On-Chip Memory Usage Guide for LatticeSC Devices

November 2008 Technical Note TN1094

Introduction

This technical note discusses memory usage in the LatticeSC™ family of devices. It is intended for design engineers as a guide to designing and integrating the EBR-based and PFU-based memories of the LatticeSC device family using Lattice ispLEVER® design software.

The LatticeSC architecture provides many resources for memory intensive applications. The sysMEM™ Embedded Block RAM (EBR) complements its distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM, FIFO and ROM memories can be constructed using the EBR. LUTs and PFU can implement Distributed Single-Port RAM, Dual-Port RAM and ROM. The internal logic of the device can be used to configure the memory elements as FIFO and other storage types.

The EBR Block RAM and PFU RAM are referred to as memory primitives and their capabilities are described later in this document. The memory primitives can be used in two ways:

- Using **IPexpress™** – The IPexpress GUI allows users to specify the memory type and size required. IPexpress takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.
- Using the **PMI (Parameterizable Module Inferencing)** – PMI allows experienced users to skip the graphical interface and utilize the configurable memory modules on the fly from the ispLEVER Project Navigator. The parameters and control signals can be set in either Verilog or VHDL. The top-level design includes the defined parameters and signals so the interface can automatically generate the black box during synthesis.

The remainder of this document discusses these approaches as well as Memory Modules and Memory Primitives.

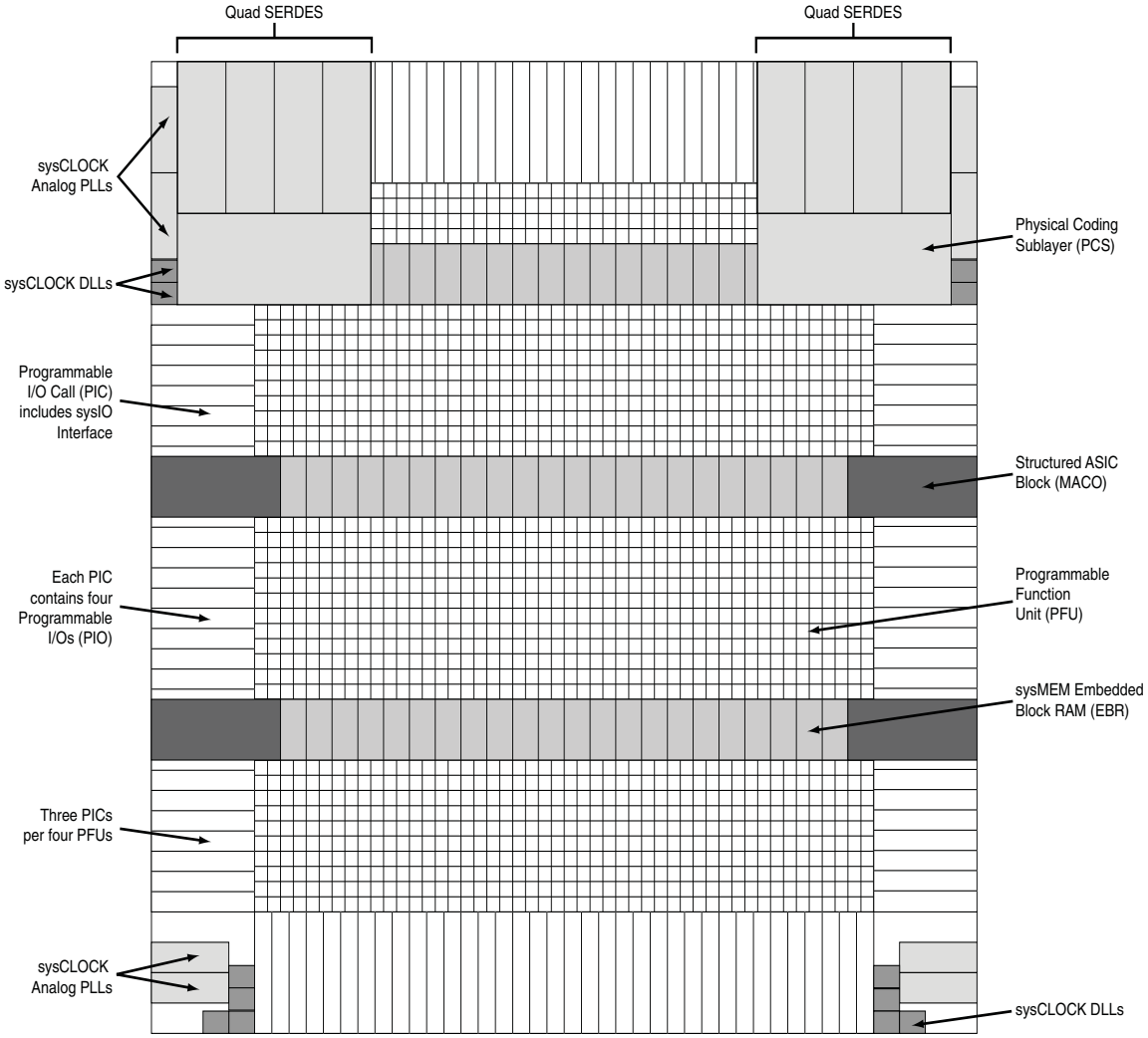
Memories in LatticeSC Devices

The LatticeSC architecture contains an array of logic blocks called PFUs surrounded by Programmable I/O Cells (PICs). sysMEM EBRs are large dedicated fast memory blocks. They can be configured as RAM, ROM or FIFO. These blocks have dedicated logic to simplify the implementation of FIFOs.

The PFU, PIC and EBR blocks are arranged in a two-dimensional grid with rows and columns as shown in Figure 1. These blocks are connected with many vertical and horizontal routing channel resources. The place and route software tool automatically allocates these routing resources.

Refer to the LatticeSC Family Data Sheet for details on the hardware implementation of the EBR and Distributed RAM.

Figure 1. Simplified Block Diagram (Top Level)



Utilizing IPexpress

Designers can use IPexpress to easily specify a variety of memories in their designs. These modules are constructed by using one or more memory primitives or the Programmable Functional Unit (PFU). The available primitives are:

- Single Port RAM (RAM_DQ) – EBR based
- Dual PORT RAM (RAM_DP_TRUE) – EBR based
- Pseudo Dual Port RAM (RAM_DP) – EBR based
- Read Only Memory (ROM) – EBR Based
- First In First Out Memory (FIFO_DC) – EBR Based
- Distributed Single Port RAM (Distributed_SPRAM) – PFU based
- Distributed Dual Port RAM (Distributed_DPRAM) – PFU based
- Distributed ROM (Distributed_ROM) – PFU/PFF based
- Distributed Shift Register (RAM_Based_Shift_Register) - PFU based (see IPexpress Help for details)

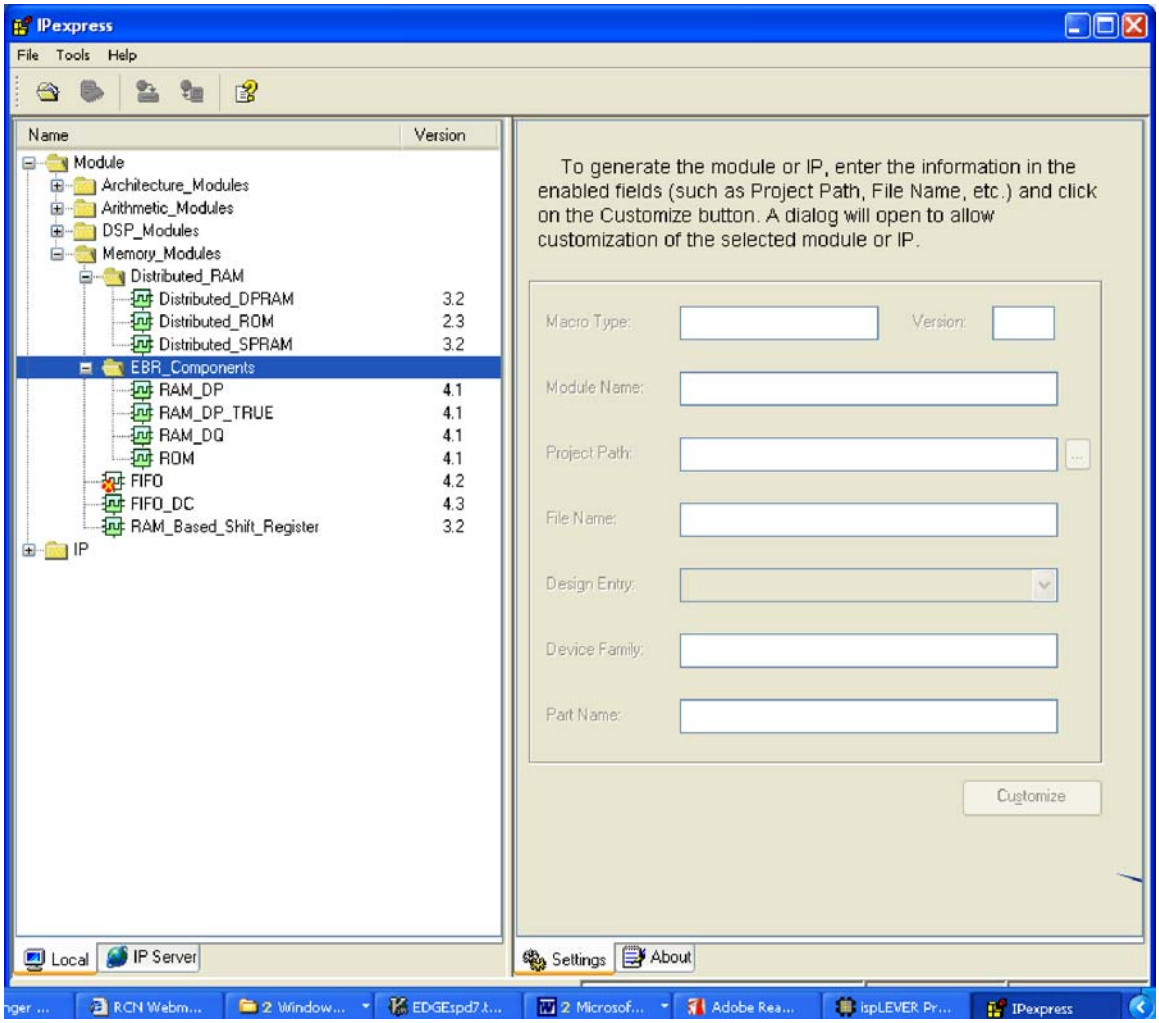
IPexpress Flow

For generating any of these memories, create (or open) a project with a LatticeSC device.

From the Project Navigator, select **Tools -> IPexpress.** Alternatively, click on the **IPexpress** button in the Project Navigator toolbar when the LatticeSC devices are targeted in the project.

This opens the IPexpress window as shown in Figure 2.

Figure 2. IPexpress Main Window



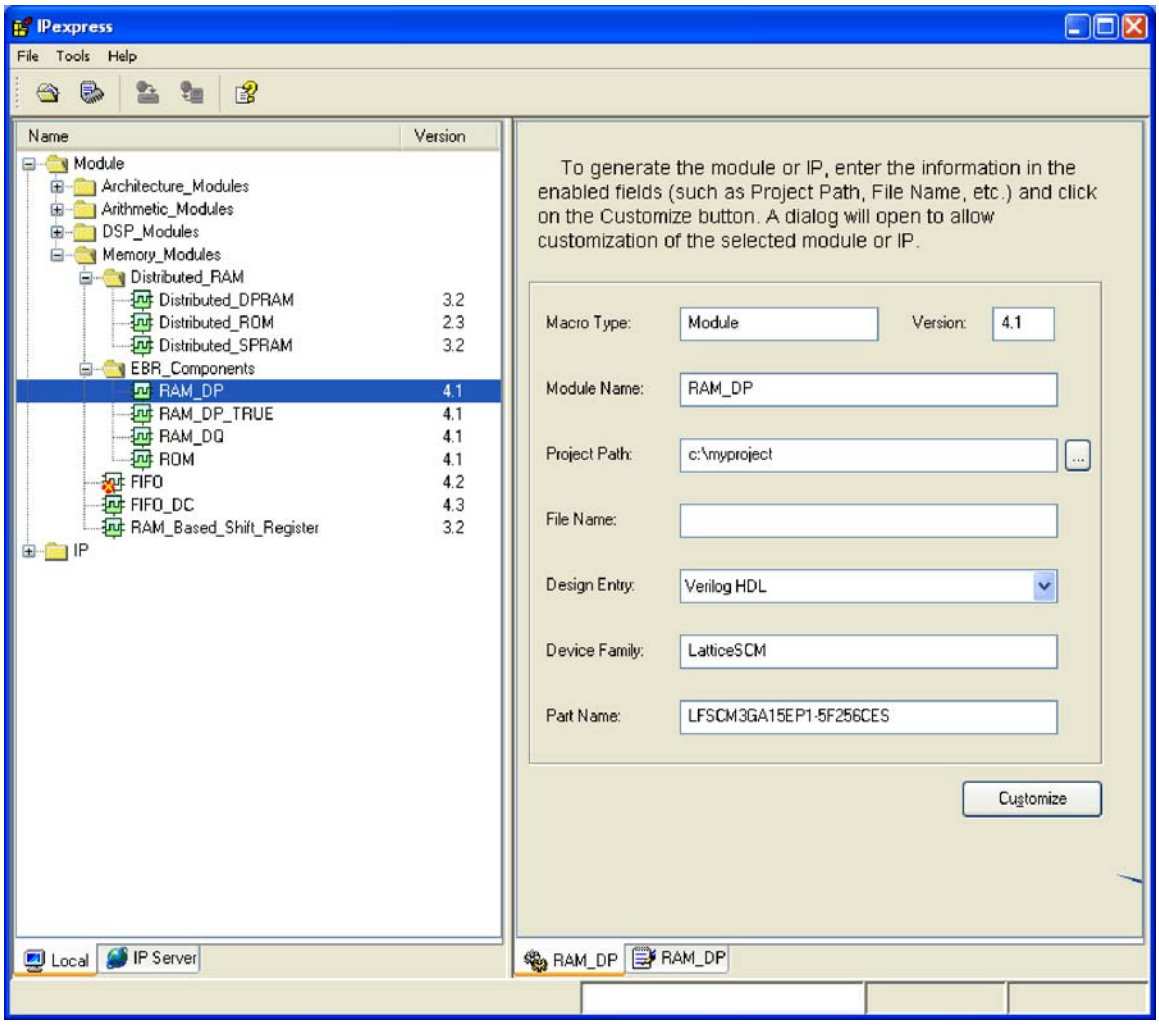
The left pane of this window displays the Module Tree. The EBR-based Memory Modules can be found under the **EBR_Components** folder and the PFU-based Distributed Memory Modules can be found under the **Distributed_RAM** folder, as shown in Figure 2. After selecting a module, enter the information necessary to generate the module (project path, file name, etc.) in the right pane of this IPexpress window.

Example Module Generation

The following example describes the generation of an EBR-based Pseudo Dual Port RAM of size 512 x 16.

In the left pane of the IPexpress window, under **EBR Components**, select **RAM_DP**. The right pane changes as shown in Figure 3.

Figure 3. Example Generating Pseudo Dual Port RAM (RAM_DP) Using IPexpress



In the right pane of the IPexpress window, the Macro Type, Macro Version, and Module Name, Project Path, Design Entry, Device Family, and Part Name are automatically populated when selecting a memory module (e.g. RAM_DP). The Project Path, Device Entry, Device Family and Part Name are derived from the information specified when creating the project in Project Navigator. The Project Path, File Name, and Design Entry are the only options that can be modified in this window.

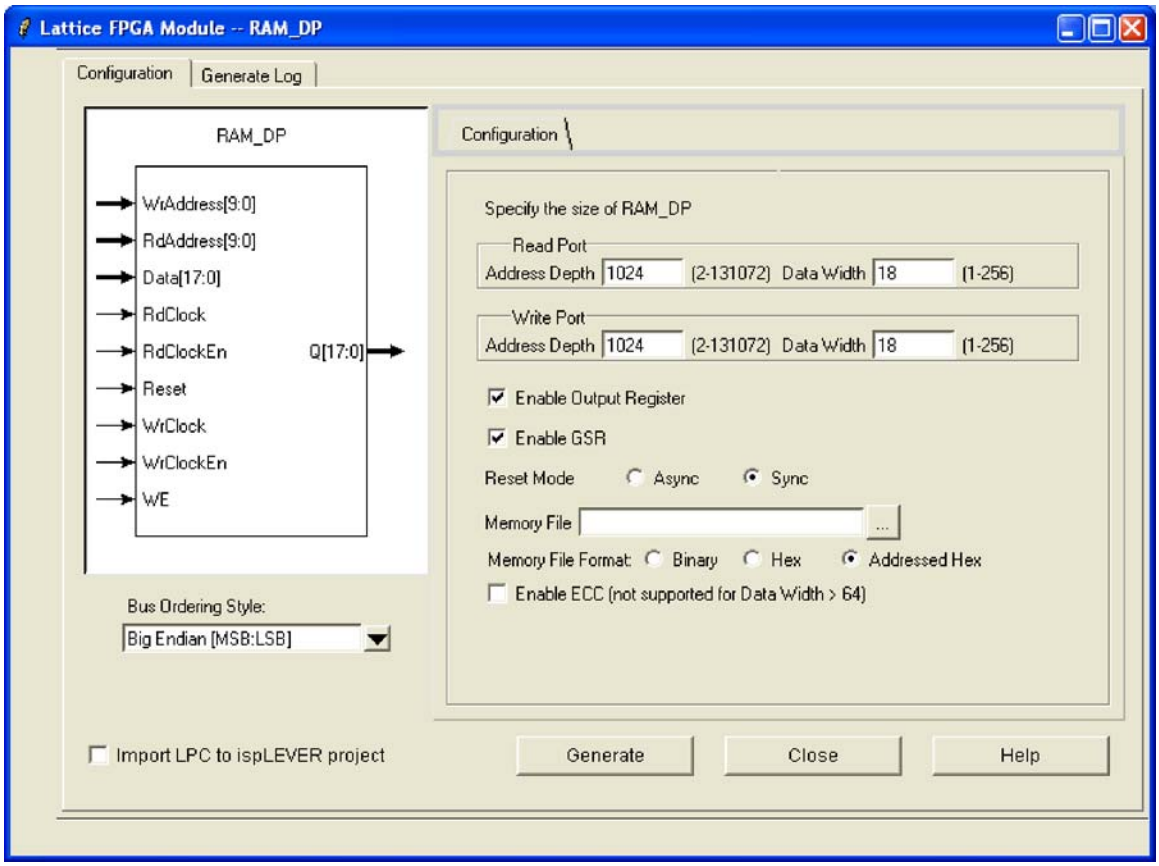
Project Path – The location to place the files created by IPexpress. For this example, we used: **C:\myproject**.

File Name – The name for the module to be generated. This name is used as the root name for the files being created in IPexpress. For this example, enter **ram_dp_512_16**.

Design Entry – The type or format of the design entry files. The choices are Schematic/VDHL, Schematic Verilog HDL, VHDL, or Verilog VHDL. For this example, use the default of **Verilog HDL**.

After all information is entered into the fields, select **Customize**. A module configuration window appears and is similar to Figure 4.

Figure 4. Generating Pseudo Dual Port RAM (RAM_DP) Module Customization – Configuration Tab



Read Port and **Write Port** – The Read Port and Write Port for the module selected defaults to an Address Depth of 1028 and a Data Width of 18 for the RAM_DP module. Port Depths and Widths can also be changed for Pseudo Dual Port and True Dual Port RAMs. The two ports can be of different widths. If different, the memory initialization file corresponds to the Write Port. For example, for a RAM_DP with Write Port of 1024x36 and Read Port of 2048x18, the initialization file should be of size 1024x36. For this example, change the **Address Depth** to **512**, and the **Data Width** to **16** for the Read Port and for the Write Port.

Enable Output Register – When selected, it causes the output registers to be inserted in the Read Data Port. (Output registers are optional for the EBR-based RAMs). The Input Data and Address Control are always registered since the hardware supports only the clocked write operation for the EBR-based RAMs. For this example, select this **check box**.

Enable GSR – When selected, this enables Global Set-Reset (GSR). For this example, select this **check box**.

Reset Mode – Choices are Async for asynchronous reset or Sync for synchronous reset. For this example, use the default value of **Sync**.

Memory File – Enter, or browse to select, the name of a Memory Initialization (.mem) File for the module. A MEM file consists of lines of addresses with corresponding data. If a memory file is not used, all the RAM contents are initialized to 0. This file is optional for the RAMs. However, it is required to provide the Memory file to the ROM. For this example, leave this field **blank**.

Memory File Format – Indicates the format of the memory file. Types of formats are Binary, Hex, or Addressed Hex. For this example, use the default value of **Addressed Hex**.

Enable ECC – When selected, allows error correction of single errors and error detection on all errors. This option is not supported for data widths greater than 64. For this example, this should be **un-checked**.

Bus Ordering Style – Specifies whether the numbers should begin at the high end (Big Endian) or the low end (Little Endian). For this example, this should be **Big Endian**.

Import LPC to ispLEVER Project – When selected, imports the Lattice Parameter configuration (.lpc) file into the project currently open in the ispLEVER Project Navigator. This option is available when IPexpress is invoked from within ispLEVER Project Navigator. This option is not available when running IPexpress as a stand-alone tool.

After specifying the desired configurations, select the **Generate** button to generate the customized module. A netlist in the desired format is generated and placed in the specified directory location. The netlist can now be incorporated in a design.

Once the module is generated, the *.lpc or the Verilog/VHDL file can be instantiated in the top-level module of a design. Note that although the .lpc file can be instantiated in the top-level module, the port names for instantiation can be obtained from the HDL files generated. The .lpc file instantiation allows users to double-click on the file in the Project Navigator and open IPexpress loaded with the parameters. This way, it is easier to edit or make changes.

The various memory modules, both EBR and Distributed, are discussed in detail later in this document.

Utilizing the PMI

Parameterizable Module Inferencing (PMI) allows experienced users to skip the graphical interface and utilize the configurable memory modules on the fly from the ispLEVER Project Navigator.

Users can instantiate the component of the memory module directly in their design, instead of using the module generated by IPexpress. The instantiated component allows users to change the parameters and attributes of the module from within Verilog or VHDL code. The instantiated component can be simulated too. The top level of the design includes the defined memory parameters and signals so the interface can automatically generate the black box during synthesis and ispLEVER can generate the net-list on the fly.

To include the component in the source code:

1. Create a design and open the source code in the Lattice Text Editor. The PMI flow is supported only through the Lattice Text Editor. If the users choose to use another text editor, they can include the component using the Lattice Text Editor and then go back to the editor of their choice.
2. Click the cursor at the place where the PMI component should be inserted.
3. Click on the **Templates** Menu, and select **Insert**. (Alternatively, press F9 as a shortcut to this command).
4. This opens the **Insert Template** window. The left pane of this window includes the **Template Files**, where users choose the appropriate template for their device. Also, Verilog or VHDL can be selected. The right-hand pane of the window, which shows the different Template Names, allows users to select the module component to be used.
5. Select the appropriate template in the Template Name pane and click **Insert**.
6. Click **Close** to close the Insert Template window.
7. The ports of the inserted template then need to be mapped in the design. Once this is complete, users can synthesize, map and place and route the design.

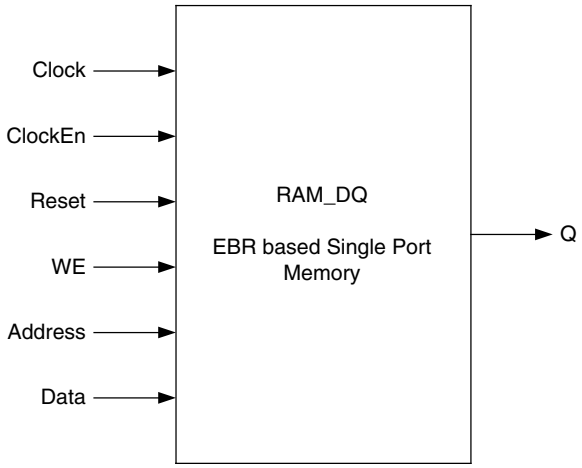
Memory Modules

Single Port RAM (RAM_DQ) - EBR Based

The EBR blocks in LatticeSC devices can be configured as Single Port RAM or RAM_DQ. IPexpress allows users to generate a Verilog or VHDL netlist along with an EDIF netlist for the memory type selected.

IPexpress generates the memory module as shown in Figure 5.

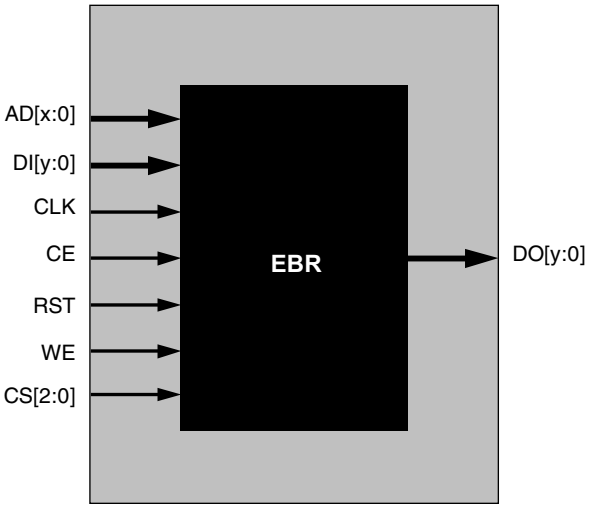
Figure 5. Single Port Memory Module Generated by IPexpress



The generated module makes use of the LatticeSC EBR blocks or primitives and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For memory sizes smaller than one EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

The memory primitive for RAM_DQ in LatticeSC devices is shown in Figure 6.

Figure 6. Single Port RAM primitive or RAM_DQ for LatticeSC



In Single Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions in Single Port Memory are listed in Table 1. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DQ primitive.

Table 1. EBR-based Single Port Memory Port Definitions

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

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

The Chip Select (CS) port in the EBR primitive is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus and can easily cascade eight memories. If the memory size specified requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 18,432 bits of RAM. The values for x (for Address) and y (Data) for each EBR block of Lattice LatticeSC device are listed in Table 2.

Table 2. Single Port Memory Sizes for 18K Memories for LatticeSC

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
16K x 1	DI	DO	AD[13:0]
8K x 2	DI[1:0]	DO[1:0]	AD[12:0]
4K x 4	DI[3:0]	DO[3:0]	AD[11:0]
2K x 9	DI[8:0]	DO[8:0]	AD[10:0]
1K x 18	DI[17:0]	DO[17:0]	AD[9:0]
512 x 36	DI[35:0]	DO[35:0]	AD[8:0]

Table 3 shows the various attributes available for the Single Port Memory (RAM_DQ). Some of these attributes are user selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 3. Single Port RAM Attributes for LatticeSC

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH	Data Word Width	1, 2, 4, 9, 18, 36	18	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset Type	ASYNC, SYNC	ASYNC	YES
CSDECODE	Chip Select Decode	000, 001, 010, 011, 100, 101, 110, 111	000	NO
WRITEMODE	Read/Write Mode	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
DISABLED_GSR	Disable Global Set Reset	0, 1	0	YES
INIT	Initialization File for Memory	DISABLED, ENABLED	DISABLED	—
INIT_RECFG	Reconfiguring Initialization through MPI Bus	DISABLED, ENABLED	DISABLED	—
INIT_ID	Initialization ID for Initializations through MPI Bus	“0000000000”	—	—

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

Additionally users can select to enable the output registers for RAM_DQ. Figures 7 through 11 show the internal timing waveforms for the Single Port RAM (RAM_DQ) with these options.

Figure 7. Single Port RAM Timing Waveform – NORMAL Mode, without Output Registers

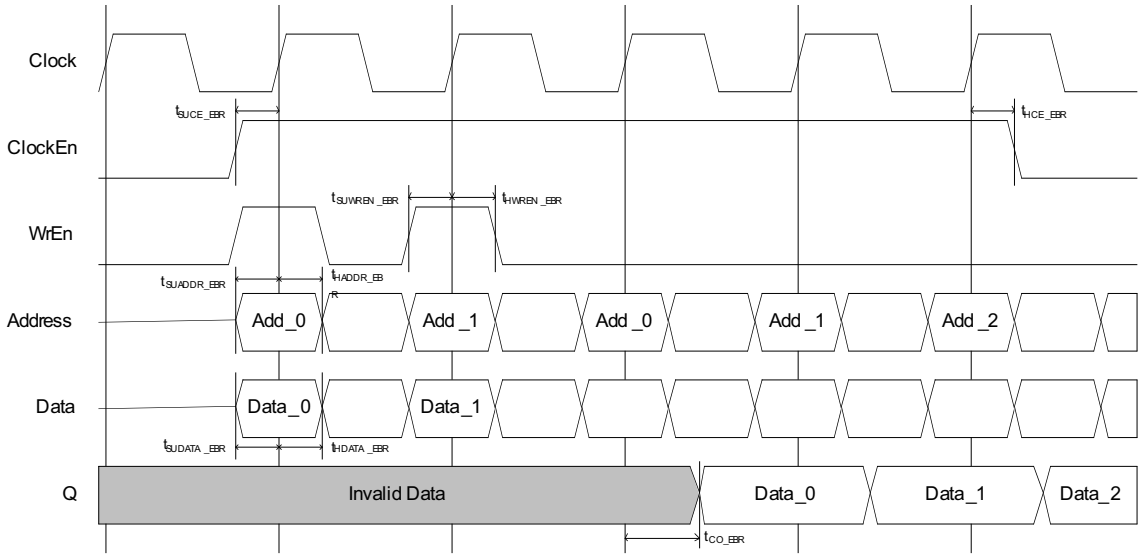


Figure 8. Single Port RAM Timing Waveform – NORMAL Mode, with Output Registers

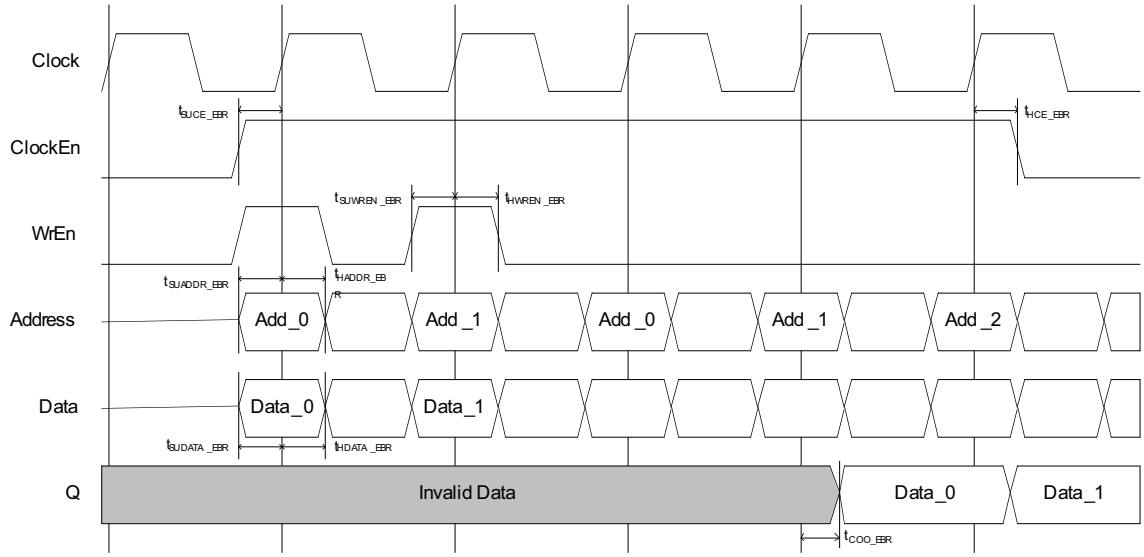


Figure 9. Single Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

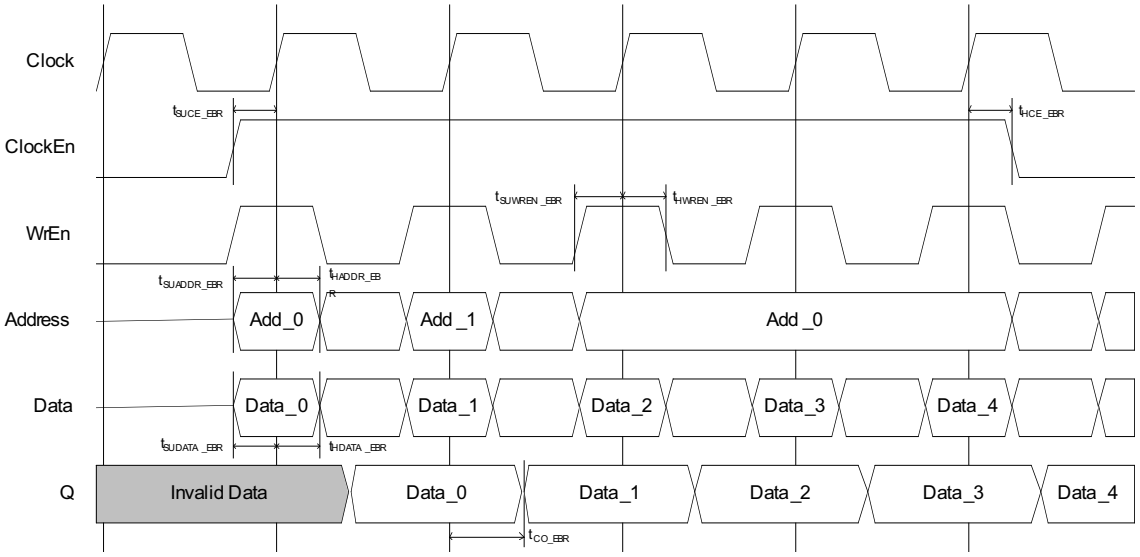
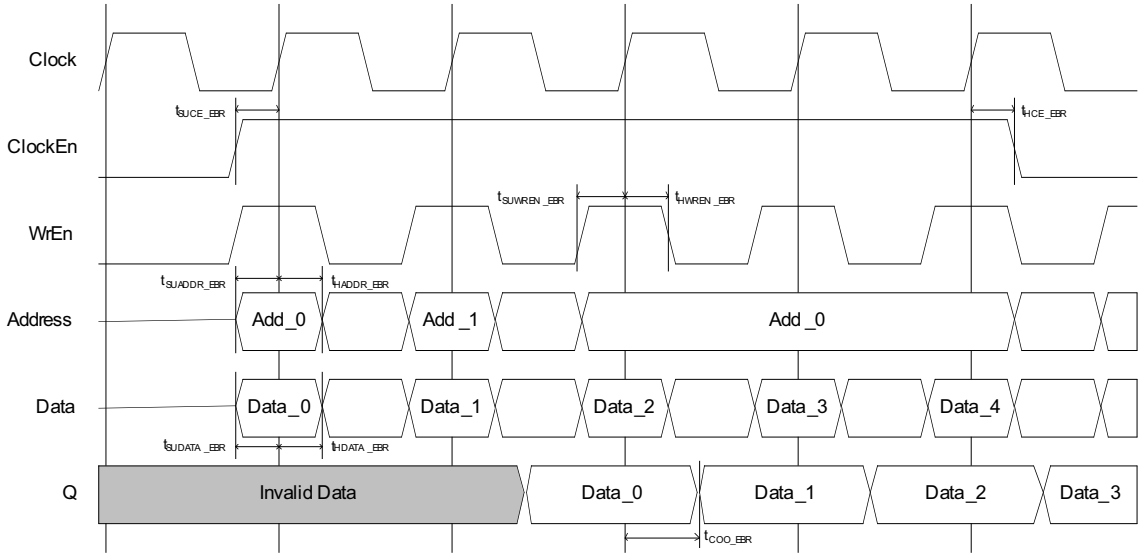


Figure 10. Single Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

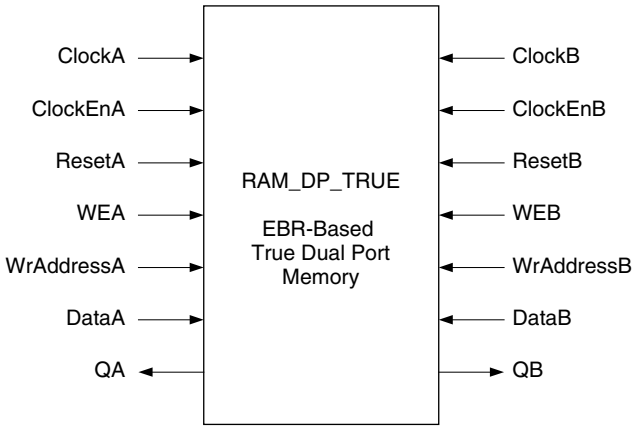


True Dual Port RAM (RAM_DP_TRUE) – EBR Based

The EBR blocks in LatticeSC devices can be configured as True Dual Port RAM or RAM_DP_TRUE. IPexpress allows users to generate a Verilog or VHDL netlist along with an EDIF netlist for the memory type selected.

IPexpress generates the memory module as shown in Figure 11.

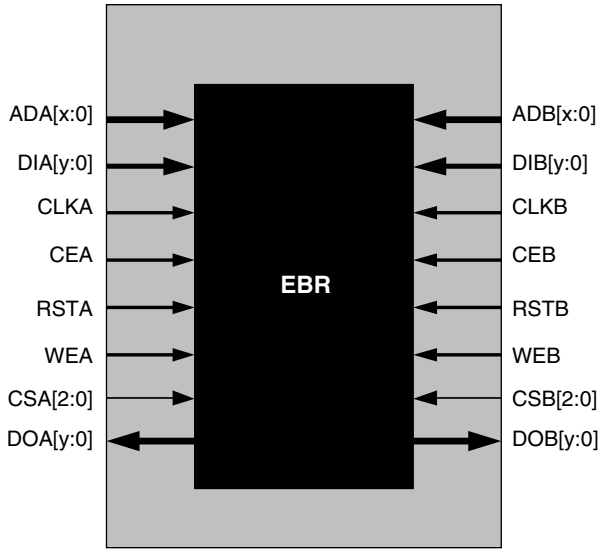
Figure 11. True Dual Port Memory Module generated by IPexpress



The generated module makes use of the LatticeSC EBR blocks or primitives and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For memory sizes smaller than one EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

The basic memory primitive for LatticeSC devices, RAM_DP_TRUE, is shown in Figure 12.

Figure 12. True Dual Port RAM Primitive or RAM_DP_TRUE for LatticeSC



In True Dual Port RAM mode the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for True Dual Port Memory are listed in Table 4. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP_TRUE primitive.

Table 4. EBR-based True Dual Port Memory Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
ClockA, ClockB	CLKA, CLKB	Clock for Port A and PortB	Rising Clock Edge
ClockEnA, ClockEnB	CEA, CEB	Clock Enable for Port A and PortB	Active High
AddressA, AddressB	ADA[x:0], ADB[x:0]	Address Bus for Port A and PortB	—
DataA, DataB	DIA[y:0], DIB[y:0]	Data In for Port A and PortB	—
QA, QB	DOA[y:0], DOB[y:0]	Data Out for Port A and PortB	—
WEA, WEB	WEA, WEB	Write Enable for Port A and PortB	Active High
ResetA, ResetB	RSTA, RSTB	Reset for Port A and PortB	Active High
—	CSA[2:0], CSB[2:0]	Chip Select for Port A and PortB	—

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

The Chip Select (CS) port in the EBR primitive is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus and can cascade eight memories easily. If the memory size specified requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) for each EBR block of the LatticeSC device are shown in Table 5.

Table 5. Dual Port Memory Sizes for 18K Memory in LatticeSC

Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Address Port A [MSB:LSB]	Address Port B [MSB:LSB]
16K x 1	DIA	DIB	DOA	DOB	ADA[13:0]	ADB[13:0]
8K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[12:0]	ADB[12:0]
4K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[11:0]	ADB[11:0]
2K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[10:0]	ADB[10:0]
1K x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	ADA[9:0]	ADB[9:0]

Table 6 shows the various attributes available in True Dual Port Memory (RAM_DP_TRUE). Some of these attributes are user selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 6. Dual Port RAM Attributes for LatticeSC

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH_A	Data Word Width Port A	1, 2, 4, 9, 18	18	YES
DATA_WIDTH_B	Data Word Width Port B	1, 2, 4, 9, 18	18	YES
REGMODE_A	Register Mode (Pipelining) for Port A	NOREG, OUTREG	NOREG	YES
REGMODE_B	Register Mode (Pipelining) for Port B	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset Type	ASYNC, SYNC	ASYNC	YES
CSDECODE_A	Chip Select Decode for Port A	000, 001, 010, 011, 100, 101, 110, 111	000	NO
CSDECODE_B	Chip Select Decode for Port B	000, 001, 010, 011, 100, 101, 110, 111	000	NO
WRITEMODE_A	Read/Write Mode for Port A	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
WRITEMODE_B	Read/Write Mode for Port B	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
DISABLED_GSR	Disable Global Set Reset	0, 1	0	YES
INIT	Initialization File for Memory	DISABLED, ENABLED	DISABLED	—
INIT_RECFG	Reconfiguring Initialization through MPI Bus	DISABLED, ENABLED	DISABLED	—
INIT_ID	Initialization ID for Initialization through MPI Bus	“0000000000”	—	—

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

Additionally, users can choose to enable the output registers for RAM_DP_TRUE. Figures 13 through 16 show the internal timing waveforms for the True Dual Port RAM (RAM_DP_TRUE) with these options.

Figure 13. True Dual Port RAM Timing Waveform – NORMAL Mode, without Output Registers

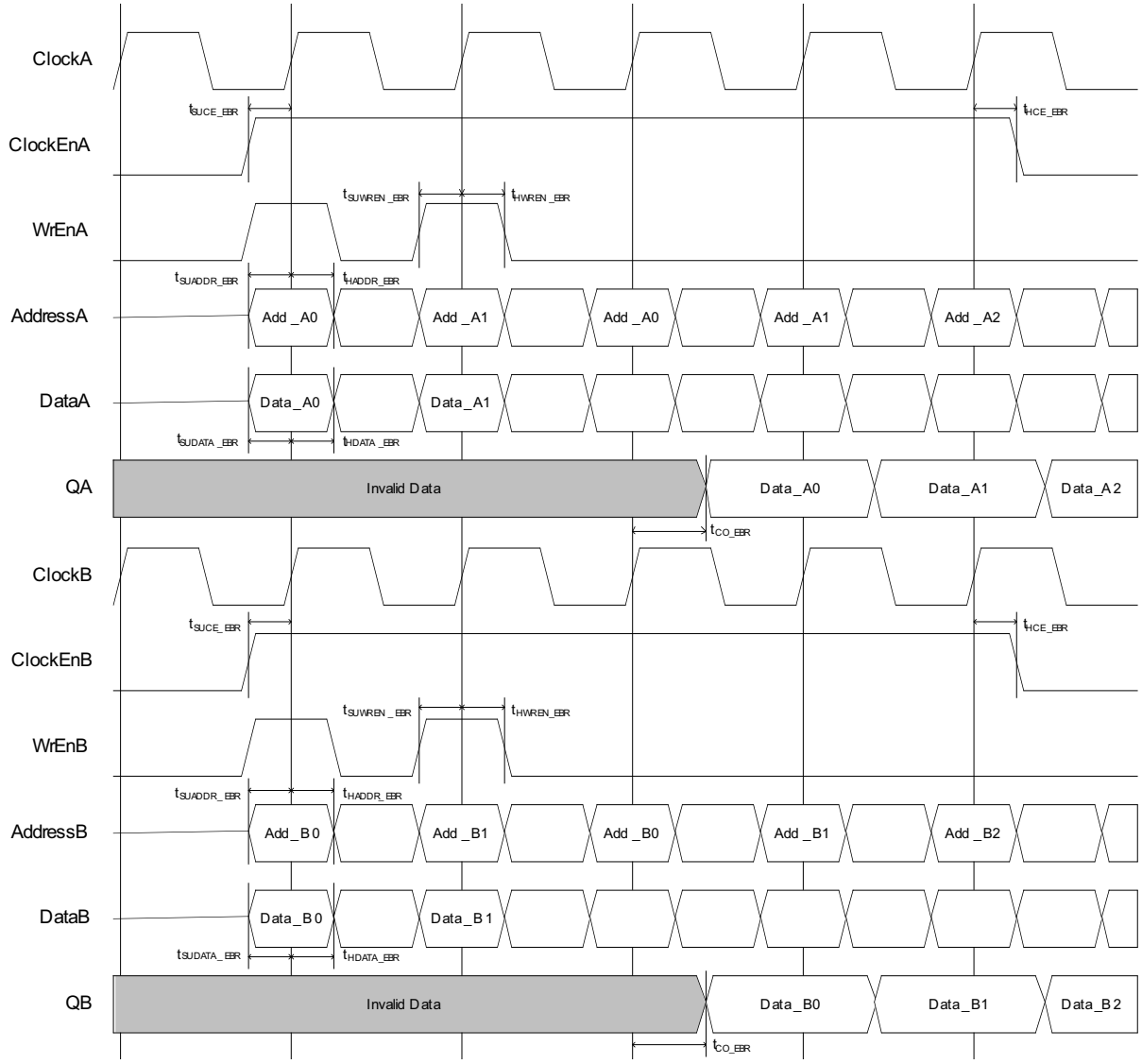


Figure 14. True Dual Port RAM Timing Waveform – NORMAL Mode with Output Registers

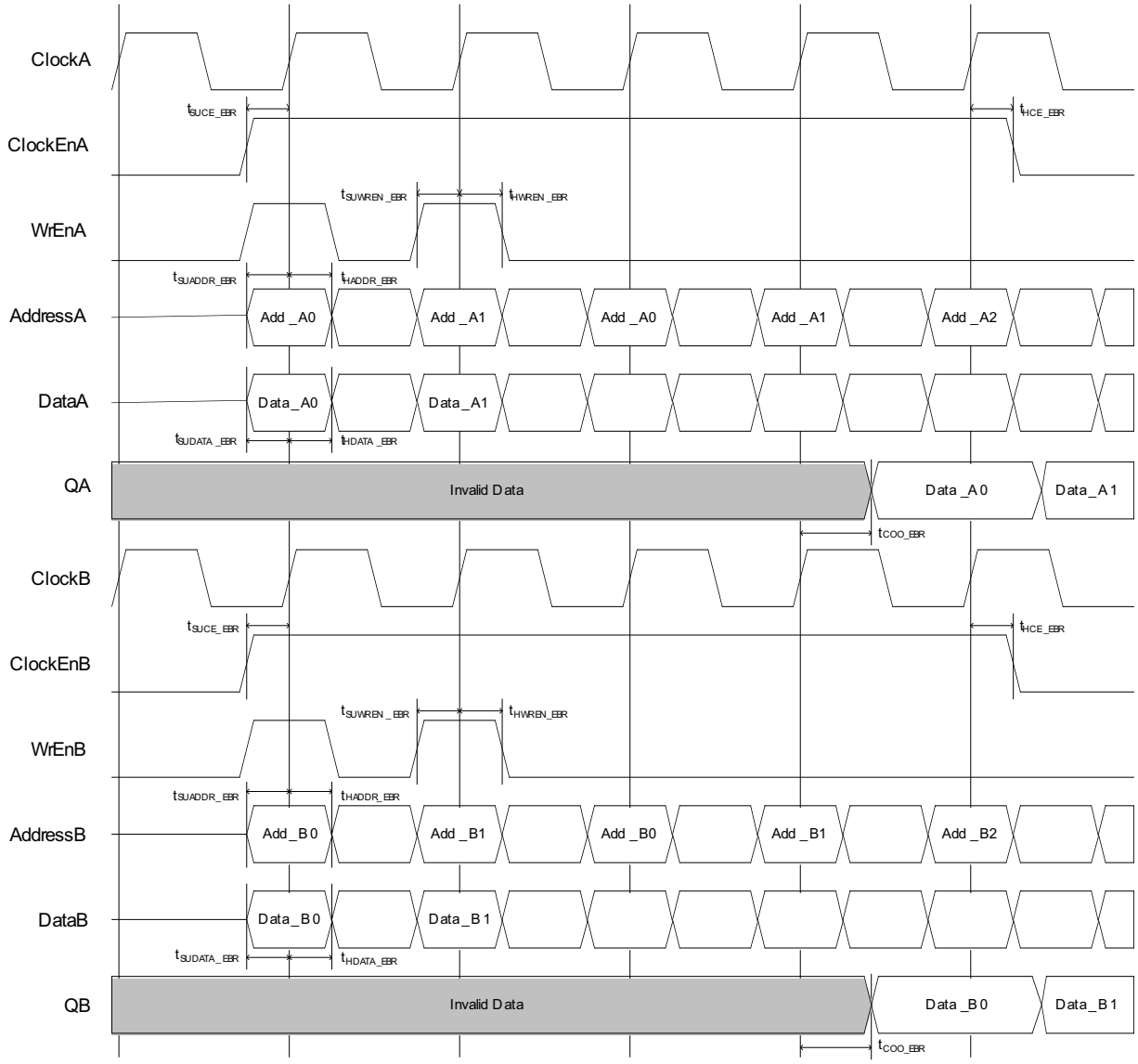


Figure 15. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

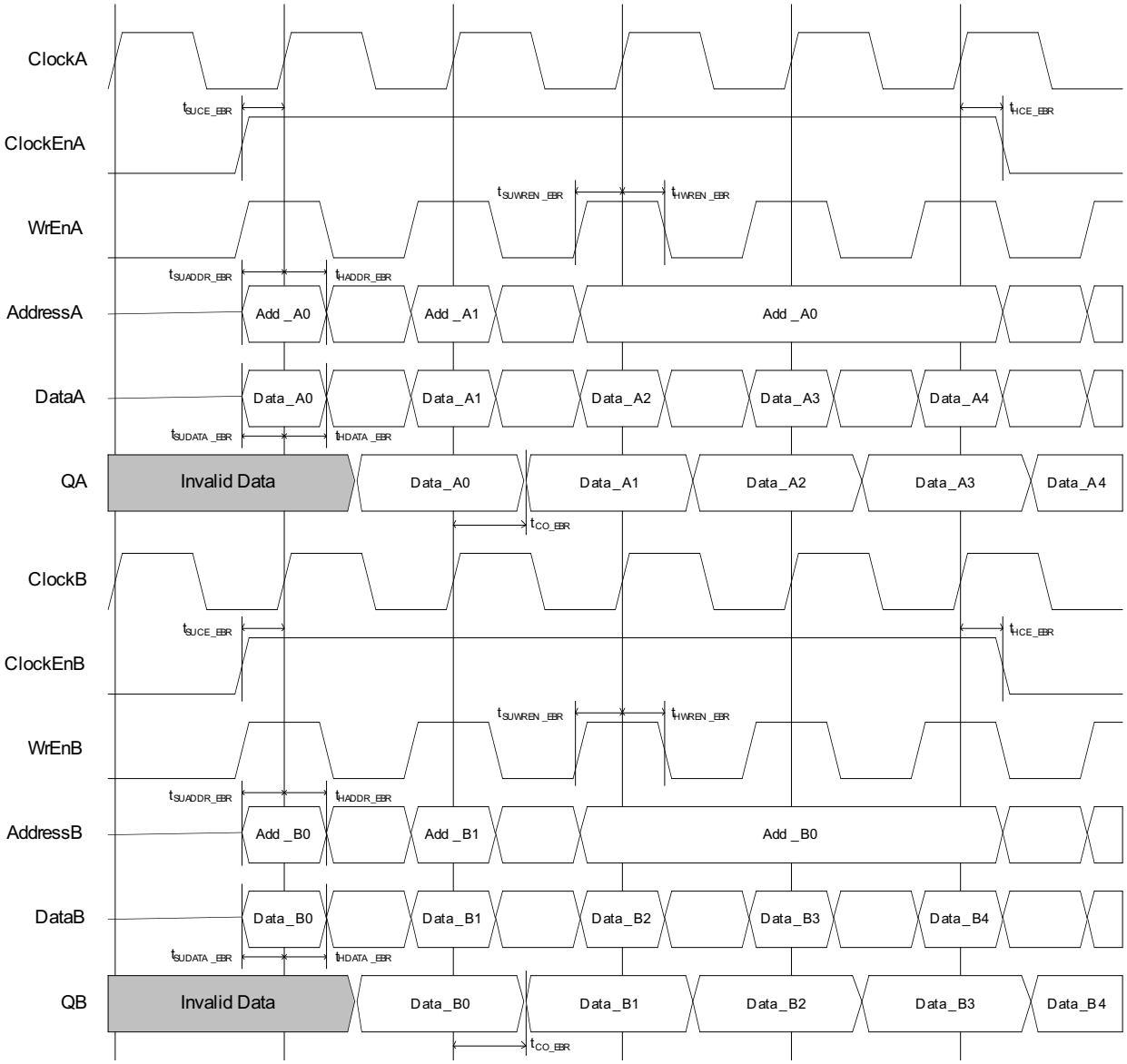
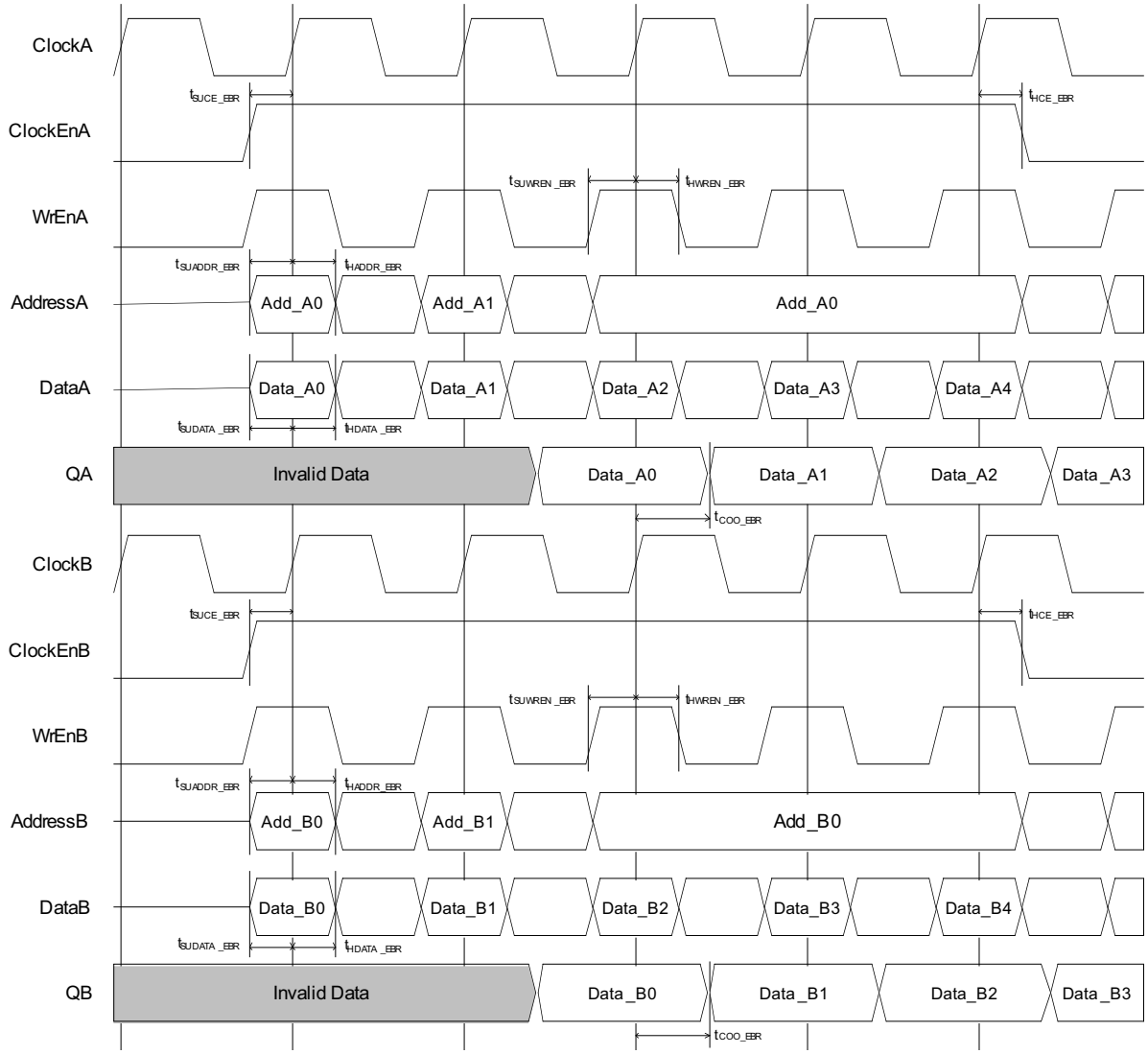


Figure 16. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

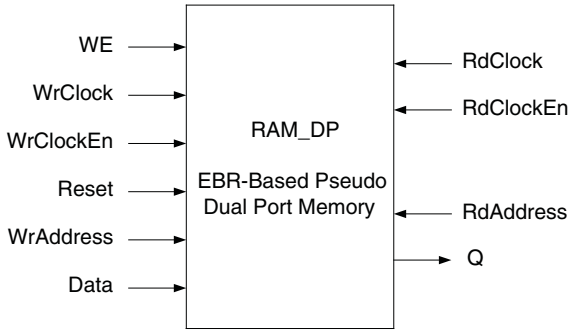


Pseudo Dual Port RAM (RAM_DP) – EBR Based

EBR blocks in LatticeSC devices can be configured as Pseudo-Dual Port RAM or RAM_DP. IPexpress allows users to generate Verilog or VHDL along with an EDIF netlist for the memory size as per design requirements.

IPexpress generates the memory module, as shown in Figure 17.

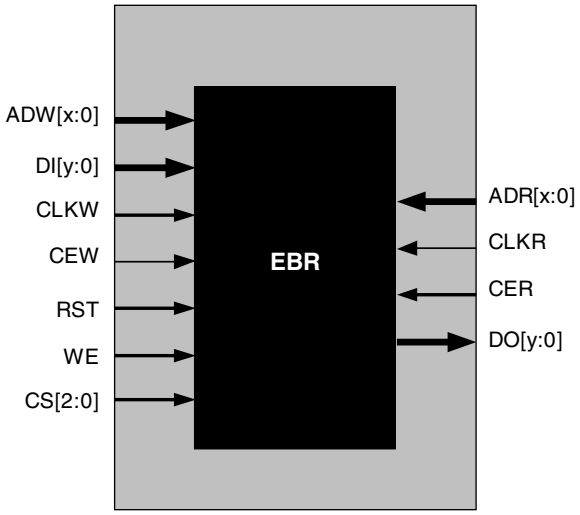
Figure 17. Pseudo Dual Port Memory Module Generated by IPexpress



The generated module makes use of the LatticeSC EBR blocks or primitives and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For memory sizes smaller than one EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

The basic Pseudo Dual Port memory primitive for the Lattice LatticeSC devices is shown in Figure 23.

Figure 18. Pseudo Dual Port RAM primitive or RAM_DP for LatticeSC



In the Pseudo Dual Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions in Pseudo Dual Port Memory are included in Table 7. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP primitive.

Table 7. EBR based Pseudo-Dual Port Memory Port Definitions

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

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

The Chip Select (CS) port in the EBR primitive is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus and can cascade eight memories easily. If the memory size specified requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) for each EBR block of the LatticeSC device are shown in Table 8.

Table 8. Pseudo-Dual Port Memory Sizes for 18K Memory for LatticeSC

Pseudo-Dual Port Memory Size	Input Data Port A (Write Port)	Output Data Port B (Read Port)	Read Address Port A [MSB:LSB]	Write Address Port B [MSB:LSB]
16K x 1	DI	DO	RAD[13:0]	WAD[13:0]
8K x 2	DI[1:0]	DO[1:0]	RAD[12:0]	WAD[12:0]
4K x 4	DI[3:0]	DO[3:0]	RAD[11:0]	WAD[11:0]
2K x 9	DI[8:0]	DO[8:0]	RAD[10:0]	WAD[10:0]
1K x 18	DI[17:0]	DO[17:0]	RAD[9:0]	WAD[9:0]
512 x 36	DI[35:0]	DO[35:0]	RAD[8:0]	WAD[8:0]

Table 9 shows the various attributes available in Pseudo Dual Port Memory (RAM_DP). Some of these attributes are user selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 9. Pseudo-Dual Port RAM Attributes for LatticeSC

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH_W	Write Data Word Width	1, 2, 4, 9, 18, 36	18	YES
DATA_WIDTH_R	Read Data Word Width	1, 2, 4, 9, 18, 36	18	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE_W	Chip Select Decode for Write	000, 001, 010, 011, 100, 101, 110, 111	000	NO
CSDECODE_R	Chip Select Decode for Read	000, 001, 010, 011, 100, 101, 110, 111	000	NO
DISABLED_GSR	Disable Global Set Reset	0, 1	0	YES
INIT	Initialization File for Memory	DISABLED, ENABLED	DISABLED	—
INIT_RECFG	Reconfiguring initialization through MPI Bus	DISABLED, ENABLED	DISABLED	—
INIT_ID	Initialization ID for initializations through MPI Bus	“0000000000”	—	—

Users have the option of enabling the output registers for Pseudo-Dual Port RAM (RAM_DP). Figures 19 and 20 show the internal timing waveforms for the Pseudo-Dual Port RAM (RAM_DP) with these options.

Figure 19. PSEUDO DUAL PORT RAM Timing Diagram - without Output Registers

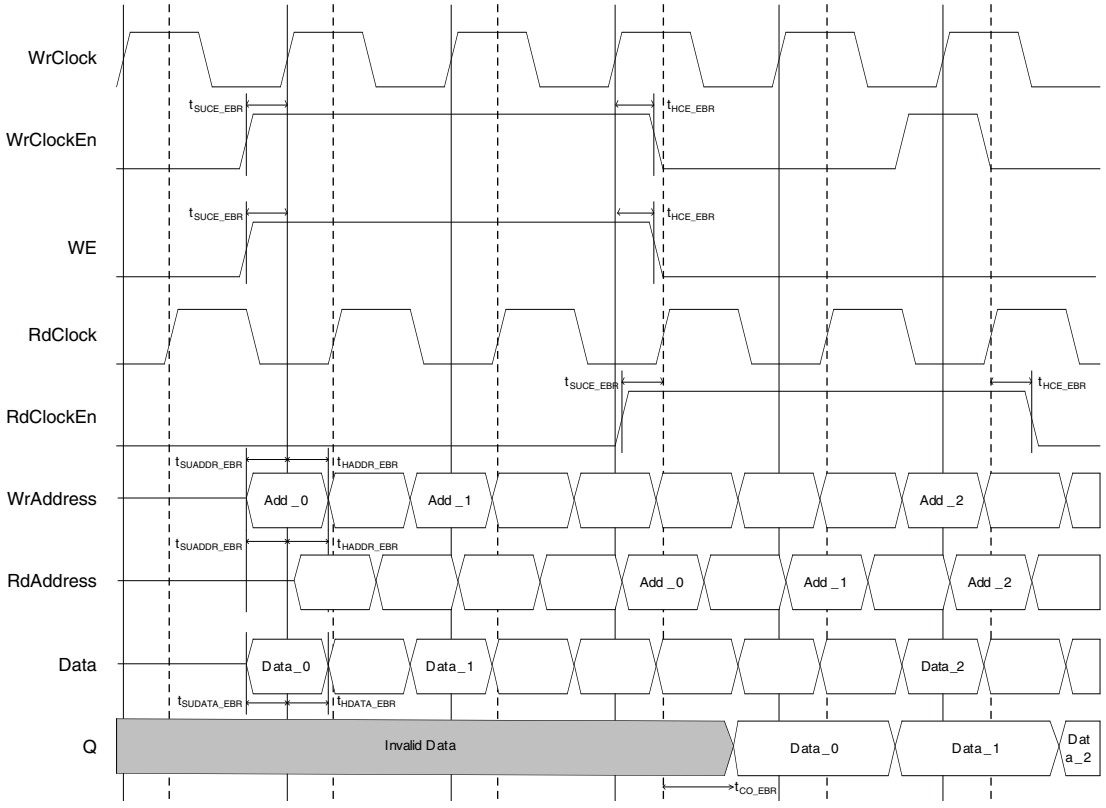
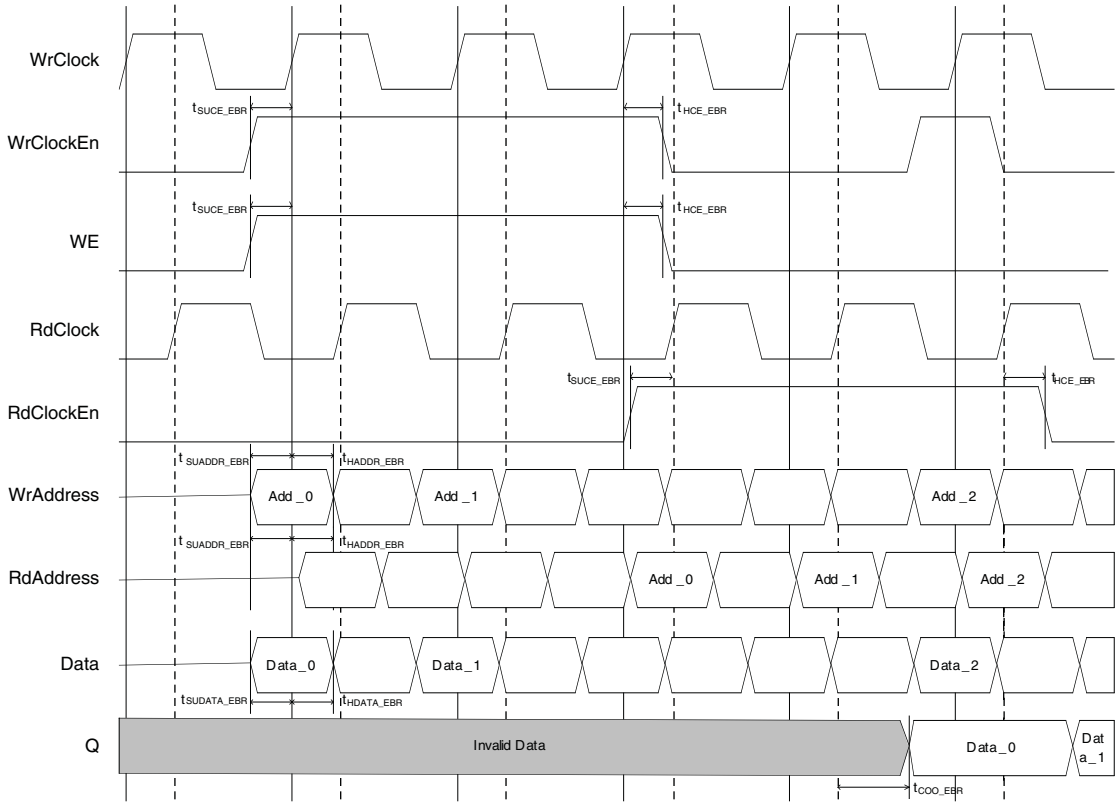


Figure 20. PSEUDO DUAL PORT RAM Timing Diagram - with Output Registers

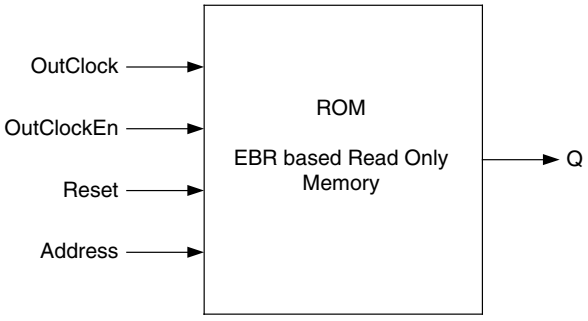


Read Only Memory (ROM) - EBR Based

EBR blocks in LatticeSC devices can be configured as Read Only Memory or ROM. IPexpress allows users to generate Verilog or VHDL along with an EDIF netlist for the memory size as per design requirements. Users are required to provide the ROM memory content in the form of an initialization file.

IPexpress generates the memory module as shown in Figure 21.

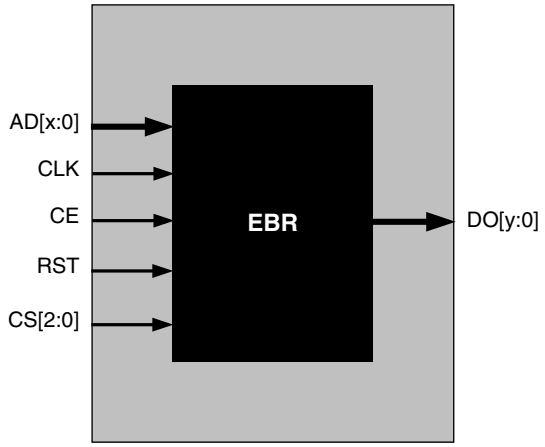
Figure 21. ROM – Read Only Memory Module Generated by IPexpress



The generated module makes use of the LatticeSC EBR blocks or primitives and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For memory sizes smaller than one EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

The basic ROM primitive for LatticeSC devices is shown in Figure 22.

Figure 22. ROM primitive for LatticeSC



In ROM mode, the address for the port is registered at the input of the memory array. The memory output data is optionally registered at the output.

The various ports and their definitions for ROM mode are listed in Table 10. The table lists the corresponding ports for the module generated by IPexpress and for the ROM primitive.

Table 10. EBR-based ROM Port Definitions

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

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

The Chip Select (CS) port in the EBR primitive is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus and can cascade eight memories easily. If the memory size specified requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

When generating the ROM using IPexpress, the user is required to provide an initialization file to pre-initialize the contents of the ROM. These *.mem files can be of Binary, Hex or Addressed Hex formats. The initialization files are discussed in detail in the Initializing Memory section of this technical note.

Users have the option of enabling the output registers for Read Only Memory (ROM). Figures 23 and 24 show the internal timing waveforms for ROM with these options.

Figure 23. ROM Timing Waveform – without Output Registers

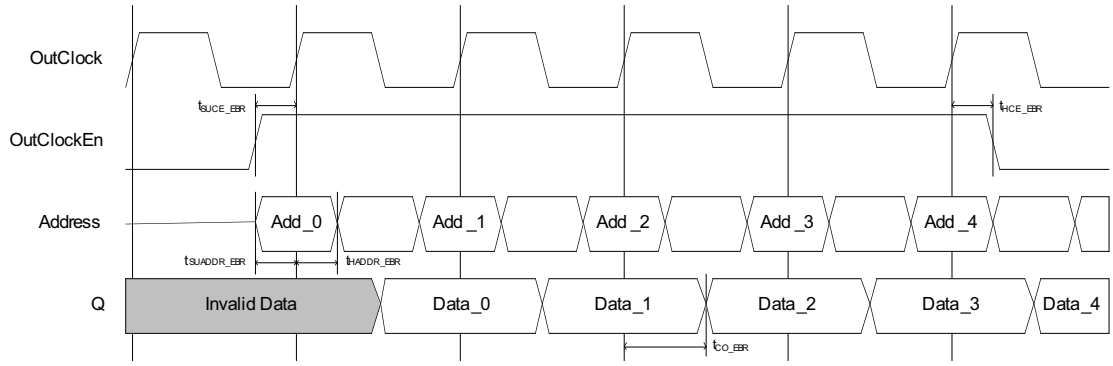
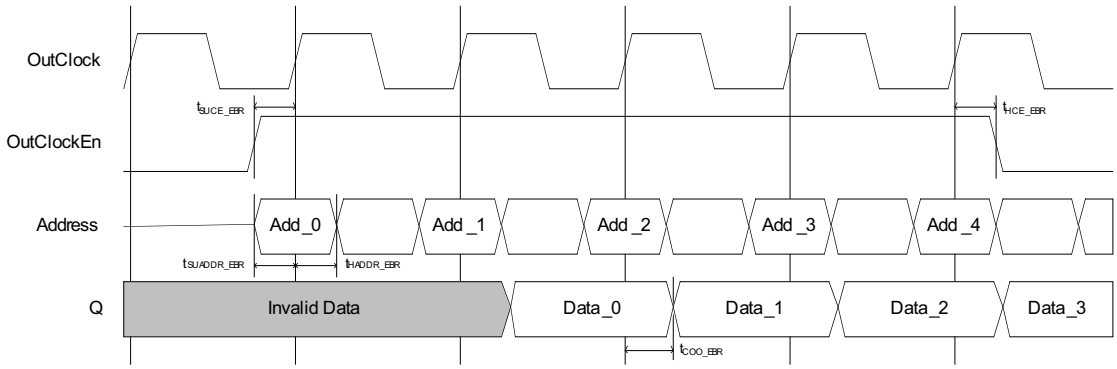


Figure 24. ROM Timing Waveform – with Output Registers



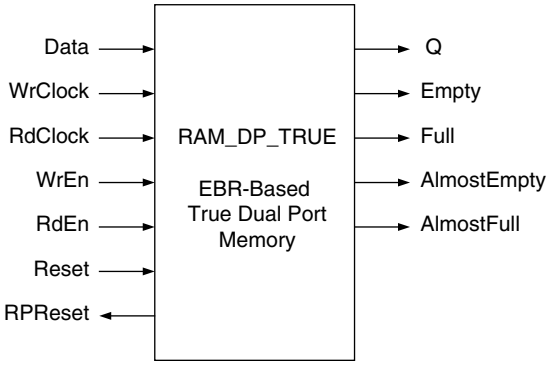
First In First Out (FIFO_DC)

EBR blocks in LatticeSC devices can be configured as First In First Out Memory – FIFO_DC. FIFO_DC (or Dual Clock FIFO) has separate clocks for the read and write ports. IPexpress allows users to generate Verilog or VHDL along with an EDIF netlist for the memory size, as per design requirements.

The LatticeSC device supports either EBR or PFU based FIFOs. They both share the same timing characteristics, the only difference is that PFU-based FIFOs should only be used for very small sizes (smaller than one EBR block). EBR-based FIFOs are much more efficient as sizes increase. It should also be noted that large PFU-based FIFOs will not only be very large relative to EBR-based FIFOs, but they will also have worse timing since they are distributed across a portion of the FPGA fabric. In the following discussion, we will focus on the EBR-based FIFO as this is the more common application.

IPexpress generates the FIFO_DC memory module shown in Figure 25.

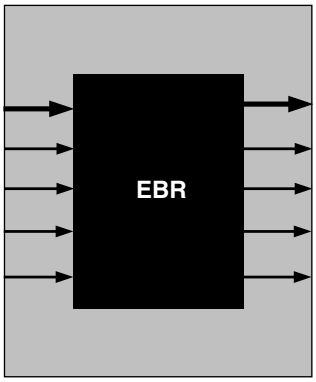
Figure 25. FIFO_DC Module Generated by IPexpress



The generated module makes use of the LatticeSC EBR blocks or primitives and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For memory sizes smaller than one EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

The basic FIFO_DC primitive for LatticeSC devices is shown in Figure 26.

Figure 26. FIFO_DC Primitive for LatticeSC



The following flags are generated for FIFO control: Full and Almost Full (to limit excess write cycles to memory), and Empty and Almost Empty (to limit excess read cycles). The range of program values for these flags is in the Attributes section. It should be noted that although assertion of Empty and Almost Empty flags is immediate, deassertion has a delay of a couple clock cycles. This delay is inherently necessary in a switch between two clock domains.

FIFO resets are supported, including FIFO reset (RST) and re-transmit reset (RPRESET). The retransmit pin is the read port reset input. Note that when the FIFO comes out of reset, both the empty and almost empty flags (EF and AEF) are initially set to high.

Each EBR block consists of 18,432 bits of RAM. Table 11 shows the FIFO_DC configurations that are supported.

For PFU-based FIFOs, the only hard restriction on size is the size of the device, but it is impractical to have a PFU-based FIFO larger than the size of one EBR block.

Table 11. EBR based FIFO_DC Sizes

FIFO Size	Input Data	Output Data
16K x 1	DI	DO
8K x 2	DI[1:0]	DO[1:0]
4K x 4	DI[3:0]	DO[3:0]
2K x 9	DI[8:0]	DO[8:0]
1K x 18	DI[17:0]	DO[17:0]
512 x 36	DI[35:0]	DO[35:0]

The Chip Select (CS) port in the EBR primitive is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus and can cascade eight memories easily. If the memory size specified requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

The various ports and their definitions for the FIFO are listed in Table 12. The table lists the corresponding ports for the module generated by IPexpress and for the FIFO_DC primitive.

Table 12. EBR-based FIFO and FIFO_DC Memory Port Definitions

Port Name in Generated Module	Description	Active State
CLKR	Read Port Clock	Rising Clock Edge
CLKW	Write Port Clock	Rising Clock Edge
WE	Write Enable	Active High
RE	Read Enable	Active High
RPRESET	Read Pointer Reset	Active High
RST	Reset	Active High
DI	Data Input	—
DO	Data Output	—
FF	Full Flag	Active High
AF	Almost Full Flag	Active High
EF	Empty Flag	Active High
AE	Almost Empty	Active High

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

Programmable Flag Values

The FIFO flags are programmable. The programmable ranges for the four FIFO flags for are specified in Table 13.

Table 13. FIFO Flag Settings

FIFO Attribute Name	Description	Programming Range	Program Bits
FF	Full flag setting	2N - 1	16
AFF	Almost full setting	1 to (FF-1)	16
AEF	Almost empty setting	1 to (FF-1)	16
EF	Empty setting	0	5

The only restriction is that the values must be in a particular order (Empty=0, Almost Empty next, followed by Almost Full and Full, respectively). The value of the Almost Empty Flag cannot be set to '0' which is the value of Empty Flag. If they are equal, a warning is generated and the value of Empty is used in place of Almost Empty.

Similarly, the Almost Full Flag cannot be equal to the Full Flag. When coming out of reset, the active high flags Empty and Almost Empty are set to high, since they are true.

The user should specify the offset value of the address at which the Almost Full Flag will go true. For example, if the Almost Full Flag is required to go true at address location 500 for a FIFO of depth 512, then user should specify the value 500 in IPexpress.

FIFO Reset
A FIFO reset will clear the contents of the FIFO by resetting the read and write pointers. It will also put the FIFO flags in the initial reset state.

Read Pointer Reset
The read pointer reset indicates retransmit of data in the FIFO and is commonly used in “packetized” communications. In this application, it is necessary to keep careful track of when a packet is written into or read from the FIFO.

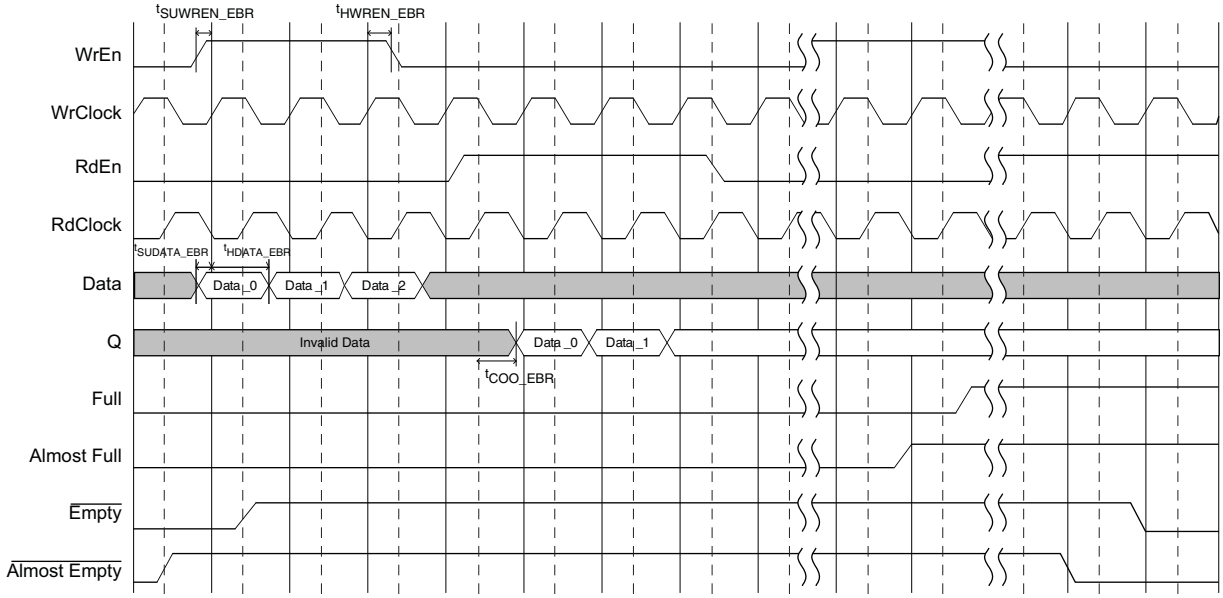
When the read pointer reset is enabled, the FIFO read address will reset to location "0" in the FIFO. This will allow a packet of data to be read again from the FIFO if four conditions are met:

1. The size of the packet of data is less than the number of words in the FIFO.
2. No writes to the FIFO occur beyond the current packet that may be re-transmitted.
3. When a packet is to be re-transmitted, the RPRESET signal is pulsed for a clock cycle before reading. This should be done following the reset rules of the FIFO.
4. When no more re-transmits of the data are required and the last word of the FIFO has been read, the FIFO must be reset with the RST input before the next packet of data is written.

Register Mode
There are two modes for registering and pipelining the read and write cycles of the memory. At the minimum mode, a single set of input registers allows synchronous write cycles into the memory array, with the other register banks bypassed. The additional mode includes using the output registers.

Figure 27 shows the timing waveforms for the FIFO_DC in the LatticeSC device.

Figure 27. FIFO_DC Timing Diagram for LatticeSC

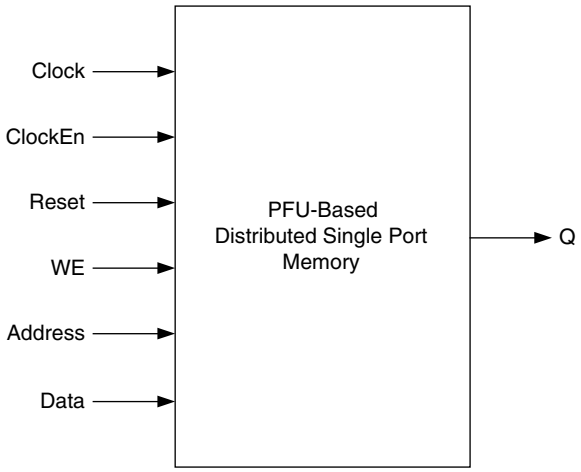


Distributed Single Port RAM (Distributed_SPRAM) – PFU Based

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

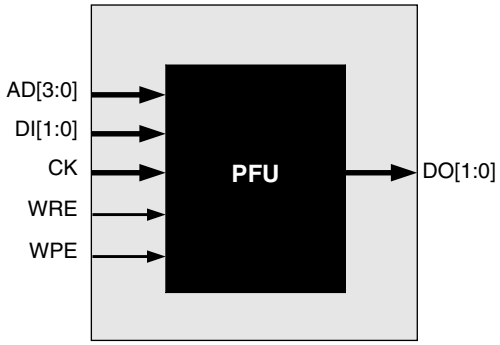
Figure 28 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 28. Distributed Single Port RAM Module generated by IPexpress



The generated module makes use of the 4-input LUT available in the PFU. The Clock and reset are generated by utilizing the resources available in the PFU. The basic Distributed Single Port RAM primitive for the LatticeSC devices is shown in Figure 29.

Figure 29. Distributed Single Port RAM for LatticeSC



The ports Read Clock (RdClock) and Clock Enable (ClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants to enable the output registers in the IPexpress configuration.

The various ports and their definitions are listed in Table 14. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 14. PFU-based Distributed Single Port RAM Port Definitions

Port Name in Generated Module	Port Name in the PFU Block Primitive	Description	Active State
Clock	CK	Clock	Rising Clock Edge
ClockEn	-	Clock Enable	Active High
Reset	-	Reset	Active High
	WRE	Write Read Enable	Write when High
Read when Low		WPE	Write Port Enable
—	WE	—	Write Enable
Write when High	Read when Low	Address	AD[3:0]
Address	—	Data	DI[1:0]
Data In	—	Q	DO[1:0]
Data Out	—		

Users have the option to enable the output registers for Distributed Single Port RAM (Distributed_SPRAM). Figures 30 and 31 show the internal timing waveforms for the Distributed Single Port RAM (Distributed_SPRAM) with these options.

Figure 30. PFU Based Distributed Single Port RAM Timing Waveform – without Output Registers

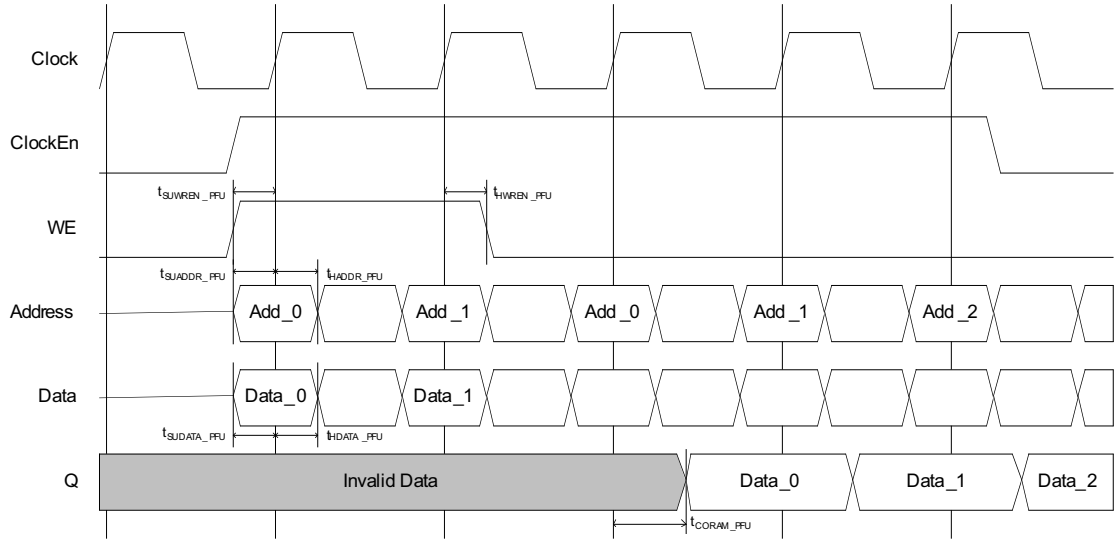
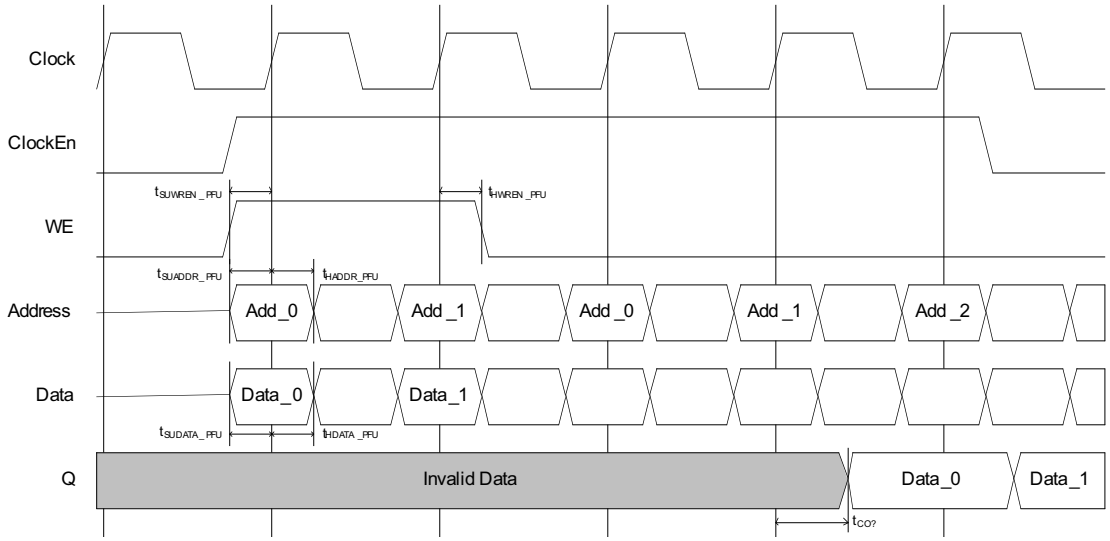


Figure 31. PFU Based Distributed Single Port RAM Timing Waveform – with Output Registers

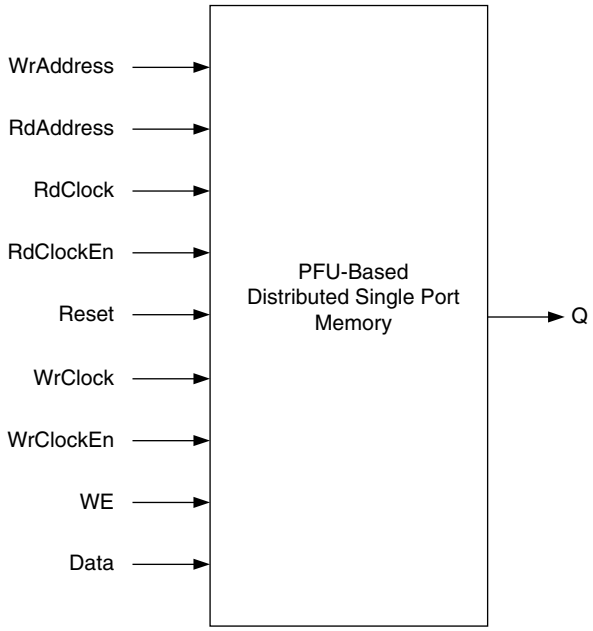


Distributed Dual Port RAM (Distributed_DPRAM) – PFU Based

PFU-based Distributed Dual Port RAM is created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

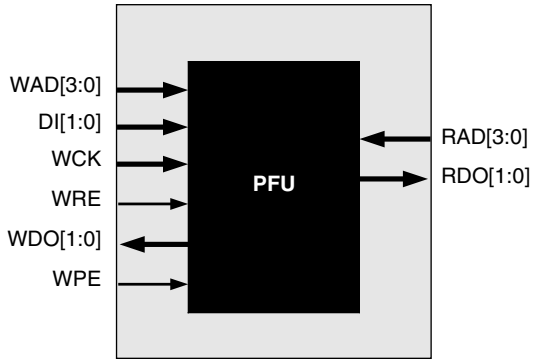
Figure 32 shows the Distributed Single Port RAM module generated by IPexpress.

Figure 32. Distributed Dual Port RAM Module Generated by IPexpress



The generated module makes use of a 4-input LUT available in the PFU. The clock and reset are generated by utilizing the resources available in the PFU. The basic Distributed Dual Port RAM primitive for LatticeSC devices is shown in Figure 33.

Figure 33. PFU-based Distributed Dual Port RAM for LatticeSC



The ports Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants the to enable the output registers in the IPexpress configuration.

The various ports and their definitions are included in Table 15. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 15. PFU-based Distributed Dual-Port RAM Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
WrAddress	WAD[23:0]	Write Address	—
RdAddress	RAD[3:0]	Read Address	—
RdClock	—	Read Clock	Rising Clock Edge
RdClockEn	—	Read Clock Enable	Active High
WrClock	WCK	Write Clock	Rising Clock Edge
WrClockEn	—	Write Clock Enable	Active High
WRE	WRE	Write Read Enable	Write when High
Read when Low	WPE	WPE	Write Port Enable
—	Data	DI[1:0]	Data Input
—	Q	RDO[1:0]	Data Out

Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed_DPRAM). Figures 34 and 35 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed_DPRAM) with these options.

Figure 34. PFU Based Distributed Dual Port RAM Timing Waveform – without Output Registers

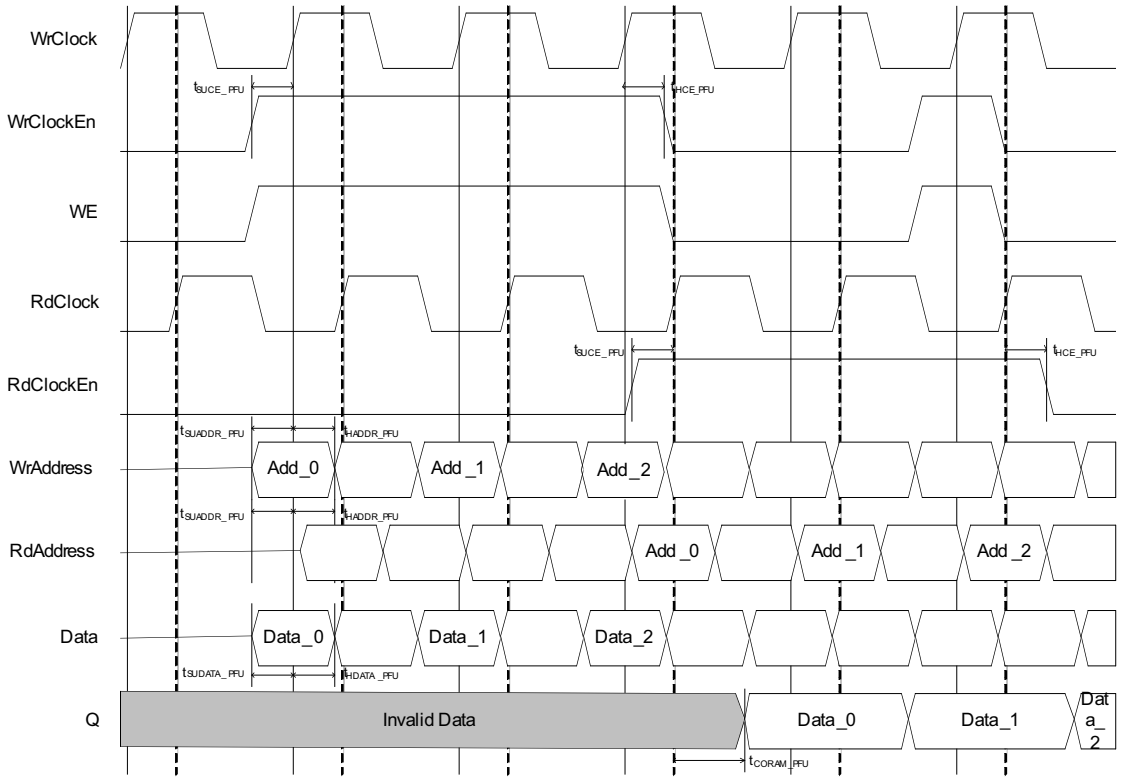
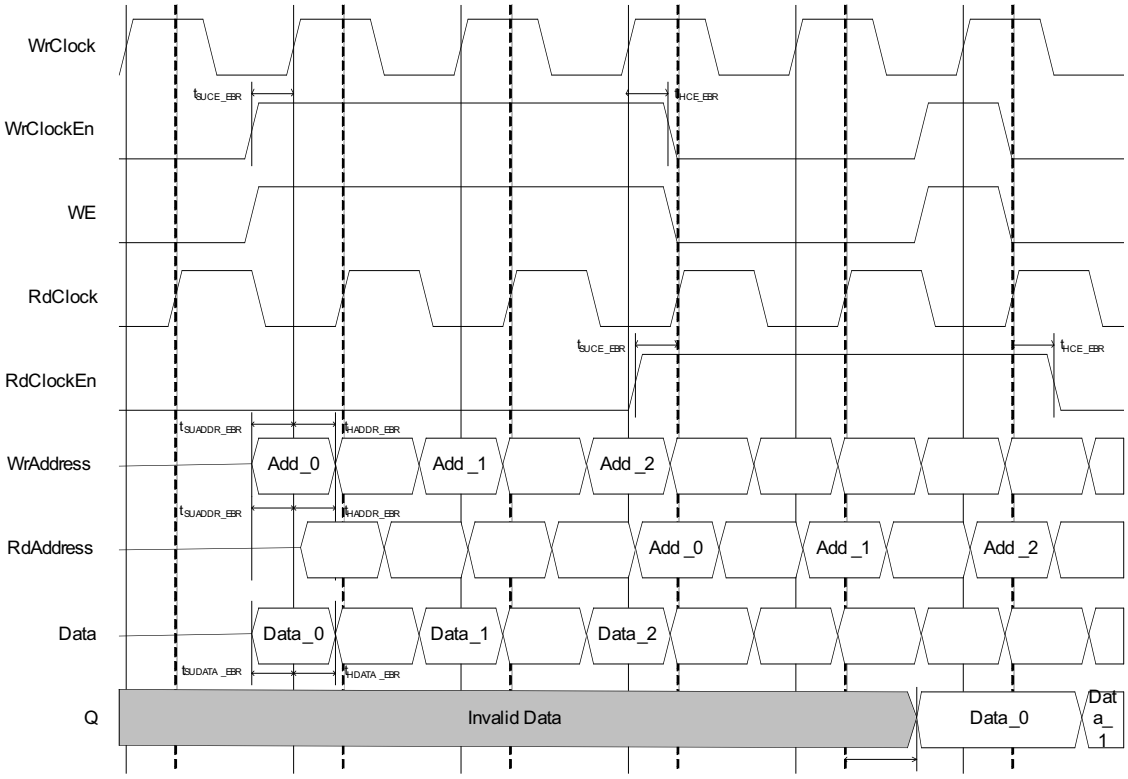


Figure 35. PFU Based Distributed Dual Port RAM Timing Waveform – with Output Registers

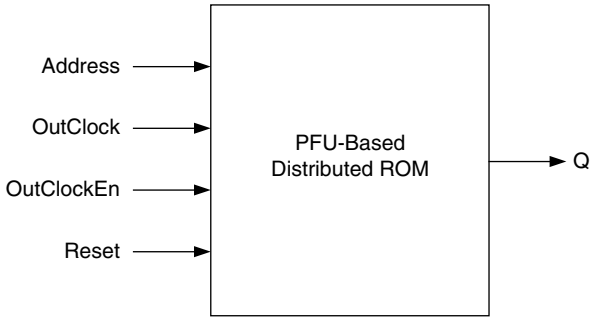


Distributed ROM (Distributed_ROM) – PFU Based

PFU-based Distributed ROM is created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

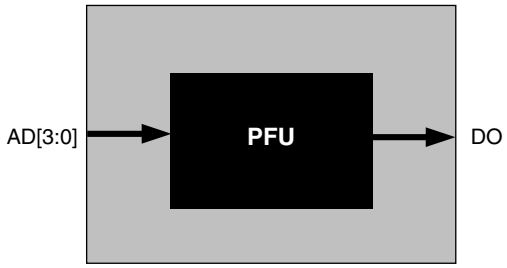
Figure 36 shows the Distributed ROM module as generated by IPexpress.

Figure 36. Distributed ROM Generated by IPexpress



The generated module makes use of 4-input LUT available in the PFU. The basic Distributed ROM primitive for the LatticeSC device is shown in Figure 37.

Figure 37. PFU-based Distributed ROM LatticeSC



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

The various ports and their definitions are included in Table 16. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 16. PFU-based Distributed ROM Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
Address	AD[3:0]	Address	—
OutClock	—	Out Clock	Rising Clock Edge
OutClockEn	—	Out Clock Enable	Active High
Reset	—	Reset	Active High
Q	DO	Data Out	—

Users have the option of enabling the output registers for Distributed ROM (Distributed_ROM). Figures 38 and 39 show the internal timing waveforms for the Distributed ROM with these options.

Figure 38. PFU Based ROM Timing Waveform – without Output Registers

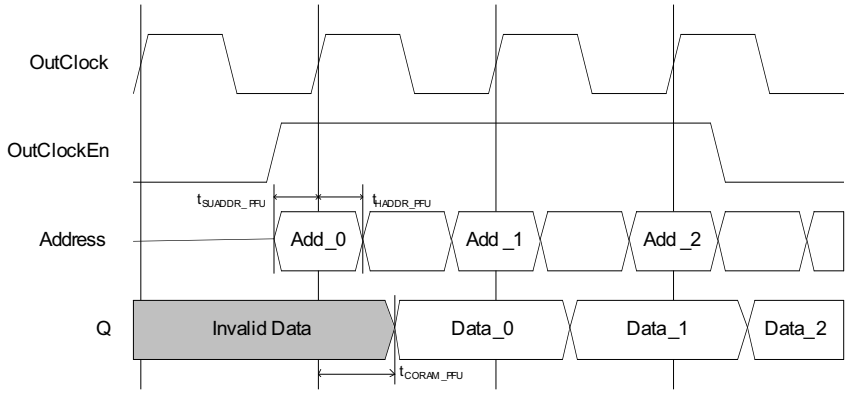
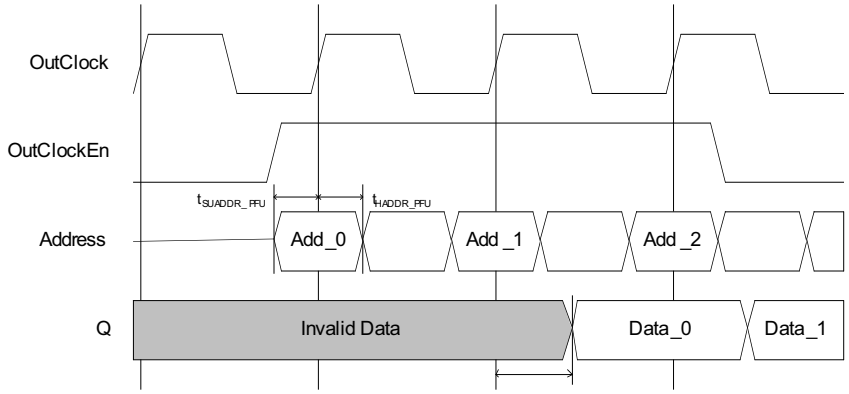


Figure 39. PFU Based ROM Timing Waveform – with Output Registers



Initializing Memory

Memories can be pre-initialized with the memory array. Users fill in the desired values during configuration and memory can be used in the read mode if desired. Each bit in the memory array can have one of two values: 0, 1.

Initialization File Format

The initialization file is an ASCII file, which can be created and edited using any ASCII editor. IPexpress supports three types of Memory File formats:

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

The file name for the memory initialization file is *.mem (<file_name>.mem). Each row depicts the value to be stored in a particular memory location and 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 Initialization File is primarily used for configuring the ROMs. RAMs can also use this Initialization File also to preload the memory contents.

Binary File

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

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

Memory Size 8x16
A001
0B03
1004
CE06
0007
040A
0017
02A4

Addressed Hex
Addressed Hex consists of lines of address and data. Each line starts with an address, followed by a colon, and any number of data. The format of memfile is address: data data data data ... where address and data are hexadecimal numbers.

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

The first line puts 03 at address A0, F3 at address A1, 3E at address A2,and 4F at address A3. The second line puts 3B at address B2 and 9F at address B3.

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

EBR Wake-up Sequence

For EBR designs, GSR (Global Set/Reset) must be released before GOE (Global Output Enable) during wake-up. See the *LatticeSC sysCONFIG Usage Guide* (TN1080) for recommended signal wake-up sequences allowed for EBR designs.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
February 2006	01.0	Initial release.
October 2006	01.1	Added dual port memory access notes in Appendix A.
February 2007	01.2	Changed references to “Module Manager” to “IPexpress” throughout.
		Updated Utilizing IPexpress section.
		Updated Read Pointer Reset section.
April 2007	01.3	Updated First In First Out (FIFO_DC) section.
July 2007	01.4	Updated First In First Out (FIFO_DC) section - added information about delay associated with deassertion of Empty and Almost Empty flags.
March 2008	01.5	Updated text and figures in the IPexpress Flow section. Updated WRITEMODE definition in Appendix A.
June 2008	01.6	Removed references to read-before-write (not supported).
		Added section on EBR Asynchronous Reset.
July 2008	01.7	Added new section for EBR wake-up sequence.
		Added definition in Appendix A for GOE.
November 2008	01.8	Added WE port to EBR RAM_DP module.

Appendix A. Attribute Definitions

DATA_WIDTH

Data Width is associated with the RAM and FIFO elements. The DATA_WIDTH attribute defines the number of bits in each word. It takes the values as defined in the RAM size tables in each memory module.

REGMODE

The REGMODE or Register mode attribute is used to enable pipelining in the memory. This attribute is associated with the RAM and FIFO elements. The REGMODE attribute takes the NOREG or OUTREG mode parameter that disables and enables the output pipeline registers.

RESETMODE

The RESETMODE attribute allows users to select the mode of reset in the memory. This attribute is associated with the block RAM elements. RESETMODE takes two parameters: SYNC and ASYNC. SYNC means that the memory reset is synchronized with the clock. ASYNC means that the memory reset is asynchronous to clock.

CSDECODE

CSDECODE or the Chip Select Decode attributes are associated to block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade 8 memories easily. CSDECODE takes the following parameters: “000”, “001”, “010”, “011”, “100”, “101”, “110”, and “111”. CSDECODE values determine the decoding value of CS[2:0]. CSDECODE_W is chip select decode for write and CSDECODE_R is chip select decode for read for Pseudo Dual Port RAM. CSDECODE_A and CSDECODE_B are used for true dual port RAM elements and refer to the A and B ports.

WRITEMODE

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

In NORMAL mode, the output data is invalid during the write operation.

In WRITETHROUGH mode, the output data is updated with the input data during the write cycle.

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write-address, during a write cycle.

WRITEMODE_A and WRITEMODE_B are used for dual port RAM elements and refer to the A and B ports in the case of True Dual Port RAM.

For all modes of the Dual Port modules, simultaneous read access from one port and write access from the other port to the same memory address is not recommended. The read data may be unknown in this situation. Also, simultaneous write access to the same address from both ports is not recommended. When this occurs, the data stored in the address becomes undetermined when one port tries to write a 'H' and the other tries to write a 'L'.

It is recommended that users control logic to identify this situation if it occurs and then either:

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

GSR

GSR, or the Global Set/Reset attribute, enables or disables the global set/reset for RAM element.

GOE

The GOE or Global Output Enable control, after the standard wake-up sequence has completed, determines when the outputs are turned over to User Logic control.