and how to run the simulation. For more details on the Lattice Radiant software, refer to the Lattice Radiant software user guide.

3.1. 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 Distributed_RAM as a 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.

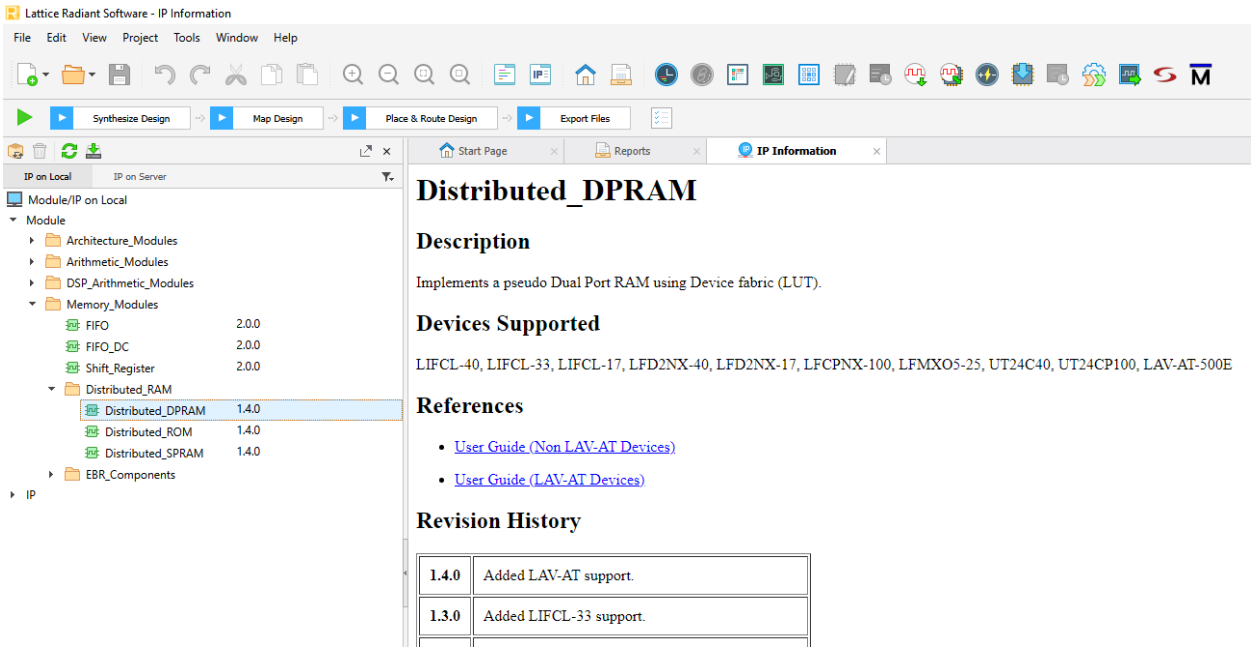

Figure 3.1. Memory Modules Under Module/IP on Local in the Lattice Radiant Software

The following section shows an example of generating a LUT-based Pseudo Dual Port RAM of size 512 x 18.

To generate a LUT-based Pseudo Dual Port RAM of size 512 x 18:

1. Double-click Distributed_DPRAM under Memory_Modules > Distributed_RAM.
2. The **Module/IP Block Wizard** opens as shown in Figure 3.2. Enter values in the **Component name** and the **Create in** fields then click **Next**.

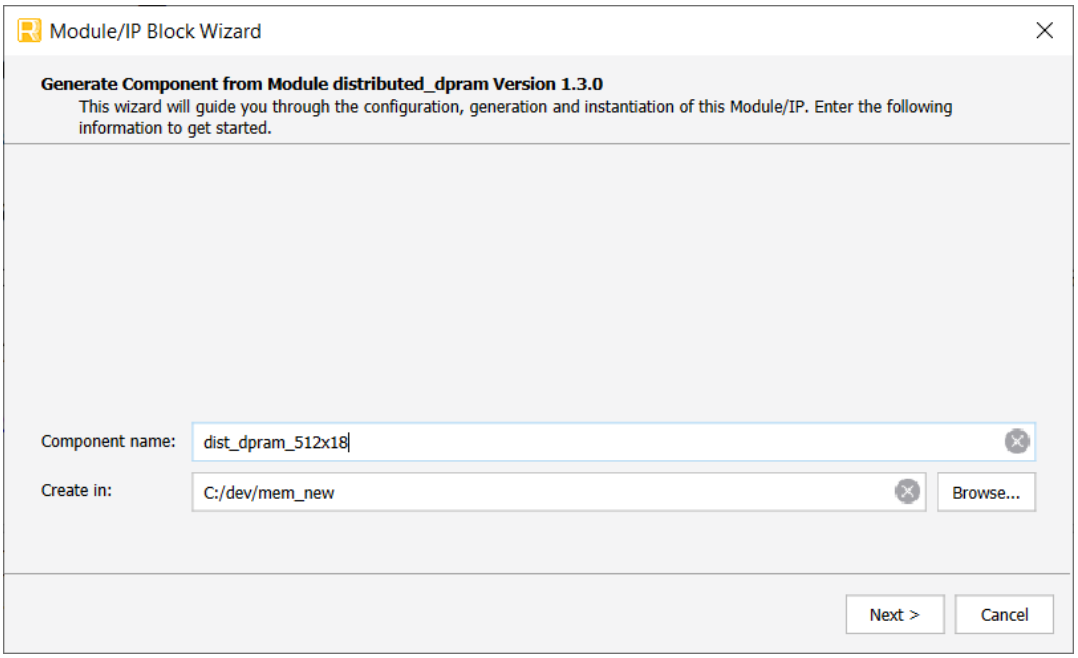

Figure 3.2. Example: Generating Distributed_DPRAM Using Module/IP Block Wizard

3. In the module's dialog box of the **Module/IP Block Wizard** window, customize the LUT-based Pseudo Dual Port RAM using drop-down menus and checkboxes as shown in Figure 3.3.

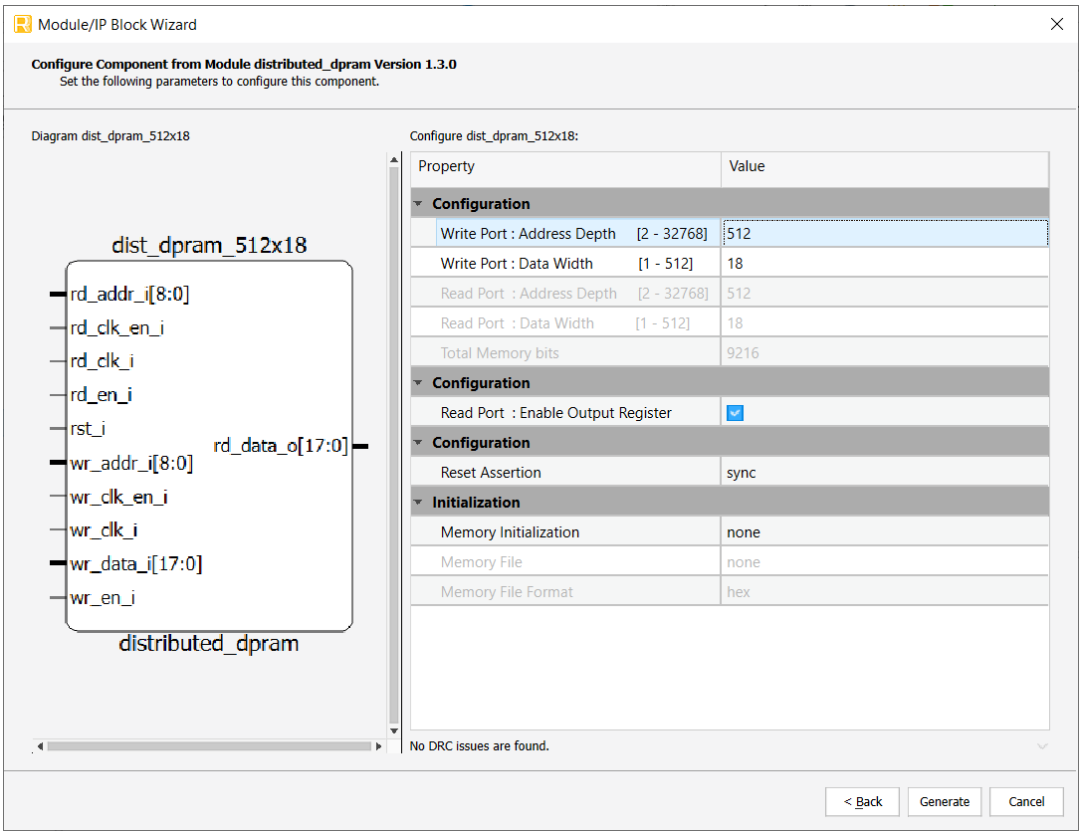

Figure 3.3. Example: Generating Distributed_DPRAM Module Customization

4. When all the options are set, click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results as shown in Figure 3.4.

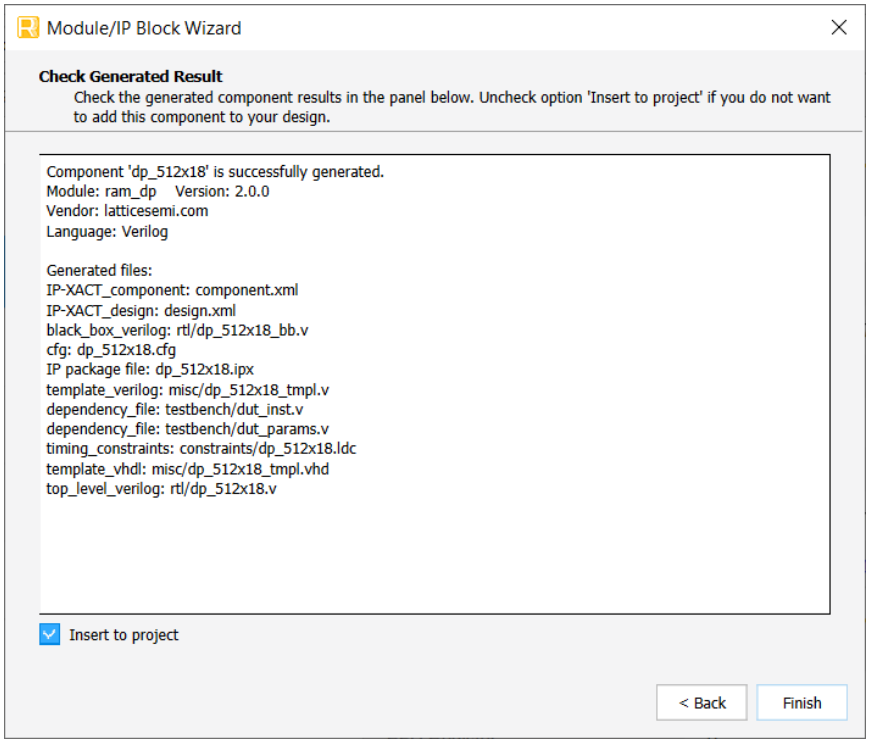

Figure 3.4. Example: Generating Distributed_DPRAM Check Generated Result

5. Click the **Finish** button. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields are shown in Figure 3.2.

The generated distributed Memory Module IP package includes the black box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.v) that can be used as an instantiation template for the IP core is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design. The generated files are listed in Table 3.1.

Table 3.1. Generated File List

File	Description
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file contains the attribute values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration attributes of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the IP core.
rtl/<Component name>_bb.v	This file provides the synthesis black box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the IP core.

3.2. Running Functional Simulation

1. Click the  button located on the **Toolbar** then click **Next** to initiate the **Simulation Wizard** shown in Figure 3.5.

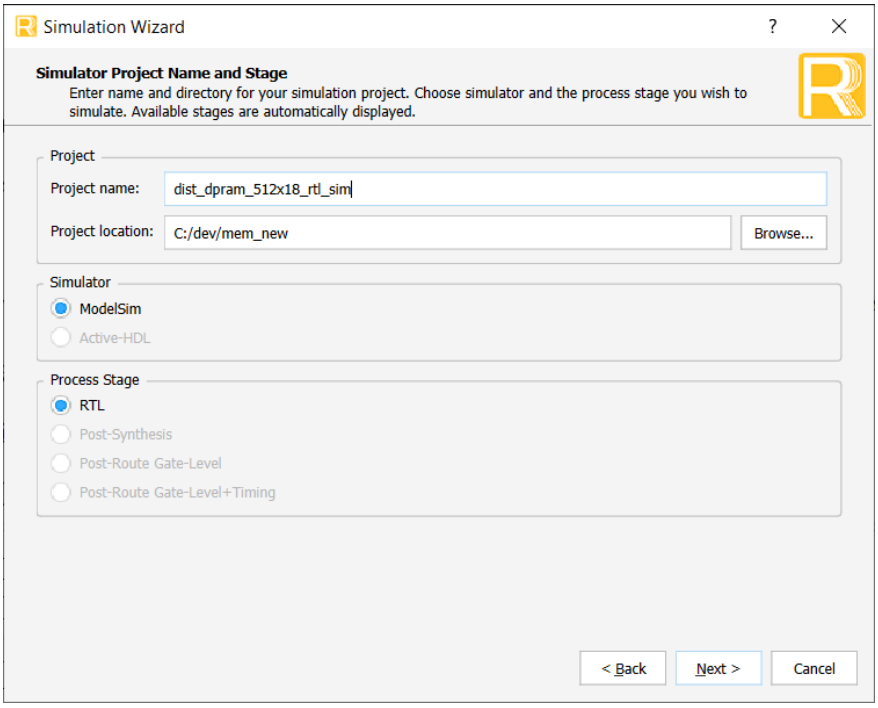

Figure 3.5. Simulation Wizard Simulator Project Name and Stage

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 3.6.

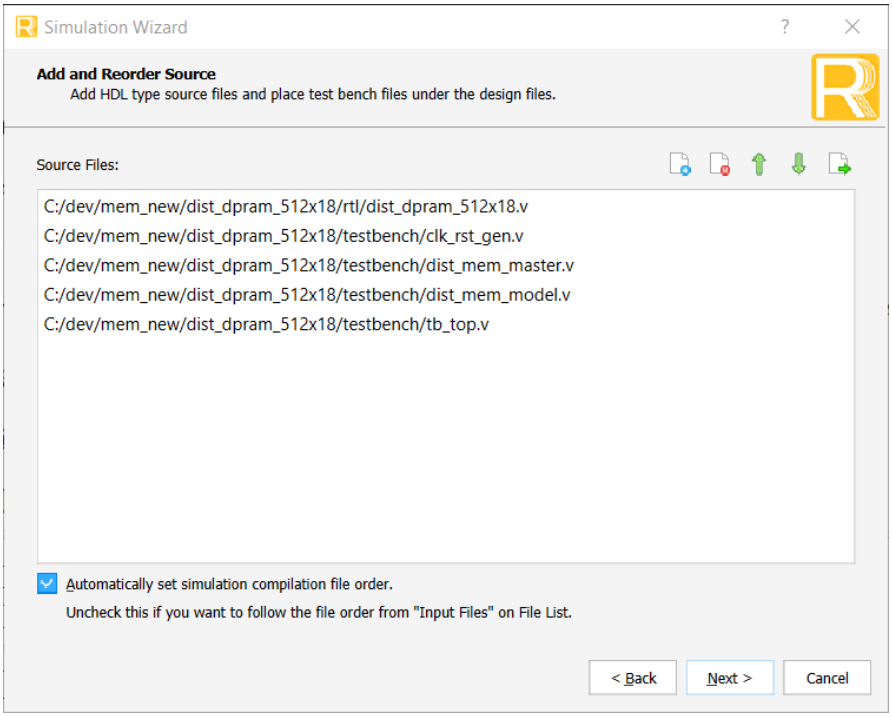

Figure 3.6. Simulation Wizard Add and Reorder Source

3. Click **Next** to open the **Parse HDL files for simulation**. Set **tb_top** as the **Simulation Top Module** as shown in Figure 3.7.

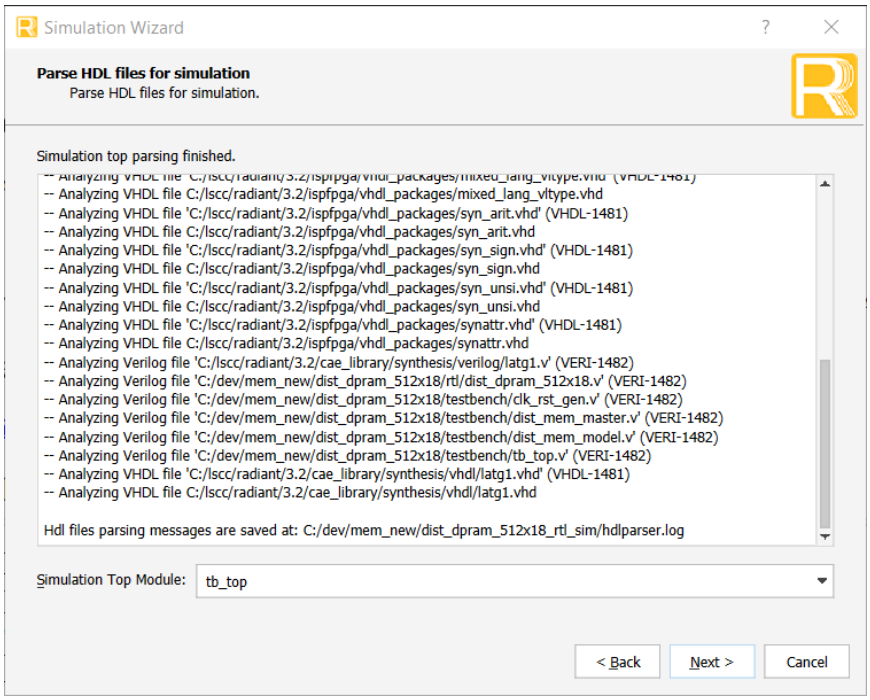

Figure 3.7. Simulation Wizard Parse HDL files for simulation

4. Click **Next**. The **Summary** window is shown. Click **Finish** to run the simulation. The complete waveform results of the simulation in this example are shown in Figure 3.8.

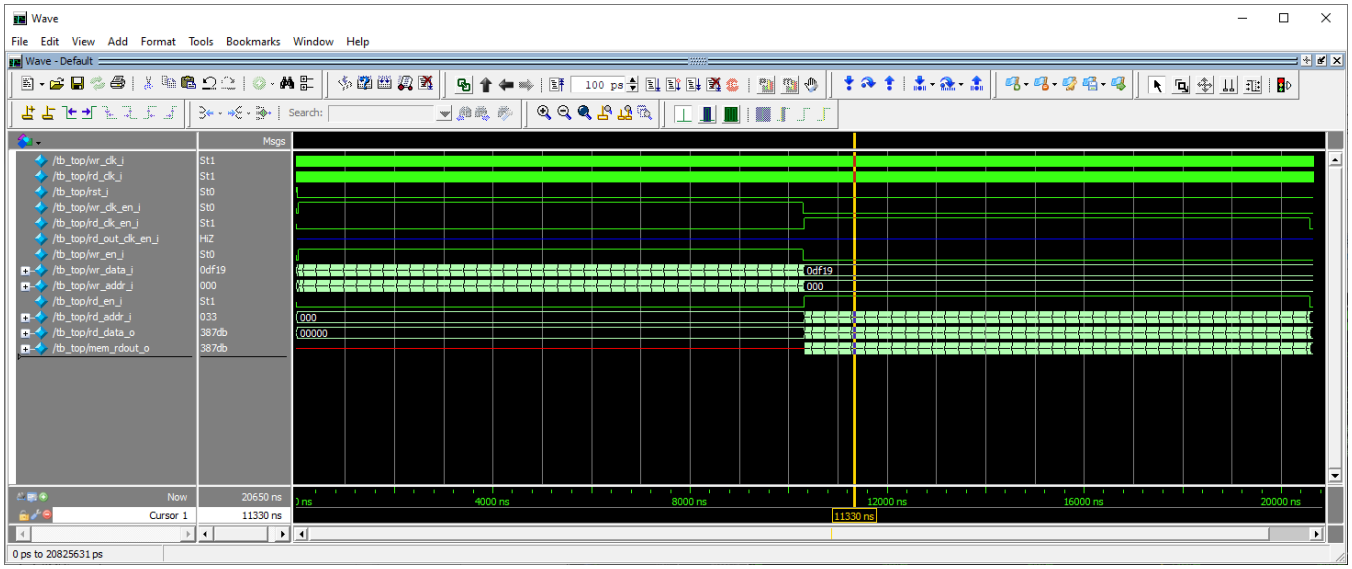

Figure 3.8. Simulation Waveform

4. PMI Support

The following sections discuss the different PMI modules which can be utilized for entering custom parameters for the distributed single port RAM, distributed pseudo dual port RAM, and distributed read only memory. 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_distributed_spram

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

Table 4.1. Distributed Single Port 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 4.2. Distributed Single Port 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. (See the Initializing Memory section for details on different formats.)	“binary”, “hex”	“binary”
pmi_family	This option selects the product family used. Defaults to common.	“LAV-AT", “common”	“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;
```

4.2. pmi_distributed_dpram

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

Table 4.3. Distributed Pseudo Dual Port Memory 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 4.4. Distributed Pseudo Dual Port Memory 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 the Port Address width. Equal to $\log_2$(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 the Memory file is selected, the 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. (See the Initializing Memory section for details on different formats.)	“binary”, “hex”	“binary”
pmi_family	Defines the FPGA family being used in the module	“LAV-AT", “common”	“common”
module_type	Refers to the type of pmi module.	“pmi_distributed_dpram”	“pmi_distributed_dpram”

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;
```

4.3. pmi_distributed_rom

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

Table 4.5. Distributed 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 4.6. Distributed Read Only Memory PMI 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 $\log_2$(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 the Memory file is selected, the 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. (See the Initializing Memory section for details on different formats.)	“binary”, “hex”	“binary”
pmi_family	Defines the FPGA family being used in the module	“LAV-AT", “common”	“common”
module_type	Refers to the type of pmi module.	“pmi_distributed_rom”	“pmi_distributed_rom”

Verilog `pmi_distributed_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_rom` 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;;
```

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 the 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 memory 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 indicates the width of the memory.

Memory Size 20x32

```
00100000010000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
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
```

Appendix A. Resource Utilization

Table A.1. Device and Tool Tested

	Value
Lattice Radiant Software Version	Radiant Software 2022.1 (for Windows)
Device Used	LAV-AT-500E
Synthesis Tool	Synplify Pro (R) S-2021.09LR-SP2-Beta3, Build 140R, Sep 14 2022

Table A.2. Distributed Single Port Memory Utilization

Memory Size	REG_EN	Synthesis Tool	Register	Logic LUTs	Distributed RAM	EBR
256 × 16	“reg”	Synplify Pro	16	105	384	0
4096 x 8	“noreg”	Synplify Pro	0	964	3072	0

Table A.3. Distributed Pseudo Dual Port Memory Utilization

Memory Size	REG_EN	Synthesis Tool	Register	Logic LUTs	Distributed RAM	EBR
512 × 16	“reg”	Synplify Pro	16	241	768	0
2048 x 32	“noreg”	Synplify Pro	0	1589	6144	0

Table A.4. Distributed Read Only Memory Utilization

Memory Size	REG_EN	Synthesis Tool	Register	LUTs	EBR
256 × 32	“noreg”	Synplify Pro	0	616	0
16384 x 16	“reg”	Synplify Pro	16	3513	0

Technical Support Assistance

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

Revision History

Revision 1.0, Lattice Radiant SW Version 2022.1, November 2022

Section	Change Summary
PMI Support	Updated pmi_family parameter values
Appendix A. Resource Utilization	Updated to specify Distributed RAM and Logic LUTs count

Revision 0.8, Lattice Radiant SW Version 3.2, May 2022

Section	Change Summary
All	Initial release

www.latticesemi.com