

## Introduction

The Xilinx LogiCORE™ IP Block Memory Generator core is an advanced memory constructor that generates area and performance-optimized memories using embedded block RAM resources in Xilinx FPGAs. Available through the CORE Generator™ software, users can quickly create optimized memories to leverage the performance and features of block RAMs in Xilinx FPGAs.

## Features

- Generates Single-port RAM, Simple Dual-port RAM, True Dual-port RAM, Single-port ROM, and Dual-port ROM
- Performance up to 450 MHz
- Supports data widths from 1 to 1152 bits
- Supports memory depths from 2 to 9M words (limited only by memory resources on selected part)
- Supports configurable port aspect ratios for dual-port configurations
- Supports read-to-write aspect ratios in Virtex®-4 and Virtex-5 FPGAs
- Optimized algorithm for minimum block RAM resource utilization
- Algorithm for low power utilization
- Configurable memory initialization
- Supports individual write enable per byte in Virtex-5, Virtex-4, and Spartan®-3A/3A DSP FPGA devices with or without parity
- Optimized VHDL and Verilog behavioral models for fast simulation times; structural simulation models for precise simulation of memory behaviors
- Selectable operating mode per port: WRITE\_FIRST, READ\_FIRST, or NO\_CHANGE
- Supports built-in Hamming Error Correction Capability (ECC)
- Supports pipelining of DOUT bus for improved performance in specific configurations

LogiCORE IP Facts	
<b>Core Specifics</b>	
Supported Device Family	Virtex-5, Virtex-4, Virtex-II Pro, Virtex-II, Spartan-3E, Spartan-3A/3AN/3A DSP, Spartan-3
Package	All
Speed Grade	All
Performance	Varied, based on core parameters
<b>Core Resources</b>	
Block RAM	Varied, based on core parameters
DCM	None
BUFG	None
IOBs/RocketIO™ Transceivers	None
PPC	None
IOB-FF/TBUFs	None
<b>Provided with Core</b>	
Documentation	Product Specification Migration Guide <sup>1</sup>
Design File Formats	NGC netlist
<b>Design Tool Requirements</b>	
Xilinx Implementation Tools	ISE® 10.1
Simulation Models	VHDL Structural Verilog Structural VHDL Behavioral <sup>2</sup> Verilog Behavioral <sup>2</sup>
Synthesis	XST
<b>Support</b>	
Provided by Xilinx, Inc. @ <a href="http://www.xilinx.com">www.xilinx.com</a>	

1. The Migration Guide provides instructions for converting legacy designs containing LogiCORE IP v6.x Single or Dual Port Block Memory instances to LogiCORE IP Block Memory Generator blocks.
2. Behavioral models do not precisely model collision behavior. See "Simulation Models" on page 23 for details.

## Overview

The Block Memory Generator core uses embedded Block Memory primitives in Xilinx FPGAs to extend the functionality and capability of a single primitive to memories of arbitrary widths and depths. Sophisticated algorithms within the Block Memory Generator core produce optimized solutions to provide convenient access to memories for a wide range of configurations.

The Block Memory Generator has two fully independent ports that access a shared memory space. Both A and B ports have a write and a read interface. In Virtex-5 and Virtex-4 FPGA architectures, all four interfaces can be uniquely configured, each with a different data width. When not using all four interfaces, the user can select a simplified memory configuration (for example, a Single-Port Memory or Simple Dual-Port Memory), allowing the core to more efficiently use available resources.

The Block Memory Generator is not completely backward-compatible with the Single-Port Block Memory and Dual-Port Block Memory cores; for information about the differences, see "[Compatibility with Older Memory Cores](#)" on page 40.

## Applications

The Block Memory Generator core is used to create customized memories to suit any application. Typical applications include:

- **Single-port RAM:** Processor scratch RAM, look-up tables
- **Simple Dual-port RAM:** Content addressable memories, FIFOs
- **True Dual-port RAM:** Multi-processor storage
- **Single-port ROM:** Program code storage, initialization ROM
- **Dual-port ROM:** Single ROM shared between two processors/systems

## Feature Summary

### Memory Types

The Block Memory Generator core uses embedded block RAM to generate five types of memories:

- Single-port RAM
- Simple Dual-port RAM
- True Dual-port RAM
- Single-port ROM
- Dual-port ROM

For dual-port memories, each port operates independently. Operating mode, clock frequency, optional output registers, and optional pins are selectable per port. For Simple Dual-port RAM, the operating modes are not selectable; they are fixed as READ\_FIRST. See "[Collision Behavior](#)" on page 15 for additional information.

### Selectable Memory Algorithm

The core concatenates block RAM primitives according to one of the following algorithms:

- **Minimum Area Algorithm:** The memory is generated using the minimum number of block RAM primitives. Both data and parity bits are utilized.
- **Low Power Algorithm:** The memory is generated such that the minimum number of block RAM primitives are enabled during a read or write operation.

- **Fixed Primitive Algorithm:** The memory is generated using only one type of block RAM primitive. For a complete list of primitives available for each device family, see the data sheet for that family.

## Configurable Width and Depth

The Block Memory Generator can generate memory structures from 1 to 1152 bits wide, and at least two locations deep. The maximum depth of the memory is limited only by the number of block RAM primitives in the target device.

## Selectable Operating Mode per Port

The Block Memory Generator supports the following block RAM primitive operating modes: WRITE FIRST, READ FIRST, and NO CHANGE. Each port may be assigned an operating mode.

## Selectable Port Aspect Ratios

The core supports the same port aspect ratios as the block RAM primitives:

- In all supported device families, the A port width may differ from the B port width by a factor of 1, 2, 4, 8, 16, or 32.
- In Virtex-5 and Virtex-4 FPGA-based memories, the read width may differ from the write width by a factor of 1, 2, 4, 8, 16, or 32 for each port. The maximum ratio between any two of the data widths (DINA, DOUTA, DINB, and DOUTB) is 32:1.

## Optional Byte-Write Enable

In Virtex-5, Virtex-4, and Spartan-3A/3A DSP FPGA-based memories, the Block Memory Generator core provides byte-write support for memory widths of 8-bit (no parity) or 9-bit multiples (with parity).

## Optional Output Registers

The Block Memory Generator provides two optional stages of output registering to increase memory performance. The output registers can be chosen for port A and port B separately. The core supports the Virtex-5, Virtex-4, and Spartan-3A DSP embedded block RAM registers as well as registers implemented in the FPGA fabric. See ["Output Register Configurations" on page 47](#) for more information about using these registers.

## Optional Pipeline Stages

The core provides optional pipeline stages within the MUX, available only when the registers at the output of the memory core are enabled and only for specific configurations. For the available configurations, the number of pipeline stages can be 1, 2, or 3. For detailed information, see ["Optional Pipeline Stages" on page 19](#).

## Optional Enable Pin

The core provides optional port enable pins (ENA and ENB) to control the operation of the memory. When deasserted, no read, write, or reset operations are performed on the respective port. If the enable pins are not used, it is assumed that the port is always enabled.

## Optional Synchronous Set/Reset Pin

The core provides optional set/reset pins (SSRA and SSRB) pin per port that synchronously initialize the read output to a programmable value.

## Memory Initialization

The memory contents can be optionally initialized using a memory coefficient (COE) file or by using the default data option. A COE file can define the initial contents of each individual memory location, while the default data option defines the initial content of all locations.

## Built-in Hamming Error Correction Capability

Single-port RAM and Simple Dual-port RAM memory types support the Virtex-5 FPGA Hamming Error Correction Capability (ECC) of the Virtex-5 FPGA block RAM primitives. The ECC memory automatically detects single- and double-bit errors, and is able to auto-correct the single-bit errors.

## Simulation Models

The Block Memory Generator core provides behavioral and structural simulation models in VHDL and Verilog for both simple and precise modeling of memory behaviors, for example, debugging, probing the contents of the memory, and collision detection.

## Functional Description

The Block Memory Generator is used to build custom memory modules from block RAM primitives in Xilinx FPGAs. The core implements an optimal memory by arranging block RAM primitives based on user selections, automating the process of primitive instantiation and concatenation. Using the CORE Generator Graphical User Interface (GUI), users can configure the core and rapidly generate a highly optimized custom memory solution.

## Memory Type

The Block Memory Generator creates five memory types: Single-port RAM, Simple Dual-port RAM, True Dual-port RAM, Single-port ROM, and Dual-port ROM. [Figures 1 through 5](#) illustrate the signals available for each type. Optional pins are displayed in italics.

For each configuration, optimizations are made within the core to minimize the total resources used. For example, a Simple Dual-port RAM with symmetric ports can utilize the special Simple Dual-port RAM primitive in Virtex-5 devices, which can save as much as fifty percent of the block RAM resources for memories 512 words deep or fewer. The Single-port ROM allows read access to the memory space through a single port, as illustrated in [Figure 1](#).

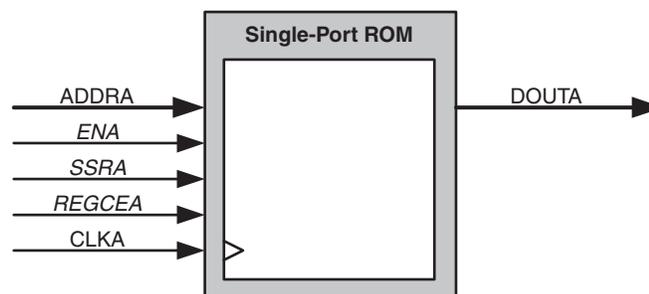


Figure 1: Single-port ROM

The Dual-port ROM allows read access to the memory space through two ports, as shown in [Figure 2](#).

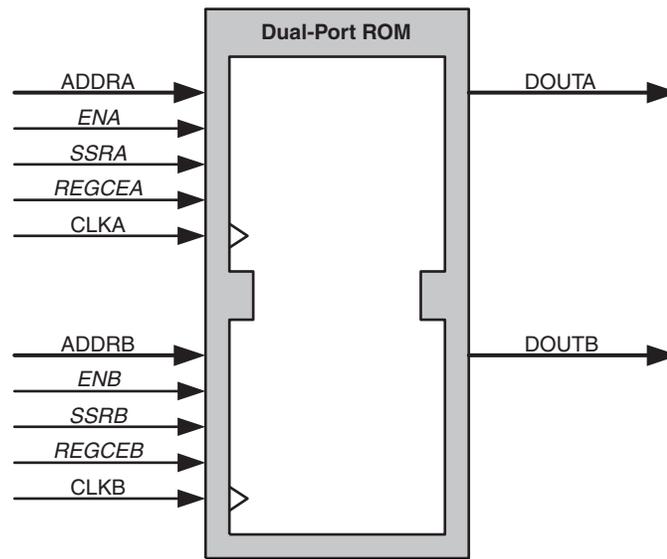


Figure 2: Dual-port ROM

The Single-port RAM allows read and write access to the memory through a single port, as shown in [Figure 3](#).

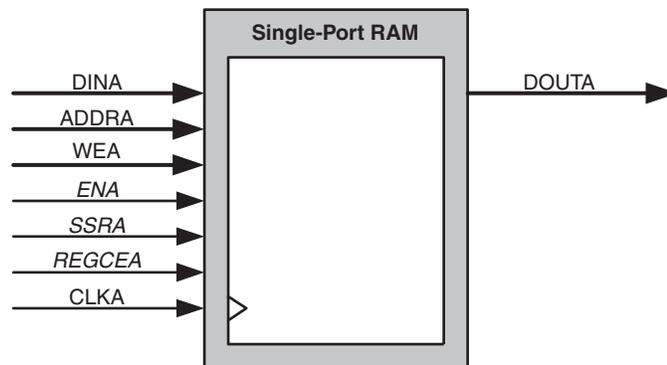


Figure 3: Single-port RAM

The Simple Dual-port RAM provides two ports, A and B, as illustrated in Figure 4. Write access to the memory is allowed via port A, and read access is allowed via port B.

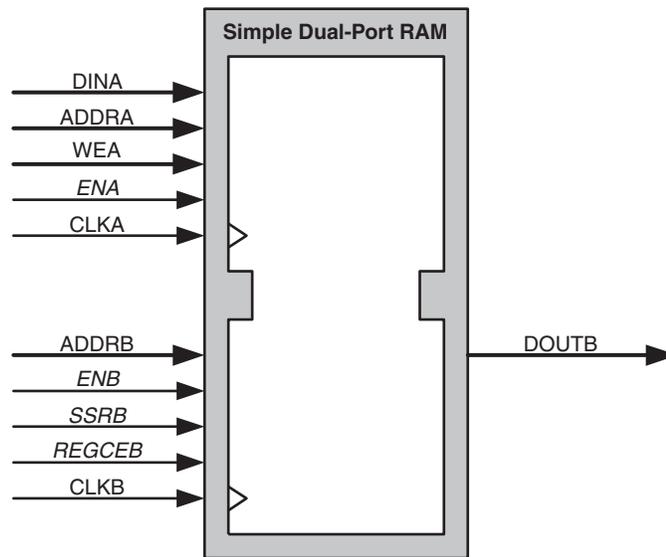


Figure 4: Simple Dual-port RAM

The True Dual-port RAM provides two ports, A and B, as illustrated in Figure 5. Read and write accesses to the memory are allowed on either port.

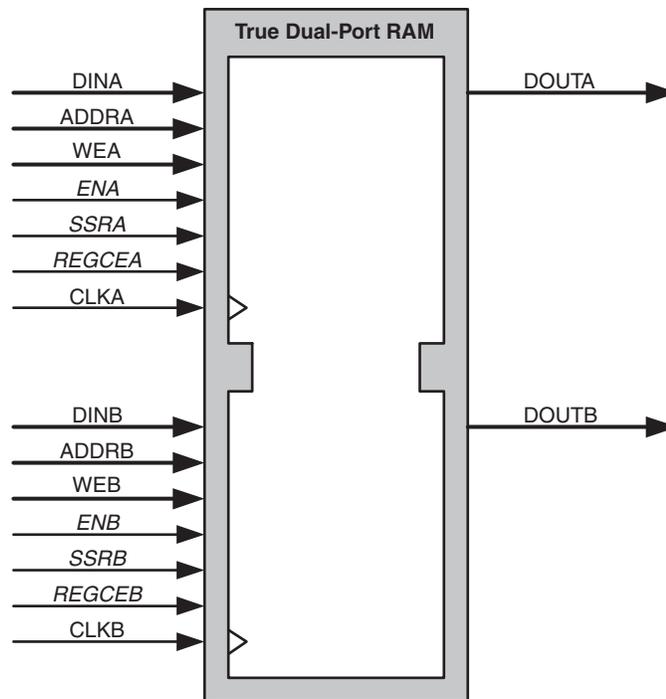


Figure 5: True Dual-port RAM

## Selectable Memory Algorithm

The Block Memory Generator core arranges block RAM primitives according to one of three algorithms: the minimum area algorithm, the low power algorithm and the fixed primitive algorithm.

### Minimum Area Algorithm

The minimum area algorithm provides a highly optimized solution, resulting in a minimum number of block RAM primitives used, while reducing output multiplexing. Figure 6 shows two examples of memories built using the minimum area algorithm.

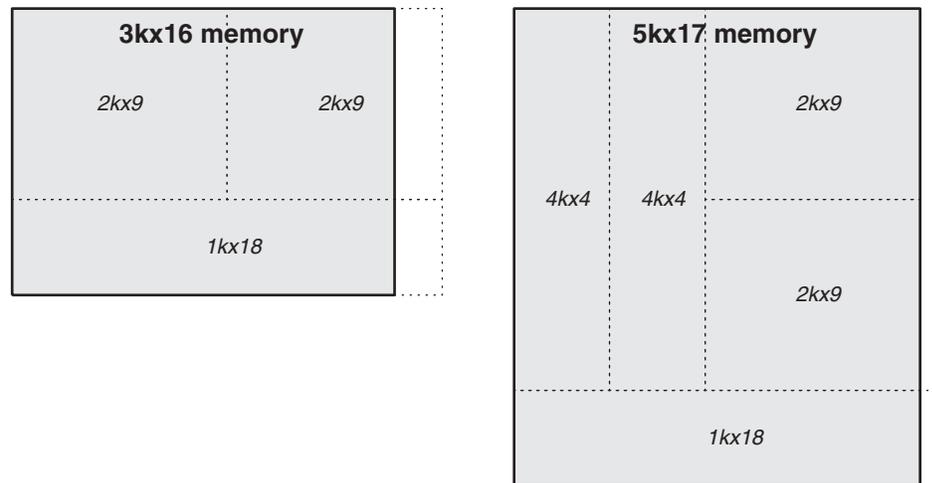


Figure 6: Examples of the Minimum Area Algorithm

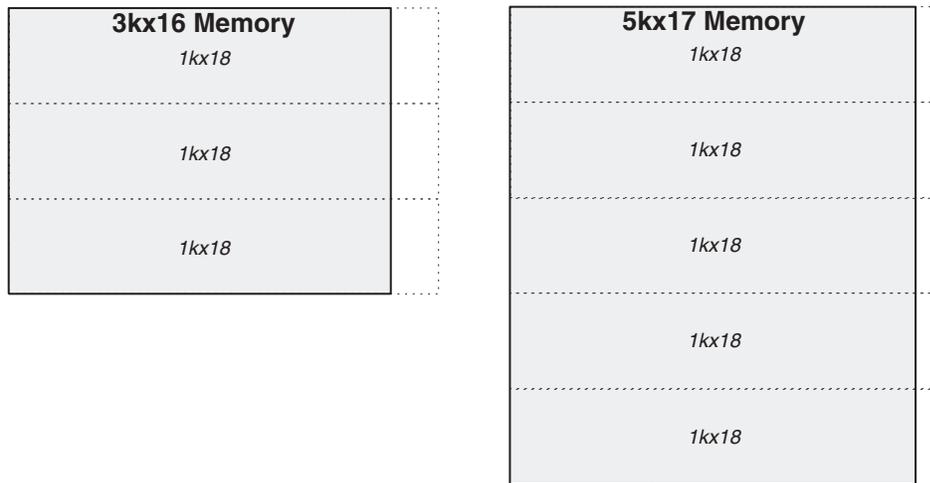
The first example, a 3kx16 memory is implemented using three block RAMs. While it may have been possible to concatenate three 1kx18 block RAMs in depth, this would require more output multiplexing. The minimum area algorithm maximizes performance in this way while maintaining minimum block RAM usage.

The second example, a 5kx17 memory, further demonstrates how the algorithm can pack block RAMs efficiently to use the fewest resources while maximizing performance by reducing output multiplexing.

### Low Power Algorithm

The low power algorithm provides a solution that minimizes the number of primitives enabled during a read or write operation. This algorithm is not optimized for area and may use more block RAMs and

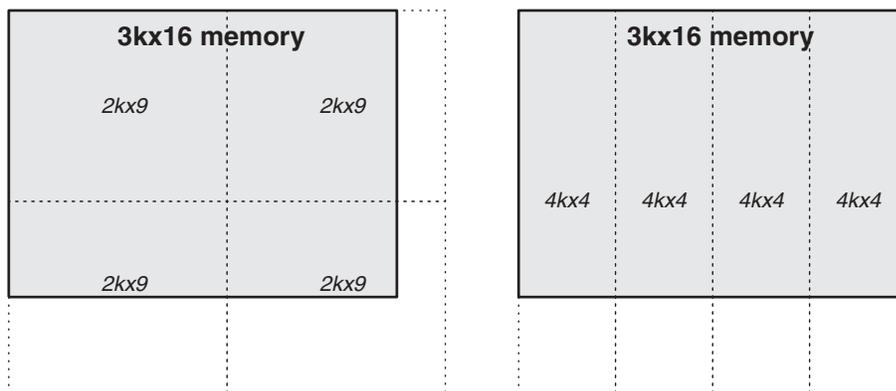
multiplexers than the minimum area algorithm. **Figure 7** shows two examples of memories built using the low power algorithm.



*Figure 7: Examples of the Low Power Algorithm*

**Fixed Primitive Algorithm**

The fixed primitive algorithm allows the user to select a single block RAM primitive type. The core will build the memory by concatenating this single primitive type in width and depth. It is useful in systems that require a fixed primitive type. **Figure 8** depicts two 3kx16 memories, one built using the 2kx9 primitive type, the other built using the 4kx4 primitive type.



*Figure 8: Examples of the Fixed Primitive Algorithm*

Note that both implementations use four block RAMs, and that some of the resources utilized extend beyond the usable memory space. It is up to the user to decide which primitive type is best for their application.

The fixed primitive algorithm provides a choice of 16kx1, 8kx2, 4kx4, 2kx9, 1kx18, 512x36 and 256x72 primitives. The primitive type selected is used to guide the construction of the total user memory space. Whenever possible, optimizations are made automatically that use deeper embedded memory struc-

tures to enhance performance. [Table 1](#) shows the primitives used to construct a memory given the specified architecture and primitive selection.

**Table 1: Memory Primitives Used Based on Architecture**

Architecture	Primitive Selection	Primitives Used
Virtex-II <sup>1</sup> , Spartan-3 <sup>2</sup>	16kx1	16kx1
Virtex-II <sup>1</sup> , Spartan-3 <sup>2</sup>	8kx2	8kx2
Virtex-II <sup>1</sup> , Spartan-3 <sup>2</sup>	4kx4	4kx4
Virtex-II <sup>1</sup> , Spartan-3 <sup>2</sup>	2kx9	2kx9
Virtex-II <sup>1</sup> , Spartan-3 <sup>2</sup>	1kx18	1kx18
Virtex-II <sup>1</sup> , Spartan-3 <sup>2</sup>	512x36	512x36
Virtex-II <sup>1</sup> , Spartan-3 <sup>2</sup>	256x72	256x72 (Single Port configurations only)
Virtex-4	16kx1	32kx1, 16kx1
Virtex-4	8kx2	8kx2
Virtex-4	4kx4	4kx4
Virtex-4	2kx9	2kx9
Virtex-4	1kx18	1kx18
Virtex-4	512x36	512x36
Virtex-4	256x72	256x72 (Single Port configurations only)
Virtex-5	16kx1	64kx1, 32kx1, 16kx1
Virtex-5	8kx2	16kx2, 8kx2
Virtex-5	4kx4	8kx4, 4kx4
Virtex-5	2kx9	4kx9, 2kx9
Virtex-5	1kx18	2kx18, 1kx18
Virtex-5	512x36	1kx36 512x36 (Single-port RAMs only)
Virtex-5	256x72	512x72 (Single and Simple Dual-port RAMs and Single Port ROMs only)

1. Virtex-II FPGA and its derivative, Virtex-II Pro FPGA.
2. Spartan-3 and its derivatives, including Spartan-3E and Spartan-3A/3A DSP devices.

When using data-width aspect ratios, the primitive type dimensions are chosen with respect to the A port write width. Note that primitive selection may limit port aspect ratios as described in "[Aspect Ratio Limitations](#)" on [page 13](#). When using the byte write feature in Virtex-5, Virtex-4, and Spartan-3A/3A DSP devices, only the 2kx9, 1kx18, and 512kx36 primitive choices are available.

### Selectable Width and Depth

The Block Memory Generator generates memories with widths from 1 to 1152 bits, and with depths of two or more words. The memory is built by concatenating block RAM primitives, and total memory size is limited only by the number of block RAMs on the target device.

Write operations to out-of-range addresses are guaranteed not to corrupt data in the memory, while read operations to out-of-range addresses will return invalid data. Note that the set/reset function should not be asserted while accessing an out-of-range address as this also results in invalid data on the output.

### Operating Mode

The operating mode for each port determines the relationship between the write and read interfaces for that port. Port A and port B can be configured independently with any one of three write modes: Write First Mode, Read First Mode, or No Change Mode. These operating modes are described in the sections that follow.

The operating modes have an effect on the relationship between the A and B ports when the A and B port addresses have a collision. For detailed information about collision behavior, see "[Collision Behavior](#)" on page 15. For more information about operating modes, see the block RAM section of the user guide specific to the device family.

- **Write First Mode:** In WRITE\_FIRST mode, the input data is simultaneously written into memory and driven on the data output, as shown in [Figure 9](#). This transparent mode offers the flexibility of using the data output bus during a write operation on the same port.

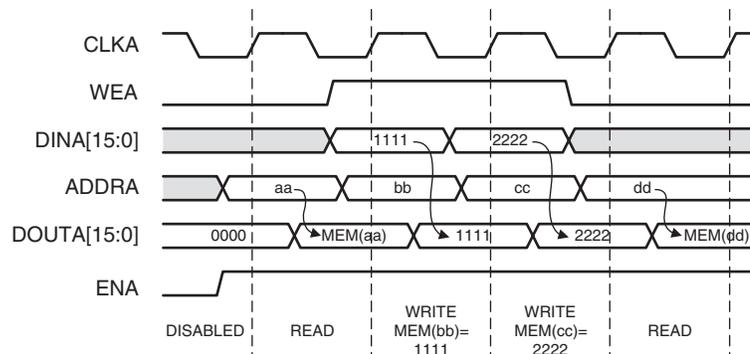


Figure 9: Write First Mode Example

**Note:** The WRITE\_FIRST operation is affected by the optional byte-write feature in Virtex-5, Virtex-4, and Spartan-3A/3A DSP devices. It is also affected by the optional read-to-write aspect ratio feature in Virtex-5 and Virtex-4 devices. For detailed information, see "[Write First Mode Considerations](#)" on page 15.

- **Read First Mode:** In READ\_FIRST mode, data previously stored at the write address appears on the data output, while the input data is being stored in memory. This read-before-write behavior is

illustrated in [Figure 10](#).

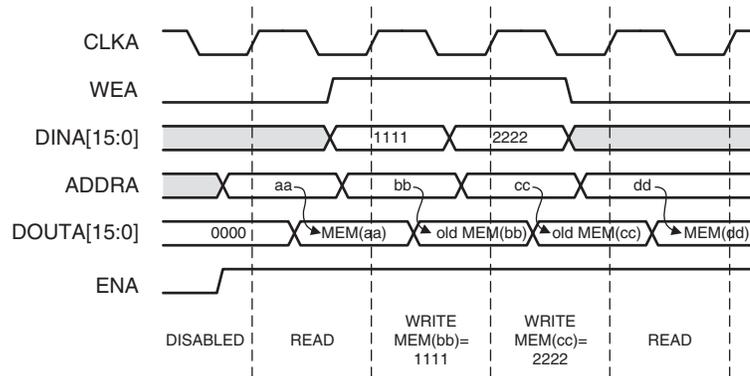


Figure 10: Read First Mode Example

- **No Change Mode:** In NO\_CHANGE mode, the output latches remain unchanged during a write operation. As shown in [Figure 11](#), the data output is still the previous read data and is unaffected by a write operation on the same port.

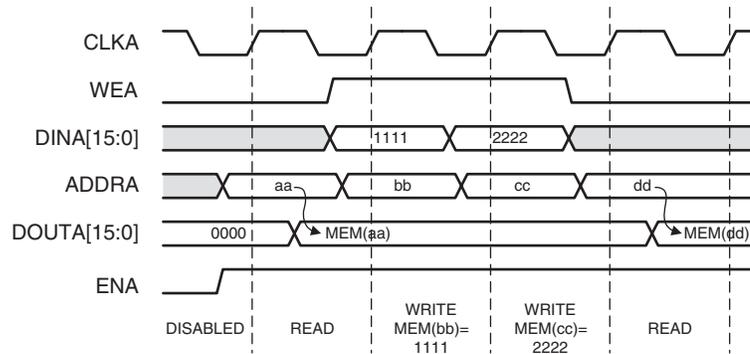


Figure 11: No Change Mode Example

### Data Width Aspect Ratios

The Block Memory Generator supports data width aspect ratios. This allows the port A data width to be different than the port B data width, as described in Port Aspect Ratios in the following section. In Virtex-5 and Virtex-4 FPGA-based memories, all four data busses (DINA, DOUTA, DINB, and DOUTB) can have different widths, as described in ["Virtex-5 and Virtex-4 Read-to-Write Aspect Ratios"](#) on page 12.

The limitations of the data width aspect ratio feature (some of which are imposed by other optional features) are described in ["Aspect Ratio Limitations"](#) on page 13. The CORE Generator GUI ensures only valid aspect ratios are selected.

### Port Aspect Ratios

The Block Memory Generator supports port aspect ratios of 1:32, 1:16, 1:8, 1:4, 1:2, 1:1, 2:1, 4:1, 8:1, 16:1, and 32:1. The port A data width can be up to 32 times larger than the port B data width, or vice versa. The smaller data words are arranged in little-endian format, as illustrated in [Figure 12](#).

### Port Aspect Ratio Example

Consider a True Dual-port RAM of 32x2048, which is the A port width and depth. From the perspective of an 8-bit B port, the depth would be 8192. The ADDR<sub>A</sub> bus is 11 bits, while the ADDR<sub>B</sub> bus is 13 bits. The data is stored little-endian, as shown in Figure 12. Note that  $A_n$  is the data word at address  $n$ , with respect to the A port.  $B_n$  is the data word at address  $n$  with respect to the B port.  $A_0$  is comprised of  $B_3$ ,  $B_2$ ,  $B_1$ , and  $B_0$ .

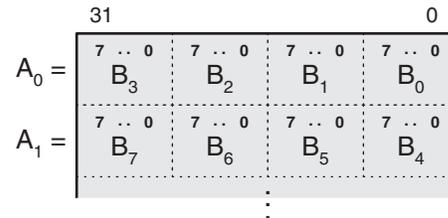


Figure 12: Port Aspect Ratio Example Memory Map

### Virtex-5 and Virtex-4 Read-to-Write Aspect Ratios

When implementing RAMs targeting Virtex-5 and Virtex-4 FPGAs, the Block Memory Generator allows read and write aspect ratios on either port. On each port A and port B, the read to write data width ratio of that port can be 1:32, 1:16, 1:8, 1:4, 1:2, 1:1, 2:1, 4:1, 8:1, 16:1, or 32:1.

Because the read and write interfaces of each port can differ, it is possible for all four data buses (DINA, DOUTA, DINB, and DOUTB) of True Dual-port RAMs to have a different width. For Single-port RAMs, DINA and DOUTA widths can be independent. The maximum ratio between any two data buses is 32:1. The widest data bus can be no larger than 1152 bits.

If the read and write data widths on a port are different, the memory depth is different with respect to read and write accesses. For example, if the read interface of port A is twice as wide as the write interface, then it is also half as deep. The ratio of the widths is *always* the inverse of the ratio of the depths. Because a single address bus is used for both the write and read interface of a port, the address bus must be large enough to address the deeper of the two depths. For the shallower interface, the least significant bits of the address bus are ignored. The data words are arranged in little-endian format, as illustrated in Figure 13.

### Virtex-5 and Virtex-4 Read-to-Write Aspect Ratio Example

Consider a True Dual-port RAM of 64x512, which is the port A write width and depth. Table 2 defines the four data-port widths and their respective depths for this example.

Table 2: Read-to-Write Aspect Ratio Example Ports

Interface	Data Width	Memory Depth
Port A Write	64	512
Port A Read	16	2048
Port B Write	256	128
Port B Read	32	1024

The ADDRA width is determined by the larger of the A port depths (2048). For this reason, ADDRA is 11 bits wide. On port A, read operations utilize the entire ADDRA bus, while write operations ignore the least significant 2 bits.

In the same way, the ADDR B width is determined by the larger of the B port depths (1024). For this reason, ADDR B is 10 bits wide. On port B, read operations utilize the entire ADDR B bus, while write operations ignore the least significant 3 bits.

The memory map in Figure 13 shows how port B write words are related to port A write words, in a little-endian arrangement. Note that  $AW_n$  is the write data word at address  $n$  with respect to port A, while  $BW_n$  is the write data word at address  $n$  with respect to port B.

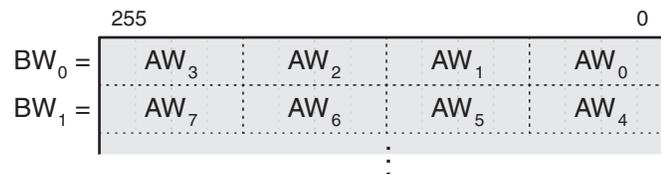


Figure 13: Read-to-Write Aspect Ratio Example Memory Map

$BW_0$  is made up of  $AW_3$ ,  $AW_2$ ,  $AW_1$ , and  $AW_0$ . In the same way,  $BR_0$  is made up of  $AR_1$  and  $AR_0$ , and  $AW_0$  is made up of  $BR_1$  and  $BR_0$ . In the example above, the largest data width ratio is port B write words (256 bits) to port A read words (16 bits); this ratio is 16:1.

### Aspect Ratio Limitations

In general, no port data width can be wider than 1152 bits, and no two data widths can have a ratio greater than 32:1. However, the following optional features further limit data width aspect ratios:

- **Byte-writes.** When using byte-writes, no two data widths can have a ratio greater than 4:1.
- **Fixed primitive algorithm.** When using the fixed primitive algorithm with an N-bit wide primitive, aspect ratios are limited to 32:N and 1:N from the port A write width. For example, using the 4kx4 primitive type, the other ports may be no more than 8 times (32:4) larger than port A write width and no less than 4 times (1:4) smaller.

### Byte-Writes

The Block Memory Generator provides byte-write support in Virtex-5, Virtex-4, and Spartan-3A/3A DSP devices. Byte-writes are available using either 8-bit or 9-bit byte sizes. When using an 8-bit byte size, no parity bits are used and the memory width is restricted to multiples of 8 bits. When using a 9-bit byte size, each byte includes a parity bit, and the memory width is restricted to multiples of 9 bits.

When byte-writes are enabled, the  $WE[A|B]$  bus is N bits wide, where N is the number of bytes in  $DIN[A|B]$ . The most significant bit in the write enable bus corresponds to the most significant byte in the input word. Bytes will be stored in memory only if the corresponding bit in the write enable bus is asserted during the write operation.

When 8-bit bytes are selected, the DIN and DOUT data buses are constructed from 8-bit bytes, with no parity. When 9-bit bytes are selected, the DIN and DOUT data buses are constructed from 9-bit bytes, with the 9th bit of each byte in the data word serving as a parity bit for that byte.

The byte-write feature may be used in conjunction with the data width aspect ratios, which may limit the choice of data widths as described in "Data Width Aspect Ratios" on page 11. However, it may not

be used with the NO\_CHANGE operating mode. This is because if a memory configuration uses multiple primitives in width, and only one primitive is being written to (using partial byte writes), then the NO\_CHANGE mode only applies to that single primitive. The NO\_CHANGE mode does not apply to the other primitives that are not being written to, so these primitives can still be read. The byte-write feature also affects the operation of WRITE\_FIRST mode, as described in "Write First Mode Considerations" on page 15.

### Byte-Write Example

Consider a Single-port RAM with a data width of 24 bits, or 3 bytes with byte size of 8 bits. The write enable bus, WEA, consists of 3 bits. Figure 14 illustrates the use of byte-writes, and shows the contents of the RAM at address 0. Assume all memory locations are initialized to 0.

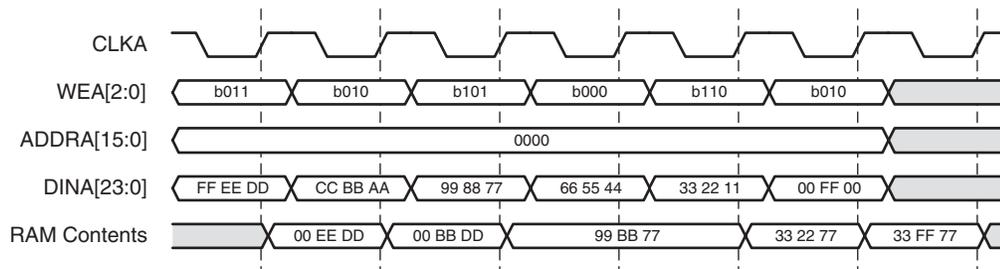


Figure 14: Byte-write Example

## Write First Mode Considerations

When performing a write operation in WRITE\_FIRST mode, the concurrent read operation shows the newly written data on the output of the core. However, when using the byte-write feature in Virtex-5, Virtex-4, and Spartan-3A/3A DSP devices or the read-to-write aspect ratio feature in Virtex-5 and Virtex-4 devices, the output of the memory cannot be guaranteed.

## Collision Behavior

The Block Memory Generator core supports Dual-port RAM implementations. Each port is equivalent and independent, yet they access the same memory space. In such an arrangement, it is possible to have data collisions. The ramifications of this behavior are described for both asynchronous and synchronous clocks below.

### Collisions and Asynchronous Clocks

Using asynchronous clocks, when one port writes data to a memory location, the other port must not read or write that location for a specified amount of time. This clock-to-clock setup time is defined in the device data sheet, along with other block RAM switching characteristics.

### Collisions and Synchronous Clocks

Synchronous clocks cause a number of special case collision scenarios, described below.

- **Synchronous Write-Write Collisions.** A write-write collision occurs if both ports attempt to write to the same location in memory. The resulting contents of the memory location are unknown. Note that write-write collisions affect memory content, as opposed to write-read collisions which only affect data output.
- **Using Byte-Writes.** When using byte-writes, memory contents are not corrupted when separate bytes are written in the same data word. RAM contents are corrupted only when both ports attempt to write the same byte. **Figure 15** illustrates this case. Assume ADDR<sub>A</sub> = ADDR<sub>B</sub> = 0.

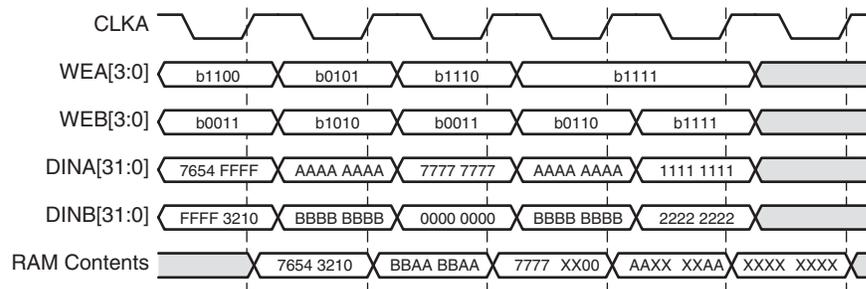


Figure 15: Write-write Collision Example

- **Synchronous Write-Read Collisions.** A synchronous write-read collision may occur if a port attempts to write a memory location and the other port reads the same location. While memory contents are not corrupted in write-read collisions, the validity of the output data depends on the write port operating mode.
  - If the write port is in READ\_FIRST mode, the other port can reliably read the old memory contents.
  - If the write port is in WRITE\_FIRST or NO\_CHANGE mode, data on the output of the read port is invalid.
  - In the case of byte-writes, only bytes which are updated will be invalid on the read port output.

Figure 16 illustrates write-read collisions and the effects of byte-writes. DOUTB is shown for when port A is in WRITE\_FIRST mode and READ\_FIRST mode. Assume ADDR<sub>A</sub> = ADDR<sub>B</sub> = 0, port B is always reading, and all memory locations are initialized to 0. The RAM contents are never corrupted in write-read collisions.

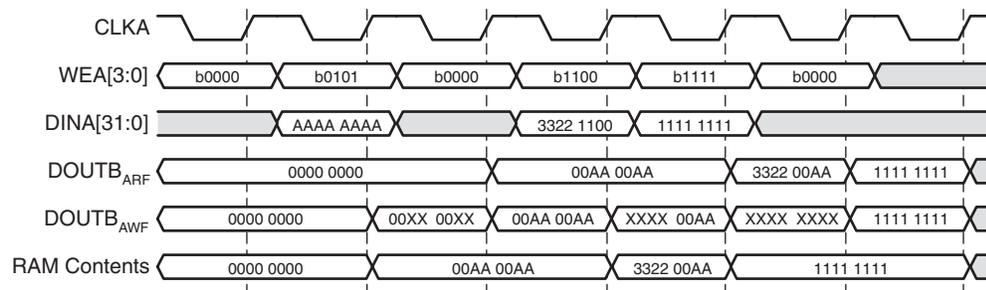


Figure 16: Write-read Collision Example

### Collisions and Simple Dual-port RAM

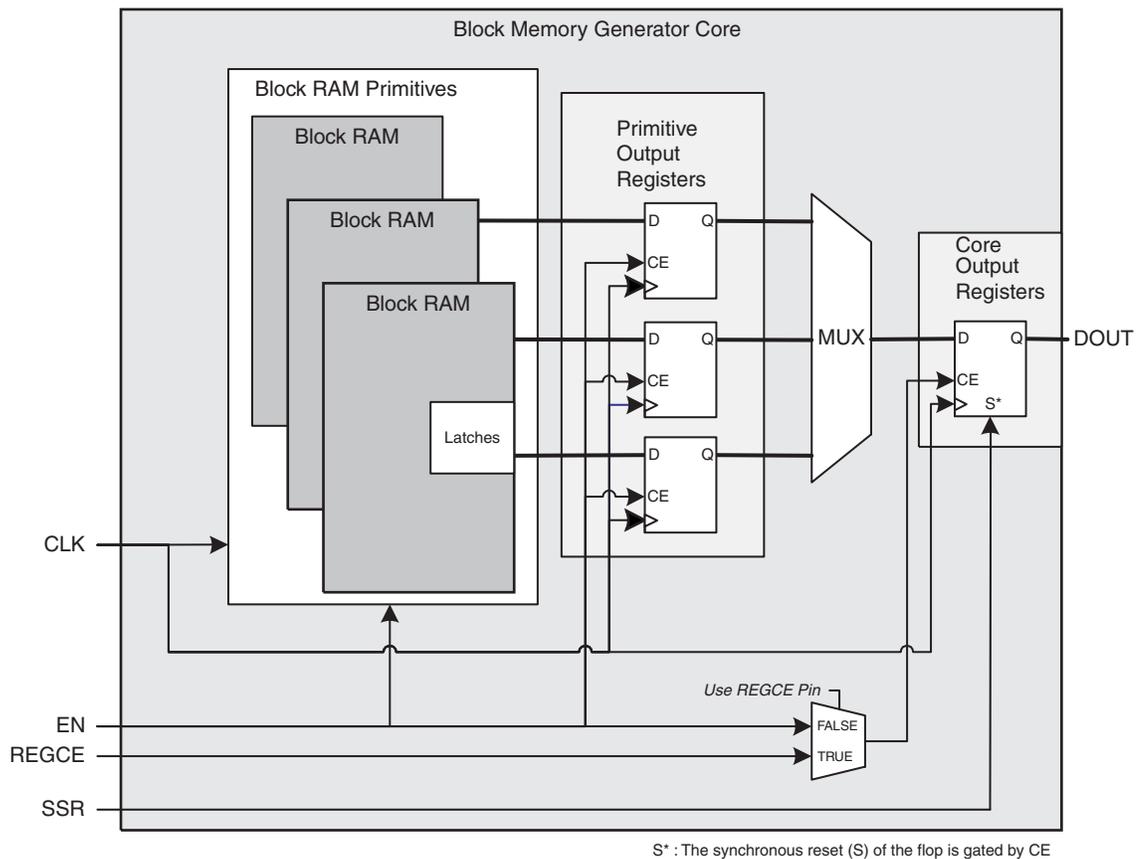
For Simple Dual-port RAM, the operating modes are not selectable, but are fixed as READ\_FIRST. The Simple Dual-port RAM is like a true dual-port RAM, where only the write interface of the A port and the read interface of B port are connected. The operating modes define the write-to-read relationship of the A or B ports, and only impact the relationship between A and B ports during an address collision.

For detailed information about this behavior, see "[Collision Behavior](#)" on page 15. During a collision, the write mode of port A can be configured such that the read operation on port B either produces data (it acts like READ\_FIRST), or produces undefined data (Xs). As a result, the core is hard-coded to produce the READ\_FIRST-like behavior when configured as a Simple Dual-port RAM.

## Optional Output Registers

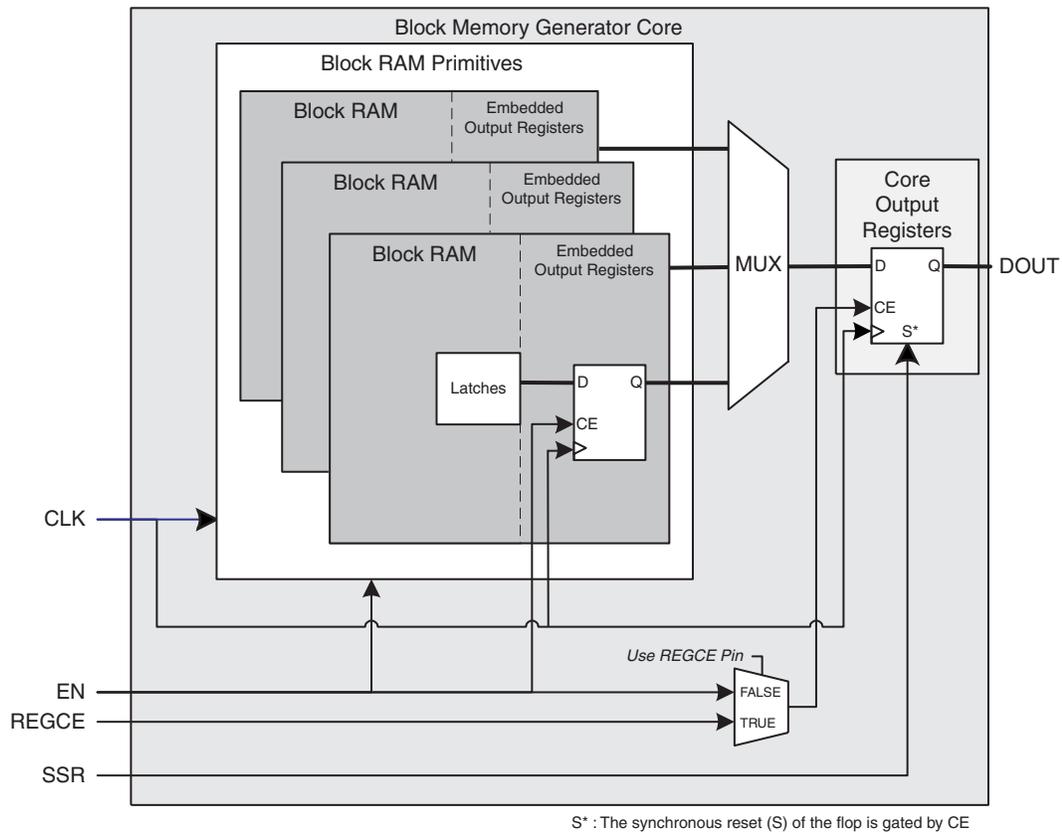
The Block Memory Generator allows optional output registers, which may improve the performance of the core. The user may choose to include register stages at two places—at the output of the block RAM primitives and at the output of the core.

Registers at the output of the block RAM primitives reduce the impact of the clock-to-out delay of the primitives. Registers at the output of the core isolate the delay through the output multiplexers, improving the clock-to-out delay of the Block Memory Generator core. Each of the two optional register stages can be chosen for port A and port B separately. Note that each optional register stage used adds an additional clock cycle of latency to the read operation. **Figure 17** shows a memory configuration with registers at the output of the memory primitives and at the output of the core for one of the ports.



**Figure 17: Virtex-II Block Memory: Register Port [A|B] Outputs of Memory Primitives and Memory Core Options Enabled**

For Virtex-5, Virtex-4, and Spartan-3A DSP FPGAs, the Register Port [A | B] Output of Memory Primitives option may be implemented using the embedded block RAM registers, requiring no further FPGA resources. All other register stages are implemented in FPGA fabric. **Figure 18** shows an example of a Virtex-4 or Virtex-5-based memory that has been configured using both output register stages for one of the ports.



**Figure 18: Virtex-5 and Virtex-4 Block Memory with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Options Enabled**

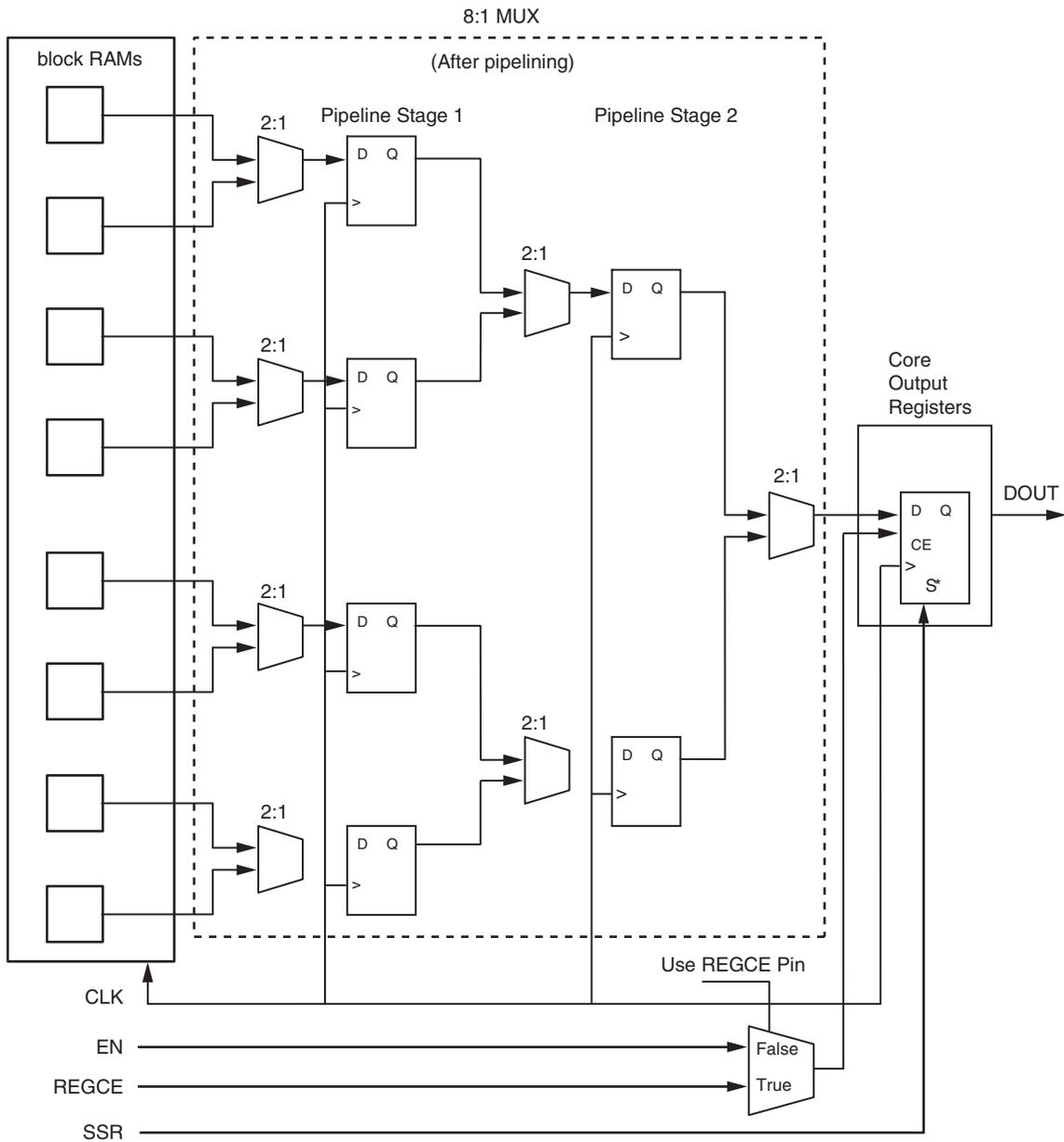
When using the Synchronous Reset Input (SSR), the behavior of the embedded output registers in the Spartan-3A DSP FPGA differs slightly from the configuration shown in **Figure 18**. By default, the Block Memory Generator builds the memory output register in the FPGA fabric to maintain functionality compatibility with Virtex-4 and Virtex-5 FPGA configurations. To force the core to use the embedded output registers in Spartan-3A DSP device, the RAMB16BWER Reset Behavior option is provided. For a complete description of the supported output options, see **"Output Register Configurations"** on [page 47](#).

### Optional Pipeline Stages

The Block Memory Generator core allows optional pipeline stages within the MUX, which may improve core performance. Users can add up to three pipeline stages within the MUX, excluding the registers at the output of the core. This optional pipeline stages option is available only when the registers at the output of the memory core are enabled and when the constructed memory has more than one primitive in depth, so that a MUX is needed on the output.

The pipeline stages are common for port A and port B and can be a value of 1, 2, or 3 if the Register Output of Memory Core option is selected in the GUI for both port A and port B. Note that each pipeline stage adds an additional clock cycle of latency to the read operation.

If the configuration has ECC, the SBITERR and DBITERR outputs are delayed to align with DOUT. Note that adding pipeline stages within the MUX improves performance only if the critical path in the design is the data through the MUX. The MUX size displayed in the GUI can be used to determine the number of pipeline stages to use within the MUX. See "Pipeline Stages within Mux" on page 31 for detailed information. [Figure 19](#) shows a memory configuration with an 8:1 MUX and two pipeline stages within the MUX. This figure explains how the 8:1 MUX is pipelined internally with two register stages.



S\*: Synchronous reset (S) of the flop is gated by CE

Figure 19: Memory Configuration with 8:1 MUX and Two Pipeline Stages within the MUX

### Optional Register Clock Enable Pins

The optional output registers are enabled by the EN signal by default. However, when the Use REGCEA/REGCEB Pin option is selected, the output register stage of the corresponding port is controlled by the REGCEA/REGCEB pins; the data output from the core can be controlled independent of the flow of data through the rest of the core. When using the REGCE pin, the last output register operates independently of the EN signal.

## Optional Synchronous Set/Reset Pins

The synchronous set/reset pins (SSRA and SSRB) control the reset operation of the last register in the output stage. For memories with no output registers, the reset pins control the memory output latches.

When SSR and REGCE are asserted on a given port, the data on the output of that port is driven to the reset value defined in the CORE Generator GUI. (The reset occurs on SSR and EN when the Use REGCE Pin option is not selected.)

- For Virtex-4 FPGAs, if the option to use the synchronous set/reset pin is selected in conjunction with memory primitive registers and without core output registers, the Virtex-4 embedded block RAM registers are not utilized for the corresponding port and are implemented in the FPGA logic instead.
- For Spartan-3A DSP FPGAs, the synchronous set/reset behavior differs when the RAMB16BWER reset behavior option is selected. However, this option saves resources by using the embedded output registers available in Spartan-3A DSP RAMB16BWER primitives. See "[Output Register Configurations](#)" on page 47 for more information.

## Memory Output Flow Control Examples

The combination of the enable (EN), reset (SSR), and register enable (REGCE) pins allow a wide range of data flows in the output stage. [Figure 20](#) and [Figure 21](#) are examples on how this can be accomplished. Keep in mind that the SSR and REGCE pins apply only to the last register stage.

[Figure 20](#) depicts how SSR can be used to control the data output to allow only intended data through. Assume that both output registers are used for port A, the port A reset value is 0xFFFF, and that EN and REGCE are always asserted. The data on the block RAM memory latch is labeled LATCH, while the output of the block RAM embedded register is labeled REG1. The output of the last register is the output of the core, DOUT.

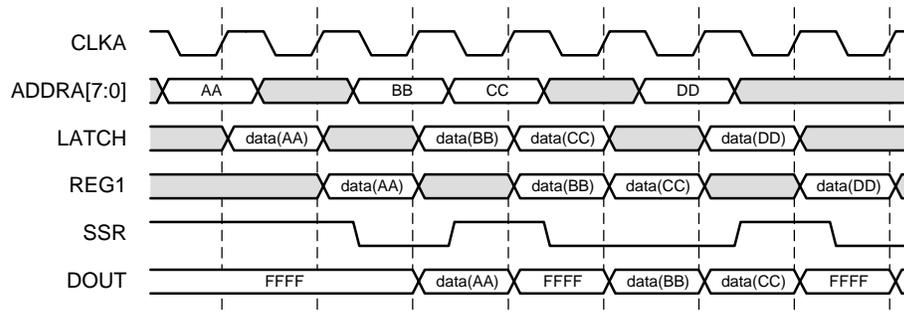


Figure 20: Flow Control Using SSR

Figure 21 depicts how REGCE can be used to latch the data output to allow only intended data through. Assume that only the memory primitive registers are used for port A, and that EN is always asserted and SSR is always deasserted. The data on the block RAM memory latch is labeled latch, while the output of the last register, the block RAM embedded register, is the core output, DOUT.

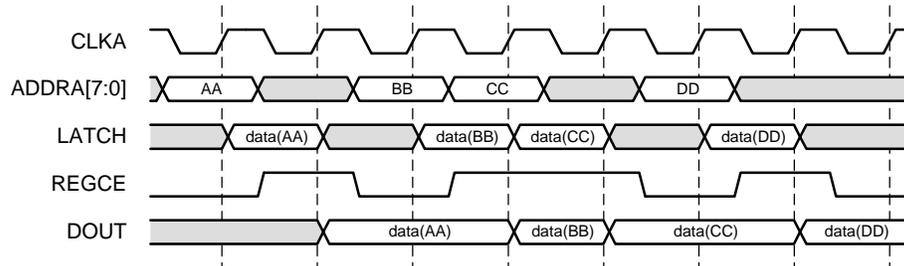


Figure 21: Flow Control Using REGCE

### Built-in Error Correction Capability

For Virtex-5 devices, the Block Memory Generator core supports built-in Hamming Error Correction Capability (ECC) for the block RAM primitives. Each write operation generates 8 protection bits for every 64 bits of data, which are stored with the data in memory. These bits are used during each read operation to correct any single bit error, or to detect (but not correct) any double bit error.

This operation is transparent to the user. Two status outputs (SBITERR and DBITERR) indicate the three possible read results: no error, single error corrected, and double error detected. For single-bit errors, the read operation does not correct the error in the memory array; it only presents corrected data on DOUT. ECC is only available when the following options are chosen:

- Virtex-5 FPGA
- Single-port RAM or Simple Dual-port RAM memory type

When using ECC, the Block Memory Generator constructs the memory from special primitives available in Virtex-5 FPGA architectures. The ECC memory block is 512x64, and is composed of two 18k block RAMs combined with dedicated ECC encoding and decoding hardware. The 512x64 primitives are used to build memory sufficient for the desired user memory space.

When using ECC, other limited core options include the following:

- Byte-write enable is not available
- All port widths must be identical
- Only Read First Operating Mode is supported
- Synchronous Reset (SSR) and the Output Reset Value options are not available
- Memory Initialization is not supported
- No Algorithm selection is available

Figure 22 illustrates a typical write and read operation for a Virtex-5 FPGA Block Memory Generator core in Simple Dual-port RAM with ECC enabled, and no additional output registers.

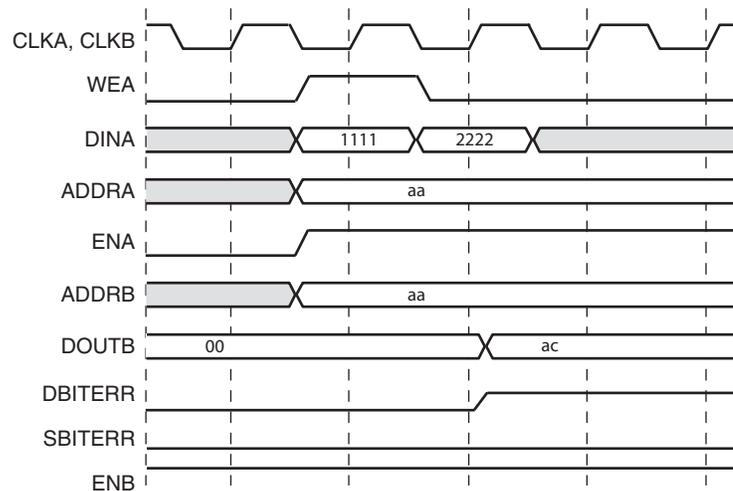


Figure 22: Read and Write Operation with ECC in Virtex-5

The Block Memory Generator core does not support the insertion of errors for correction by ECC in simulation. For this reason, the simulated functionality of ECC is identical to non-ECC behavior with the SBITERR and DBITERR outputs always equal to 0.

### Simulation Models

The Block Memory Generator core provides two types of functional simulation models:

- Behavioral Simulation Models (VHDL and Verilog)
- Structural/Unisim based Simulation Models (VHDL and Verilog)

The behavioral simulation models provide a simplified model of the core while the structural simulation models (UniSim) are an accurate modeling of the internal structure of the core. The behavioral simulation models are written purely in RTL and simulate faster than the structural simulation models and are ideal for functional debugging. Moreover, the memory is modeled in a two-dimensional array, making it easier to probe contents of the memory.

The structural simulation model uses primitive instantiations to model the behavior of the core more precisely. Use the structural simulation model to accurately model memory collision behavior and 'x' output generation. Note that simulation time is longer and debugging may be more difficult. The Simulation Files options in the CORE Generator Project Options determine the type of functional simulation models generated. Table 3 defines the differences between the two functional simulation models.

Table 3: Differences between Simulation Models

	Behavioral Models	Structural Models (Unisim)
When core output is undefined	Never generates 'X'	Generates 'X' to match core
Out-of-range address access	Optionally flags a warning message	Generates 'X'
Collision behavior	Does not generate 'X' on output, and flags a warning message	Generates 'X' to match core
Byte-write collision behavior	Flags all byte-write collisions	Does not flag collisions if byte-writes do not overlap

## Signal List

Table 4 provides a description of the Block Memory Generator core signals. The widths of the data ports (DINA, DOUTA, DINB, and DOUTB) are selected by the user in the CORE Generator GUI. The address port (ADDRA and ADDR B) widths are determined by the memory depth with respect to each port, as selected by the user in the GUI. The write enable ports (WEA and WEB) are busses of width 1 when byte-writes are disabled. When byte-writes are enabled, WEA and WEB widths depend on the byte size and write data widths selected in the GUI.

Table 4: Core Signal Pinout

Name	Direction	Description
CLKA	Input	<b>Port A Clock:</b> Port A operations are synchronous to this clock. For synchronous operation, this must be driven by the same signal as CLKB.
ADDRA	Input	<b>Port A Address:</b> Addresses the memory space for port A read and write operations. Available in all configurations.
DINA	Input	<b>Port A Data Input:</b> Data input to be written into the memory via port A. Available in all RAM configurations.
DOUTA	Output	<b>Port A Data Output:</b> Data output from read operations via port A. Available in all configurations except Simple Dual-port RAM.
ENA	Input	<b>Port A Clock Enable:</b> Enables read, write, and reset operations via port A. Optional in all configurations.
WEA	Input	<b>Port A Write Enable:</b> Enables write operations via port A. Available in all RAM configurations.
SSRA	Input	<b>Port A Synchronous Set/Reset:</b> Resets the port A memory output latch or output register. Optional in all configurations.
REGCEA	Input	<b>Port A Register Enable:</b> Enables the last output register of port A. Optional in all configurations with port A output registers.
CLKB	Input	<b>Port B Clock:</b> Port B operations are synchronous to this clock. Available in dual-port configurations. For synchronous operation, this must be driven by the same signal as CLKA.
ADDRB	Input	<b>Port B address:</b> Addresses the memory space for port B read and write operations. Available in dual-port configurations.
DINB	Input	<b>Port B Data Input:</b> Data input to be written into the memory via port B. Available in True Dual-port RAM configurations.
DOUTB	Output	<b>Port B Data Output:</b> Data output from read operations via Port B. Available in dual-port configurations.
ENB	Input	<b>Port B Clock Enable:</b> Enables read, write, and reset operations via Port B. Optional in dual-port configurations.
WEB	Input	<b>Port B Write Enable:</b> Enables write operations via Port B. Available in Dual-port RAM configurations.
SSRB	Input	<b>Port B Synchronous Set/Reset:</b> Resets the Port B memory output latch or output register. Optional in dual-port configurations.

Table 4: Core Signal Pinout (Continued)

Name	Direction	Description
REGCEB	Input	<b>Port B Register Enable:</b> Enables the last output register of port B. Optional in dual-port configurations with port B output registers.
SBITERR	Output	<b>Single Bit Error:</b> Flags the presence of a single-bit error in memory which has been auto-corrected on the output bus.
DBITERR	Output	<b>Double Bit Error:</b> Flags the presence of a double-bit error in memory. Double-bit errors cannot be auto-corrected by the built-in ECC decode module.

## Generating the Core

The Block Memory Generator is available from the CORE Generator software. To open the Block Memory core from the main CORE Generator window, do the following:

Click View by Function > Memories & Storage Elements > RAMs & ROMs

The following section defines the Block Memory Generator GUI screens and available customization options.

### CORE Generator Parameter Screens

The Block Memory Generator GUI includes five main screens:

- [Block Memory Generator Main Screen](#)
- [Port A Options Screen](#)
- [Port B Options Screen](#)
- [Output Registers and Memory Initialization Screen](#)
- [Simulation Model Options and Information Screen](#)

In addition, all the screens share common tabs and buttons to provide information about the core and to navigate the Block Memory Generator GUI.

## Block Memory Generator Main Screen

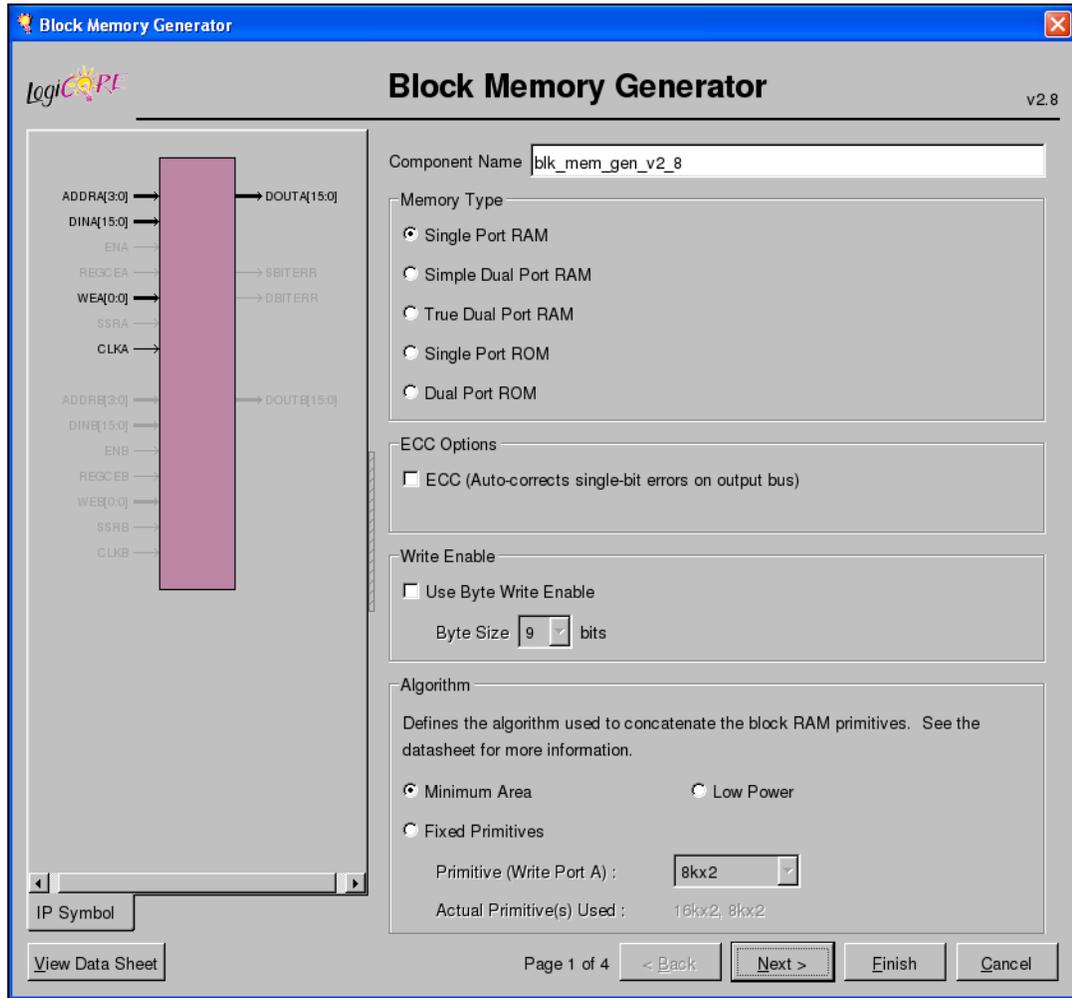


Figure 23: Block Memory Generator Main Screen

### Component Name

The base name of the output files generated for the core. Names must begin with a letter and be composed of any of the following characters: a to z, 0 to 9, and “\_”. Names can not be Verilog or VHDL reserved words.

### Memory Type

Select the type of memory to be generated.

- Single-port RAM
- Simple Dual-port RAM
- True Dual-port RAM
- Single-port ROM
- Dual-port ROM

### ECC Options

When targeting Virtex-5 devices, and when either Single-port RAM or Simple dual-port RAM memory type is selected, the ECC options become available. Selecting ECC enables built-in Hamming error correction for the Virtex-5 FPGA architecture. See "[Built-in Hamming Error Correction Capability](#)" on [page 4](#) for more information.

When using ECC, the following options are limited:

- Byte-write Enable is not available
- All port widths must be identical
- Only Read First Operating mode is supported
- Synchronous Reset (SSR) and the Output Reset Value options are not available
- Memory Initialization is not supported
- No algorithm selection is available

### Write Enable

When targeting Virtex-5, Virtex-4, or Spartan-3A/3A DSP devices, select whether to use the byte-write enable feature. Byte size is either 8-bits (no parity) or 9-bits (including parity). The data width of the memory will be multiples of the selected byte-size.

### Algorithm

Select the algorithm used to implement the memory:

- **Minimum Area Algorithm:** Generates a core using the least number of primitives.
- **Low Power Algorithm:** Generates a core such that the minimum number of block RAM primitives are enabled during a read or write operation.
- **Fixed Primitive Algorithm:** Generates a core that concatenates a single primitive type to implement the memory. Choose which primitive type to use in the drop-down list.

## Port A Options Screen

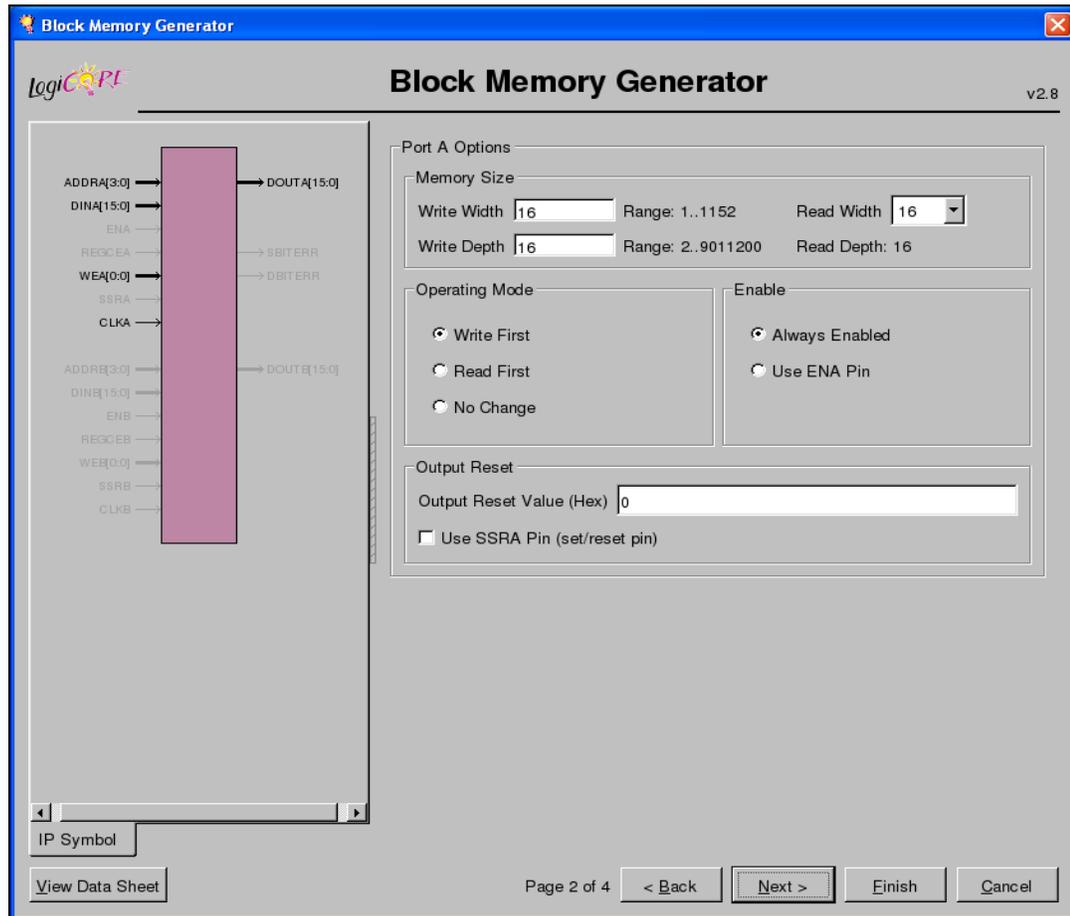


Figure 24: Port A Options

### Memory Size (Port A)

Specify the port A write width and depth. Select the port A read width from the drop-down list of valid choices. The read depth is calculated automatically.

### Operating Mode (Port A)

Specify the port A operating mode.

- READ\_FIRST
- WRITE\_FIRST
- NO\_CHANGE

### Enable (Port A)

Select the enable type:

- Always enabled (no ENA pin available)
- Use ENA pin

**Output Reset (Port A)**

Specify the reset value of the memory output latch and output registers. These values are with respect to the read port widths. Choose whether a set/reset pin (SSRA) is needed.

**Port B Options Screen**

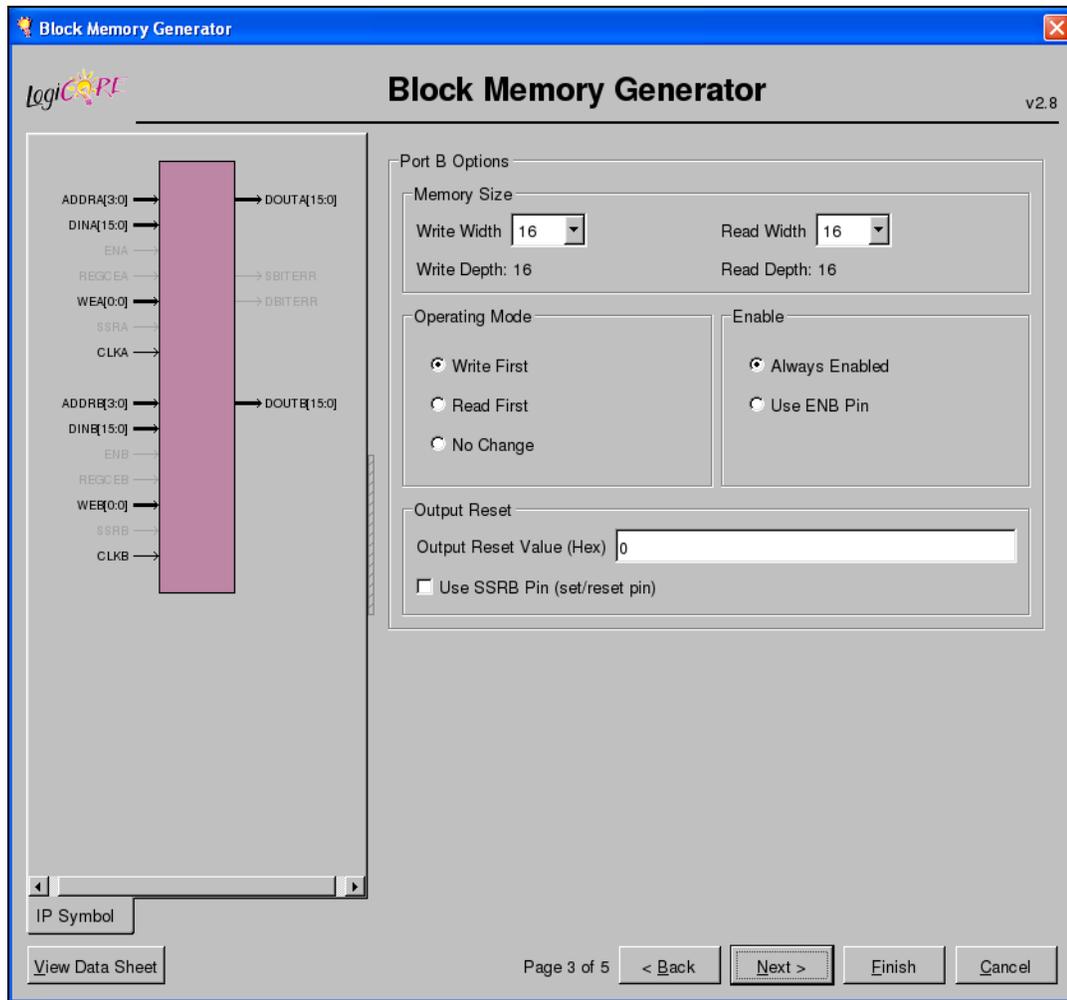


Figure 25: Port B Options

**Memory Size (Port B)**

Select the port B write and read widths from the drop-down list of valid choices. The read depth is calculated automatically.

**Operating Mode (Port B)**

Specify the port B write mode.

- READ\_FIRST
- WRITE\_FIRST
- NO\_CHANGE

### Enable (Port B)

Select the enable type:

- Always enabled (no ENB pin available)
- Use ENB pin

### Output Reset (Port B)

Specify the reset value of the memory output latch and output registers. These values are with respect to the read port widths. Choose whether a set/reset pin (SSRB) is needed.

### Output Registers and Memory Initialization Screen

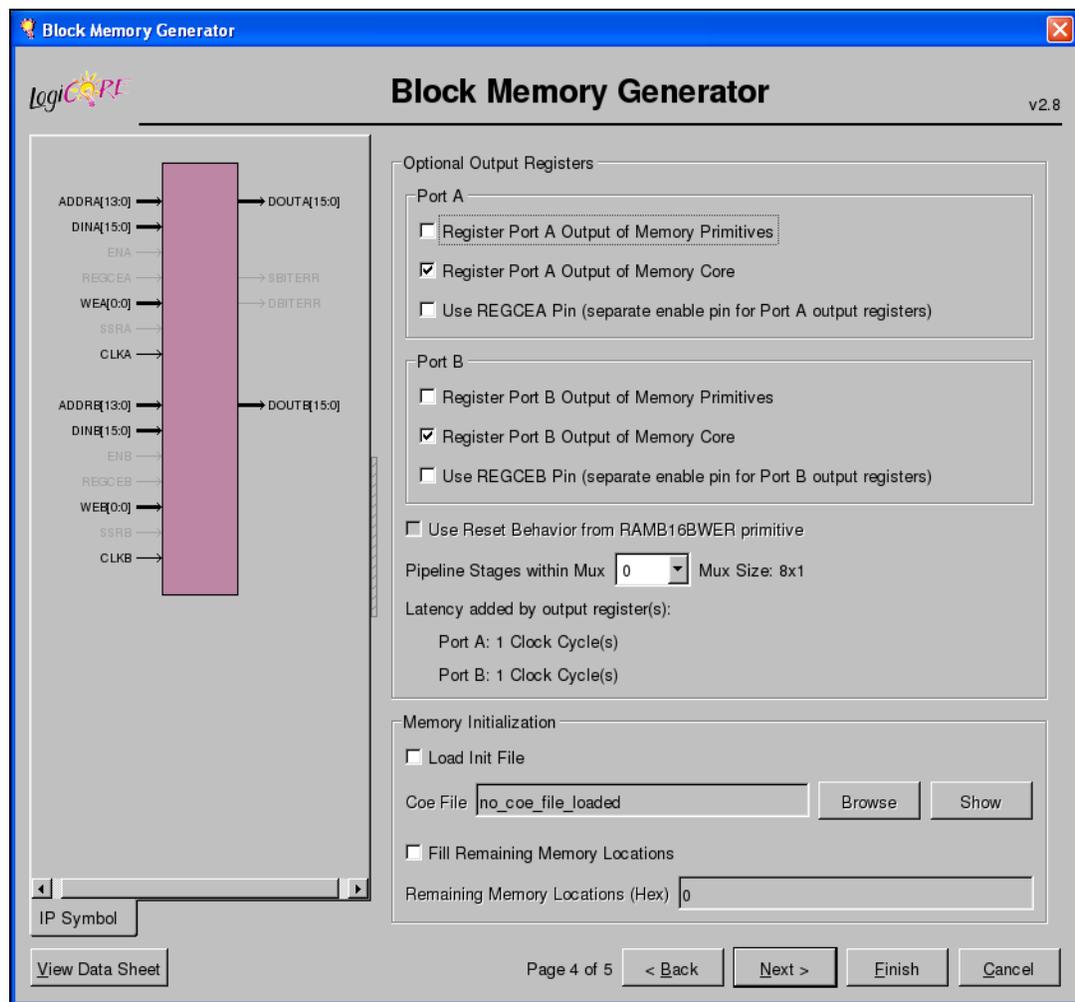


Figure 26: Output Registers and Memory Initialization Screen

### Optional Output Registers

Select the output register stages to include:

- **Register Port [A | B] Output of Memory Primitives.** Select to insert output register after the memory primitives for port A and port B separately. When targeting Virtex-4 or Virtex-5 FPGAs, the embedded output registers in the block RAM primitives are used if the user chooses to register the output of the memory primitives. For other architectures, the registers in the FPGA slices are

used. Note that in Virtex-4 devices, the use of the SSR input prevents the core from using the embedded output registers. See "[Output Register Configurations](#)" on page 47 for more information.

- **Register Port [A | B] Output of Memory Core.** Select for each port (A or B) to insert a register on the output of the memory core for that port. When selected, registers are implemented using FPGA slices to register the core's output.
- **Use REGCE [A | B] Pin.** Select to use a separate REGCEA or REGCEB input pin to control the enable of the last output register stage in the memory. When unselected, all register stages are enabled by ENA/ENB.
- **Use Reset Behavior from RAMB16BWER Primitive.** Selecting this option causes the Block Memory Generator to use the embedded output registers in the Spartan-3A DSP RAMB16BWER primitives; however, it also changes the behavior of the core during reset. See "[Output Register Configurations](#)" on page 47 for more information.
- **Pipeline Stages within Mux.** Available only when the Register Output of Memory Core option is selected for both port A and port B and when the constructed memory has more than one primitive in depth, so that a MUX is needed at the output. Select a value of 0, 1, 2, or 3 from the drop-down list.

The MUX size displayed in the GUI can be used to determine the number of pipeline stages to use within the MUX. Select the appropriate number of pipeline stages for your design based on the device architecture.

#### Memory Initialization

Select whether to initialize the memory contents using a COE file, and whether to initialize the remaining memory contents with a default value. When using asymmetric port widths or data widths, the COE file and the default value are with respect to the port A write width.

## Simulation Model Options and Information Screen

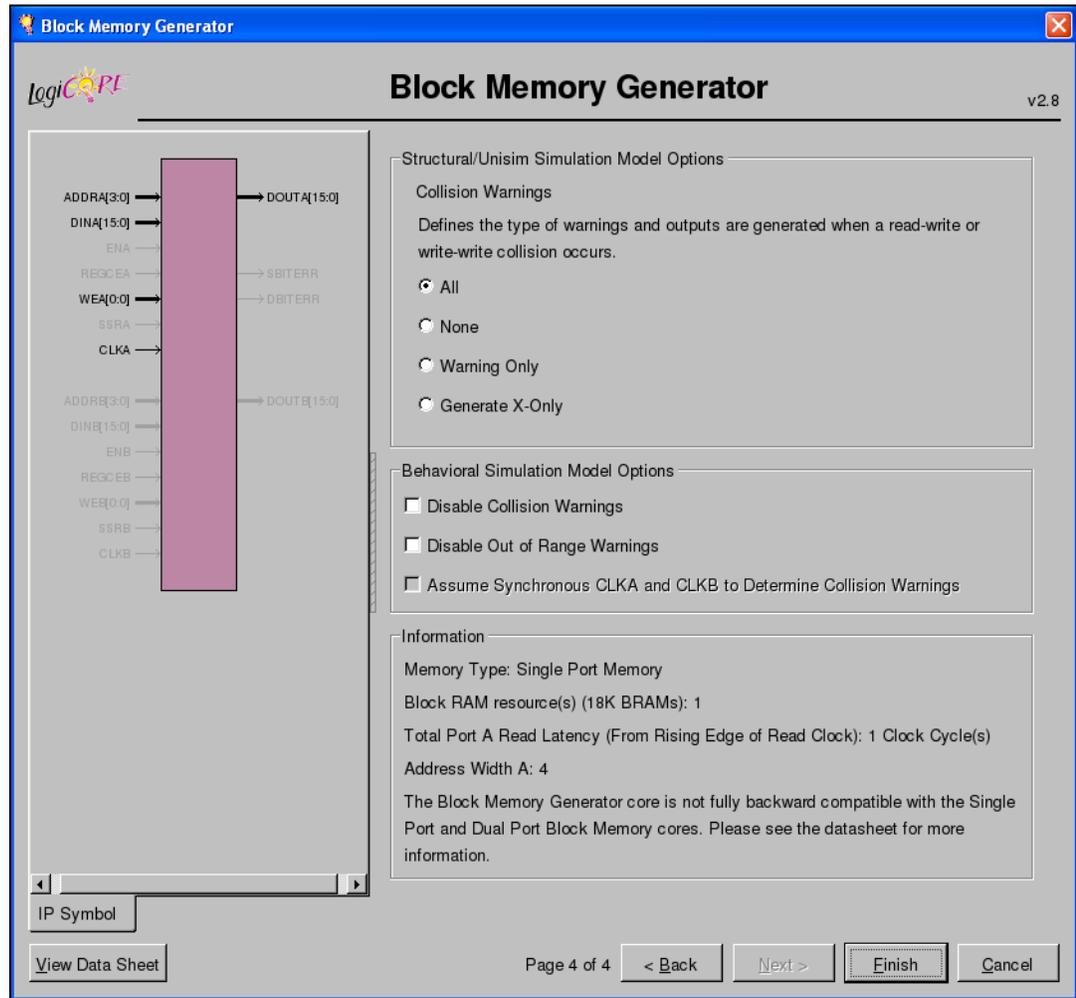


Figure 27: Simulation Model Options and Information Screen

### Structural/UNISIM Simulation Model Options

Select the type of warning messages and outputs generated by the structural simulation model in the event of collisions.

### Behavioral Simulation Model Options

Select the type of warning messages generated by the behavioral simulation model. Select whether the model should assume synchronous clocks for collision warnings.

### Information Section

This section displays an informational summary of the selected core options.

- **Memory Type:** Reports the selected memory type.
- **Block RAM Resources (18k Block RAMs):** Reports the exact number of 18k block RAM primitives which will be used to construct the core. For Virtex-5 devices, each 36k block RAM primitive is counted as two 18k block RAMs.
- **Total Port A Read Latency:** The number of clock cycles for a read operation for port A. This value is

controlled by the optional output registers options for port A on the previous screen.

- **Total Port B Read Latency:** The number of clock cycles for a read operation for port B. This value is controlled by the optional output registers options for port B on the previous screen.
- **Address Width:** The actual width of the address bus to each port.

## Specifying Initial Memory Contents

The Block Memory Generator core supports memory initialization using a memory coefficient (COE) file or the default data option in the CORE Generator GUI, or a combination of both.

The COE file can specify the initial contents of each memory location, while default data specifies the contents of all memory locations. When used in tandem, the COE file can specify a portion of the memory space, while default data fills the rest of the remaining memory space. COE files and default data is formatted with respect to the port A write width (or port A read width for ROMs).

A COE is a text file which specifies two parameters:

- **memory\_initialization\_radix:** The radix of the values in the memory\_initialization\_vector. Valid choices are 2, 10, or 16.
- **memory\_initialization\_vector:** Defines the contents of each memory element. Each value is LSB-justified, separated by a space, and assumed to be in the radix defined by memory\_initialization\_radix.

The following is an example COE file. Note that semi-colon is the end of line character.

```
; Sample initialization file for a
; 32-bit wide by 16 deep RAM
memory_initialization_radix = 16;
memory_initialization_vector =
0 1 2 3 4 5 6 7
8 9 A B C D E F;
```

## Block RAM Usage

The Information panel (screen 5 of the Block Memory Generator GUI) reports the actual number of block RAM blocks to be used. This number is always given as the number of 18k block RAM primitives used.

To estimate this value when using the fixed primitive algorithm, the number of block RAM primitives used is equal to the width ratio (rounded up) multiplied by the depth ratio (rounded up), where the width ratio is the width of the memory divided by the width of the selected primitive, and the depth ratio is the depth of the memory divided by the depth of the primitive selected.

To estimate block RAM usage when using the low power algorithm requires a few more calculations:

- If the memory width is an integral multiple of the width of the widest available primitive for the chosen architecture, then the number of primitives used is calculated in the same way as the fixed primitive algorithm. The width and depth ratios are calculated using the width and depth of the widest primitive. For example, for a memory configuration of 2kx72, the width ratio is 2 and the depth ratio is 4 using the widest primitive of 512x36. As a result, the total available primitives used is 8.
- If the memory width is greater than an integral multiple of the widest primitive, then in addition to the above calculated primitives, more primitives are needed to cover the remaining width. This additional number is obtained by dividing the memory depth by the depth of the additional

primitive chosen to cover the remaining width. For example, a memory configuration of 17kx37 requires one 512x36 primitive to cover the width of 36, and an additional 16kx1 primitive to cover the remaining width of 1. To cover the depth of 17K, 34 512x36 primitives and 2 16kx1 primitives are needed. As a result, the total number of primitives used for this configuration is 36.

- If the memory width is less than the width of the widest primitive, then the smallest possible primitive that covers the memory width is chosen for the solution. The total number of primitives used is obtained by dividing the memory depth by the depth of the chosen primitive. For example, for a memory configuration of 2kx32, the chosen primitive is 512x36, and the total number of primitives used is 2k divided by 512, which is 4.

When using the minimum area algorithm, it is not as easy to determine the exact block RAM count. This is because the actual algorithms perform complex manipulations to produce optimal solutions. The optimistic estimate is total memory bits divided by 18k (the total number of bits per primitive) rounded up. Given that this algorithm packs block RAMs very efficiently, this estimate is often very accurate for most memories.

### LUT Utilization and Performance

The LUT utilization and performance of the core are directly related to the arrangement of primitives and the selection of output registers. Particularly, the number of primitives cascaded in depth to implement a memory determines the size of the output multiplexer and the size of the input decoder, which are implemented in the FPGA fabric.

**Note:** Although the primary goal of the minimum area algorithm is to use the minimum number of block RAM primitives, it has a secondary goal of maximizing performance – as long as block RAM usage does not increase.

### Resource Utilization and Performance Examples

The following tables provide examples of actual resource utilization and performance for Block Memory Generator implementations. Each section highlights the effects of a specific feature on resource utilization and performance.

Benchmarks were performed targeting a Virtex-II Pro FPGA in the -5 speed grade (2vp30-ff1152-5), and a Virtex-4 FPGA in the -10 speed grade (4vlx60-ff1148-10), and a Virtex-5 FPGA in the -1 speed grade (5vlx30-ff324-1). Better performance may be possible with higher speed grades.

In the benchmark designs described below, the core was encased in a wrapper with input and output registers to remove the effects of IO delays from the results; performance may vary depending on the user design. The minimum area algorithm was used unless otherwise noted. The examples below highlight the use of embedded registers in Virtex-5 and Virtex-4 devices, and the subsequent performance improvement that may result.

## Single Primitive

The Block Memory Generator does not add additional logic if the memory can be implemented in a single block RAM primitive. [Table 5](#) and [Table 6](#) define performance data for single-primitive memories.

*Table 5: Single Primitive Examples - Virtex-4 and Virtex-II Pro*

Memory Type	Options	Width x Depth	Resource Utilization				Performance (MHz)	
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Virtex-II Pro	Virtex-4
True Dual-port RAM		36x512	1	0	0	0	330	310
		9x2k	1	0	0	0	345	335
	Virtex-4 or Virtex-5 Embedded Output Registers	36x512	1	0	0	0	N/A	385
		9x2k	1	0	0	0	N/A	395

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

*Table 6: Single Primitive Examples - Virtex-5*

Memory Type	Options	Width x Depth	Resource Utilization				Performance (MHz)
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Virtex-5
True Dual-port RAM		36x512	1	0	0	0	300
		9x2k	1	0	0	0	300
	Virtex-4 or Virtex-5 Embedded Output Registers	36x512	1	0	0	0	395
		9x2k	1	0	0	0	455

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

## Output Registers

The Block Memory Generator optional output registers increase the performance of memories by isolating the block RAM primitive clock-to-out delays and the data output multiplexer delays.

The output registers are only implemented for output ports. For this reason, when output registers are used, a Single-port RAM requires fewer resources than a True Dual-port RAM. Note that the effects of the core output registers are not fully illustrated due to the simple register wrapper used. In a full-scale user design, core output registers may improve performance notably.

For Virtex-II Pro devices, the performance of shift registers is worse than flip flops. So, the shift registers used for registering the MUX select lines is the main performance bottleneck, and a configuration with no output registers gives a slightly better performance than a configuration with output registers.

**Table 7: Virtex-II Pro Output Register Examples**

Memory Type	Width x Depth	Output Register Option	Virtex-II Pro				
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Performance (MHz)
True Dual-port RAM	17x5k		5	0	6	60	325
		Primitive	5	6	92	66	300
		Core	5	6	40	142	300
		Primitive, Core	5	6	126	148	300
Single-port RAM	17x5k		5	0	3	30	325
		Primitive	5	3	46	33	300
		Core	5	0	22	30	350
		Primitive, Core	5	2	66	32	300

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

In Virtex-4 architectures, the embedded block RAM may be utilized, reducing the FPGA fabric resources required to create the registers.

**Table 8: Virtex-4 Output Register Examples**

Memory Type	Width x Depth	Output Register Option	Virtex-4				
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Performance (MHz)
True Dual-port RAM	17x5k		5	0	6	60	275
		Primitive	5	6	6	66	425
		Core	5	0	40	142	250
		Primitive, Core	5	6	40	148	375
Single-port RAM	17x5k		5	0	3	30	275
		Primitive	5	3	3	33	425
		Core	5	0	20	30	275
		Primitive, Core	5	2	22	32	450

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

In Virtex-5 FPGA architectures, the embedded block RAM may be used to reduce the FPGA fabric resources required to create the registers.

Table 9: Virtex-5 Output Register Examples

Memory Type	Width x Depth	Output Register Options	Virtex-5				
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Performance (MHz)
True Dual-port RAM	17x5k		3	0	6	36	300
		Primitive	3	6	6	42	525
		Core	3	0	40	36	300
		Primitive, Core	3	6	40	42	500
Single-port RAM	17x5k		3	0	3	18	275
		Primitive	3	3	3	21	475
		Core	3	0	20	18	300
		Primitive, Core	3	3	20	21	500

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

## Aspect Ratios

The Block Memory Generator selectable port and data width aspect ratios may increase block RAM usage and affect performance, because aspect ratios limit the primitive types available to the algorithm, which can reduce packing efficiency. Large aspect ratios, such as 1:32, have a greater impact than small aspect ratios. Note that width and depth are reported with respect to the port A write interface.

Table 10: Virtex-II Pro Aspect Ratio

Memory Type	Width x Depth	Port Aspect Ratio	Virtex-II Pro				
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Performance (MHz)
True Dual-port RAM	17x5k	1:1	5	0	6	60	275
	136x640	1:8 <sup>2</sup>	9	0	0	0	275

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

2. A port is 136x640; B port is 17x5k.

Table 11: Virtex-4 Aspect Ratio

Memory Type	Width x Depth	Data Width Aspect Ratio	Virtex-4				
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Performance (MHz)
Single-port RAM	17x5k	1:1	5	0	3	30	275
		1:8 <sup>2</sup>	9	0	0	0	300
	136x640	8:1 <sup>3</sup>	9	0	0	0	350

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

2. Read port is 136x640; write port is 17x5k.

3. Read port is 17x5k, write port is 136x640.

Table 12: Virtex-5 Aspect Ratio

Memory Type	Width x Depth	Aspect Ratio	Virtex-5				
			Block RAMs	Shift Regs	FFs	LUTs <sup>1</sup>	Performance (MHz)
Single-port RAM	17x5k	1:1	3	0	3	18	275
		1:8 <sup>2</sup>	5	0	0	0	275
	136x640	8:1 <sup>3</sup>	5	0	0	0	275 <sup>4</sup>

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.
2. Read port is 136x640; write port is 17x5k.
3. Read port is 17x5k, write port is 136x640.
4. This configuration was generated using the xc5vlx50-ff676-1 part.

## Algorithm

The differences between the minimum area, low power and fixed primitive algorithms are discussed in detail in "[Selectable Memory Algorithm](#)" on page 2. Table 13 shows examples of the resource utilization and the performance difference between them for two selected configurations for Virtex-II Pro and Virtex-4 FPGA architectures.

Table 13: Memory Algorithm Examples Virtex-II Pro and Virtex-4

Memory Type	Width x Depth	Algorithm Type	Resource Utilization				Performance (MHz)	
			Block RAM	Shift Regs	FFs	LUTs <sup>1</sup>	Virtex-II Pro	Virtex-4
Single-port RAM	17x5k	Minimum area	5	0	3	30	300	275
		Fixed Primitive using 18x1k block RAM	5	0	3	57	225	225
		Low power	5	0	3	57	225	225
	36x4k	Minimum area	8	0	1	36	300	300
		Fixed Primitive using 36x512 block RAM	8	0	3	152	225	225
		Low power	8	0	3	152	225	225

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

**Table 14** shows examples of the resource utilization and the performance difference between them for two selected configurations for Virtex-5 FPGA architecture.

**Table 14: Memory Algorithm Examples Virtex-5**

Memory Type	Width x Depth	Algorithm Type	Resource Utilization				Performance (MHz)
			Block RAM	Shift Regs	FFs	LUTs <sup>1</sup>	Virtex-5
Single-port RAM	17x5k	Minimum area	3	0	3	18	275
		Fixed Primitive using 18x1k block RAM	3	0	3	30	300
		Low power	3	0	3	20	300
	36x4k	Minimum area	4	0	0	0	300
		Fixed Primitive using 36x512 block RAM	4	0	2	40	275
		Low power	4	0	2	40	275

1. LUTs are reported as the number of 4-input LUTs, and do not reflect the number of LUTs used as a route-through.

## Supplemental Information

The following sections provide additional information about working with the Block Memory Generator core.

- **Compatibility with Older Memory Cores:** Defines the differences between older memory cores and the Block Memory Generator core.
- **Construction of Smaller Memories:** Explains the process of creating shallower or wider memories using dual port block memory.
- **SIM Parameters:** Defines the SIM parameters used to specify the core configuration.
- **Output Register Configurations:** Provides information optional output registers used to improve core performance.

## Compatibility with Older Memory Cores

This section contains important information for users working with a previous version of the Block Memory Generator core.

### Using the Block Memory Generator Migration Kit

The Block Memory Generator core replaces the following legacy cores:

- Dual Port Block Memory (v6.x)
- Single Port Block Memory (v6.x)

Cores generated by the Block Memory Generator are not drop-in replacements for the v6.x cores listed above as there are a number of feature and interface changes in the Block Memory Generator.

To help you migrate designs containing legacy LogiCORE IP v6.x Dual and Single Port Block Memory cores to the new Block Memory Generator core, Xilinx provides the Block Memory Generator Migration Kit to manage migration-related tasks without manually editing your design. For information about using the Migration Kit, see the [Block Memory Generator Migration Kit Product Page](#).

### Differences Between Cores

This section describes the differences between the Single and Dual Port Block Memory cores and the Block Memory Generator. The new Block Memory Generator is not backward compatible with the Single and Dual Port Block Memory core in several aspects.

- Single Port Block Memory core and the Block Memory Generator do not have the same port names for a single-port memory configuration.
- Dual Port Block Memory core and the Block Memory Generator do not have the same reset pin name.
- XCO files for the previous memory cores are NOT compatible with the Block Memory Generator.
- Pin polarity, handshaking, and input register features supported by the previous memory cores are not supported in the Block Memory Generator.
- The behavior of set/reset pin (SSR) and the enable pin (EN) in the Block Memory Generator will differ from previous cores when using optional output registers.
- The Dual-port Block Memory core supports all combinations of read-only, write-only, and read/write ports for A & B ports. Some of these combination are only available in the Block Memory Generator by manually tying-off unused ports of a True Dual-port RAM.

**Note:** The Block Memory Migration Kit automatically ties-off the appropriate ports when used to migrate from a Dual-port Block Memory core.

## Port Memory Pin Names

In the Block Memory Generator, the port names have been changed from the older Single Port Block Memory and Dual Port Block Memory cores to reflect the actual ports on the block RAM primitives. [Table 15](#) and [Table 16](#) reflect the changes in port names.

*Table 15: Port Name Changes from Single Port Block Memory Core*

<b>Single Port Block Memory v6.2 (Old core)</b>	<b>Block Memory Generator v2.1 (New Core, Single Port Configuration)</b>
DIN	DINA
ADDR	ADDRA
EN	ENA
WE	WEA
SINIT	SSRA
CLK	CLKA
DOUT	DOUTA
ND	Not supported
RFD	Not supported
RDY	Not supported

*Table 16: Port Name Changes from Dual Port Block Memory Core*

<b>Dual Port Block Memory v6.2 (Old core)</b>	<b>Block Memory Generator v2.1 (New Core, Single Port Configuration)</b>
SINITA	SSRA
SINITB	SSRB
ND [A B]	Not supported
RFD [A B]	Not supported
RDY [A B]	Not supported

## Obsolete Features

### Pin Polarity Option

The Block Memory Generator does not support pin polarity options on the clock, enable, write enable and set/reset input pins. When active low signaling is desired, users can invert the signals prior to the input of the core.

### Handshaking Pins

The Block Memory Generator does not support handshaking pins: ND[A|B], RFD[A|B] and RDY[A|B].

## Input Registers

The Block Memory Generator does not support input registers on the DIN, ADDR, and WE input ports.

## Read/Write Ports

The Block Memory Generator does not support all combinations of read-only, write-only, and read/write ports for ports A & B. Use a True Dual-port RAM type and manually tie-off unused ports to achieve configurations with only one write-only or read-only port.

**Note:** The Block Memory Generator Migration Kit automatically ties-off appropriate ports when used to migrate from a Dual-port Block Memory core.

## Modified Behaviors

### Enable Pin

**Table 17** illustrates the difference in enable behavior between previous cores and the Block Memory Generator.

*Table 17: Old and New Core Differences*

Single (or Dual) Port Block Memory v6.2	Block Memory Generator v2.1
Single enable pin that controls the enables of all registers and memories	Two optional enable pins provided: <ul style="list-style-type: none"> <li>• REGCE—optionally controls the enables of the registers in the last output stage</li> <li>• EN—controls the enables of all other registers and memory</li> </ul>

### Synchronous Reset Pin

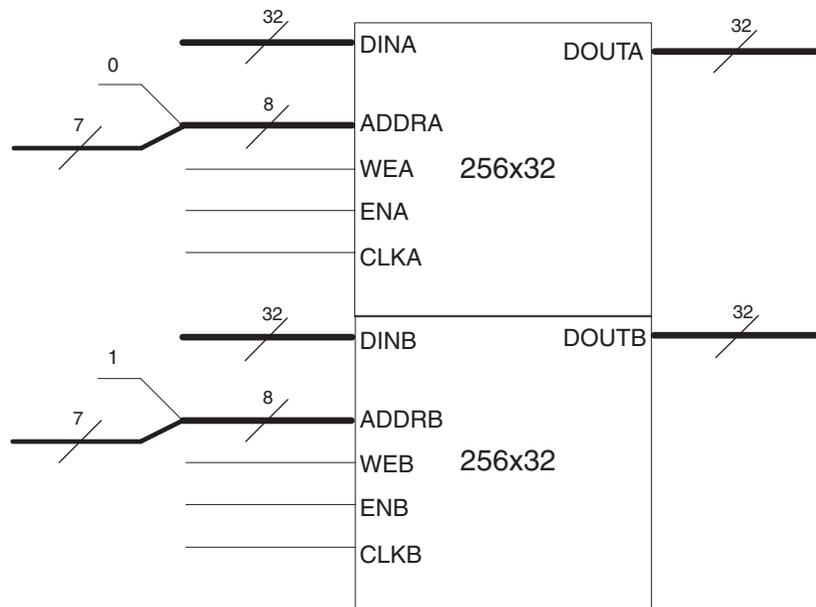
In the previous memory cores, the synchronous reset (SINIT pin) initializes all memory latches and output registers. In the Block Memory Generator, the synchronous set/reset (SSR pin) only resets the registers in the last output stage. When there are no output register stages present, it will initialize the memory latches.

## Construction of Smaller Memories

A single memory of a given depth can be used to construct two independent memories of half the depth. This can be achieved by tying the MSB of the address of one port to '0' and the MSB of the address of the second port to '1'. This feature can be used only for memories that are at most one primitive deep.

As an example, consider the construction of two independent 128x32 memories using a single 256x32 memory. Generate a 256x32 True Dual Port memory core in Core Generator using the desired parameters. Once generated, the MSB of ADDRA is connected to '0' and the MSB of ADDR B is connected to '1'. This effectively turns a True Dual Port 256x32 memory (with both A and B ports sharing the same memory space) into two single-port 128x32 memories, where port A is a single-port memory addressing memory locations 0-127, and port B is a single-port memory addressing memory locations 128-255. In this configuration, the two 128x32 memories function completely independent of each other, in a single block RAM primitive. While initializing such a memory, the input COE file should contain 256 32-bit

wide entries. The first 128 entries initialize memory A, while the second 128 entries initialize memory B, as shown in **Figure 28**.



**Figure 28: Construction of two independent 128x32 memories using a single 256x32 memory**

For construction of memories narrower than 32-bits, for example 19 bits, the widths of both ports A and B can be set to 19 in the Core Generator GUI. The cores are then connected the same way. The COE entries must then be just 19 bits wide. Alternately, a 32-bit wide memory may be generated, and only the least significant 19 bits may be used. COE entries in this case will be trimmed or padded with 0s automatically to fill the appropriate number of bits.

For construction of memories shallower than 128 words, for example 23 words, simply use the first 23 entries in the memory, tying off the unused address bits to '0'. Alternately, use the same strategy as above to turn a True Dual Port 64-word deep memory into two 32-word deep memories.

## SIM Parameters

**Table 18** defines the SIM parameters used to specify the configuration of the core. These parameters are only used to manually instantiate the core in HDL, calling the CORE Generator dynamically. This parameter list does not apply to users that generate the core using the CORE Generator GUI.

*Table 18: SIM Parameters*

	SIM Parameter	Type	Values	Description
1	C_FAMILY	String	"Virtex2" "Virtex4" "Virtex5"	Target device family
2	C_XDEVICEFAMILY	String	"Virtex-II" "Virtex-II Pro" "Virtex4" "Virtex5" "Spartan-3A" "Spartan-3A DSP"	Finest resolution target family derived from the parent C_FAMILY
3	C_MEM_TYPE	Integer	0: Single Port RAM 1: Simple Dual Port RAM 2: True Dual Port RAM 3: Single Port ROM 4: Dual Port RAM	Type of memory
4	C_ALGORITHM	Integer	0 (selectable primitive), 1 (minimum area), 2 (low power)	Type of algorithm
5	C_PRIM_TYPE	Integer	0 (1-bit wide) 1 (2-bit wide) 2 (4-bit wide) 3 (9-bit wide) 4 (18-bit wide) 5 (36-bit wide) 6 (72-bit wide, single-port only)	If fixed primitive algorithm is chosen, determines which type of primitive to use to build memory
6	C_BYTE_SIZE	Integer	9, 8	Defines size of a byte: 9 bits or 8 bits
7	C_SIM_COLLISION_CHECK	String	None, Generate_X, All, Warnings_only	Defines warning collision checks in structural/unisim simulation model
8	C_COMMON_CLOCK	Integer	0, 1	Determines whether to optimize behavioral model collision check by assuming clocks are synchronous
9	C_DISABLE_WARN_BHV_COLL	Integer	0, 1	Disables the behavioral model from generating warnings due to read-write collisions
10	C_DISABLE_WARN_BHV_RANGE	Integer	0, 1	Disables the behavioral model from generating warnings due to address out of range
11	C_LOAD_INIT_FILE	Integer	0, 1	Defined whether to load initialization file
12	C_INIT_FILE_NAME	String	"..."	Name of initialization file (MIF format)
13	C_USE_DEFAULT_DATA	Integer	0, 1	Determines whether to use default data for the memory
14	C_DEFAULT_DATA	String	"..."	Defines a default data for the memory

**Table 18: SIM Parameters (Continued)**

	<b>SIM Parameter</b>	<b>Type</b>	<b>Values</b>	<b>Description</b>
15	C_HAS_MEM_OUTPUT_REGS_A	Integer	0, 1	Determines whether port A has a register stage added at the output of the memory latch
16	C_HAS_MEM_OUTPUT_REGS_B	Integer	0,1	Determines whether port B has a register stage added at the output of the memory latch
17	C_HAS_MUX_OUTPUT_REGS_A	Integer	0,1	Determines whether port A has a register stage added at the output of the memory core
18	C_HAS_MUX_OUTPUT_REGS_B	Integer	0,1	Determines whether port B has a register stage added at the output of the memory core
19	C_MUX_PIPELINE_STAGES	Integer	0,1,2,3	Determines the number of pipeline stages within the MUX for both port A and port B
20	C_WRITE_WIDTH_A	Integer	1 to 1152	Defines width of write port A
21	C_READ_WIDTH_A	Integer	1 to 1152	Defines width of read port A
22	C_WRITE_DEPTH_A	Integer	2 to 9011200	Defines depth of write port A
23	C_READ_DEPTH_A	Integer	2 to 9011200	Defines depth of read port A
24	C_ADDRA_WIDTH	Integer	1 to 24	Defines the width of address A
25	C_WRITE_MODE_A	String	Write_First, Read_first, No_change	Defines the write mode for port A
26	C_HAS_ENA	Integer	0, 1	Determines whether port A has an enable pin
27	C_HAS_REGCEA	Integer	0, 1	Determines whether port A has an enable pin for its output register
28	C_HAS_SSRA	Integer	0, 1	Determines whether port A has an synchronous reset pin
29	C_SINITA_VAL	String	"..."	Defines initialization/power-on value for port A output
30	C_USE_BYTE_WEA	Integer	0, 1	Determines whether byte-write feature is used on port A For True Dual Port configurations, this value is the same as C_USE_BYTE_WEB, since there is only a single byte write enable option provided
31	C_WEA_WIDTH	Integer	1 to 128	Defines width of WEA pin for port A
32	C_WRITE_WIDTH_B	Integer	1 to 1152	Defines width of write port B
33	C_READ_WIDTH_B	Integer	1 to 1152	Defines width of read port B
34	C_WRITE_DEPTH_B	Integer	2 to 9011200	Defines depth of write port B
35	C_READ_DEPTH_B	Integer	2 to 9011200	Defines depth of read port B
36	C_ADDRB_WIDTH	Integer	1 to 24	Defines the width of address B

Table 18: SIM Parameters (Continued)

	SIM Parameter	Type	Values	Description
37	C_WRITE_MODE_B	String	Write_First, Read_first, No_change	Defines the write mode for port B
38	C_HAS_ENB	Integer	0, 1	Determines whether port B has an enable pin
39	C_HAS_REGCEB	Integer	0, 1	Determines whether port B has an enable pin for its output register
40	C_HAS_SSRB	Integer	0, 1	Determines whether port B has an synchronous reset pin
41	C_SINITB_VAL	String	"..."	Defines initialization/power-on value for port B output
42	C_USE_BYTE_WEB	Integer	0, 1	Determines whether byte-write feature is used on port B This value is the same as C_USE_BYTE_WEA, since there is only a single byte write enable provided
43	C_WEB_WIDTH	Integer	1 to 128	Defines width of WEA pin for port B
44	C_USE_ECC	Integer	0,1	Determines ECC options 0 = No ECC 1 = ECC
45	C_USE_RAMB16BWER_RST_BHV	Integer	0,1	Determines if the embedded registers in Spartan-3A DSP RAMB16BWER primitives are used

## Output Register Configurations

The Block Memory Generator core provides optional output registers that can be selected for port A and port B separately, and that may improve the performance of the core. Because Virtex-5, Virtex-4, and Spartan-3A DSP FPGA block RAM primitives have embedded output registers that previous architectures did not support, the configurations described in the sections that follow are separated into Virtex-4 and Virtex-5 devices, Spartan-3A DSP device, and Virtex-II device and implementations. Figure 29 shows the Optional Output Registers section of the Block Memory Generator GUI.

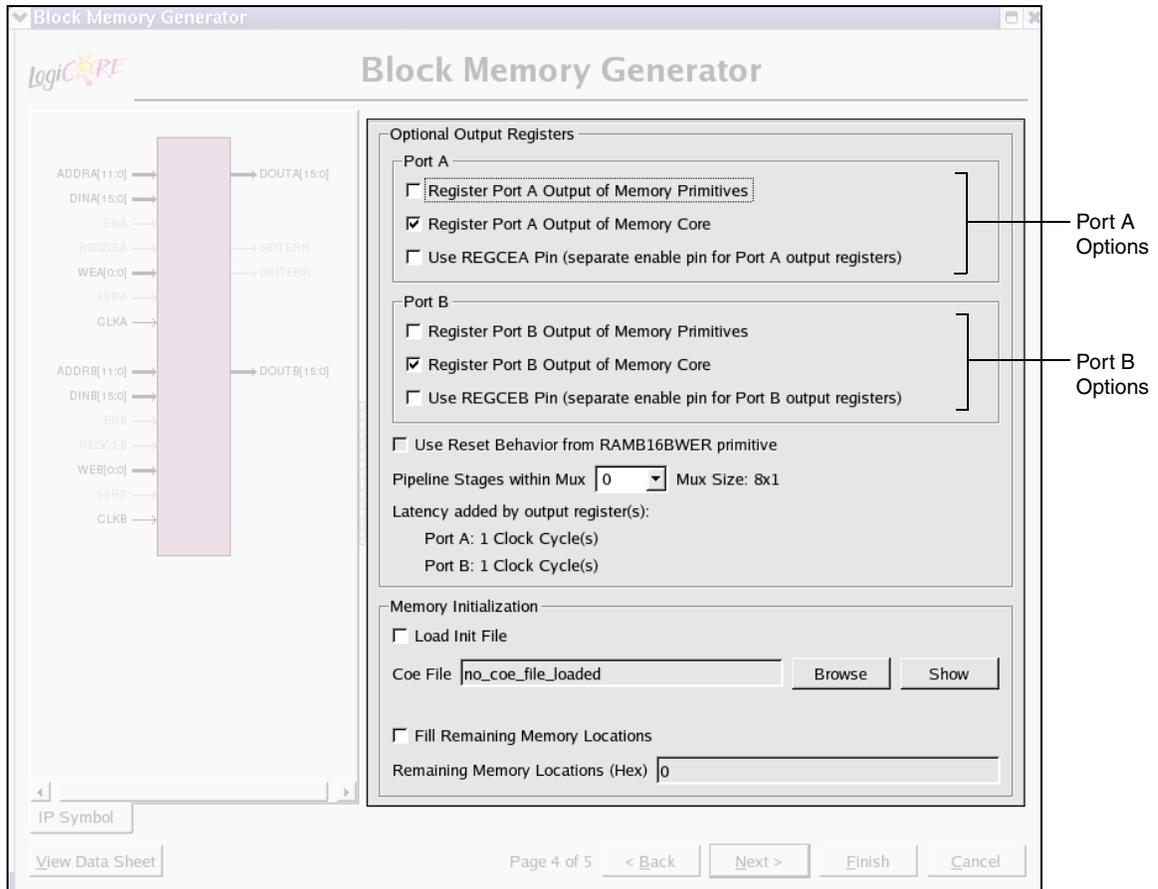


Figure 29: Optional Output Registers Section

### Virtex-5 and Virtex-4 FPGA: Output Register Configurations

To tailor register options for Virtex-5 and Virtex-4 FPGA configurations, two selections, one each for port A and port B, are provided. The embedded output registers are enabled for the corresponding port(s) when Register Port [A | B] Output of Memory Primitives is selected. Similarly, registers at the output of the core are enabled for the corresponding port(s) by selecting Register Port [A | B] Output of Memory Core. Figures 30 through 35 illustrate the Virtex-5 and Virtex-4 FPGA output register configurations.

### Virtex-5 and Virtex-4 FPGA: Memory with Primitive and Core Output Registers

With both Register Output of Memory Primitives and Register Output of Memory Core options selected, a memory core is generated with both the Virtex-4 or Virtex-5 FPGA embedded output registers and a register on the output of the core for the selected port(s), as shown in Figure 30. This configuration may provide improved performance for building large memory constructs.

- |   |           |   |
|---|-----------|---|
| <b>Port A</b>   | <b>or</b> | <b>Port B</b>   |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |           | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Core       |           | <input checked="" type="checkbox"/> Register Port B Output of Memory Core       |

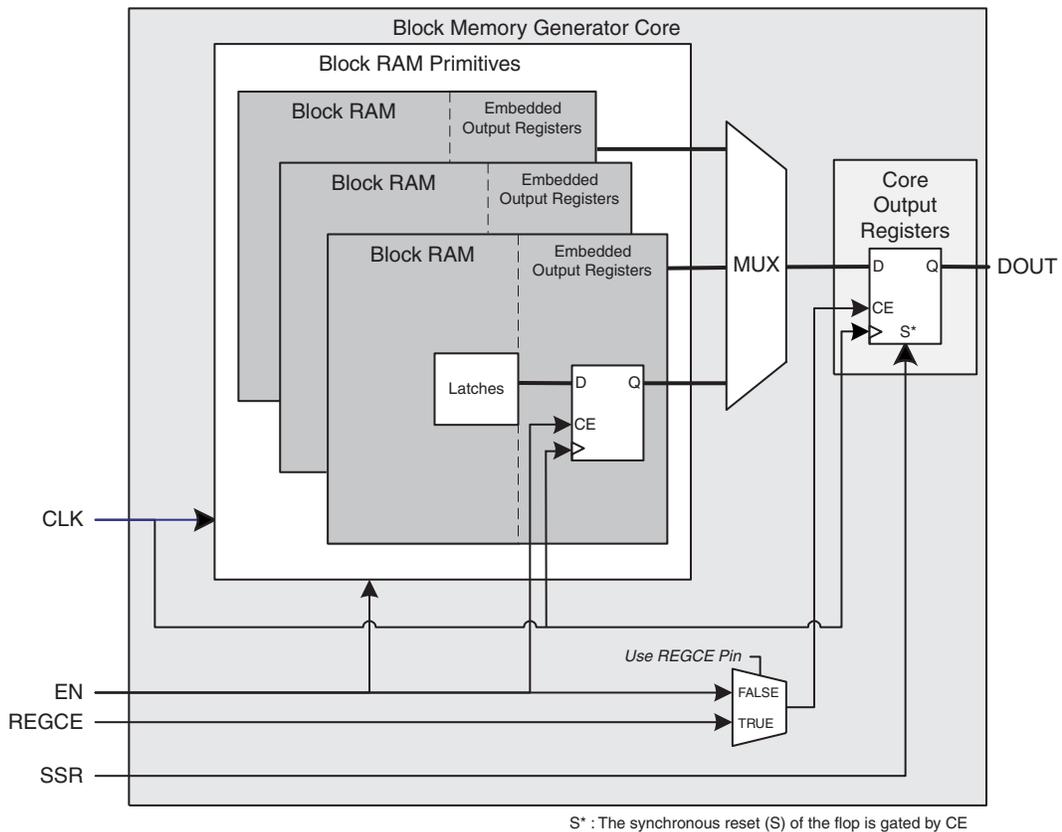


Figure 30: Virtex-4 or Virtex-5 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Enabled

**Virtex-5 FPGA: Memory with Primitive Output Registers**

When Register Port [A | B] Output of Memory Primitives is selected, a memory core that registers the output of the block RAM primitives for the selected port(s) is generated. In Virtex-5 devices, these registers are always implemented using the output registers embedded in the Virtex-5 FPGA block RAM architecture. The output of any multiplexing that may be required to combine multiple primitives is not registered in this configuration, as shown in Figure 31.

- |   |           |   |
|---|-----------|---|
| <b>Port A</b>   | <b>or</b> | <b>Port B</b>   |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |           | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core                  |           | <input type="checkbox"/> Register Port B Output of Memory Core                  |

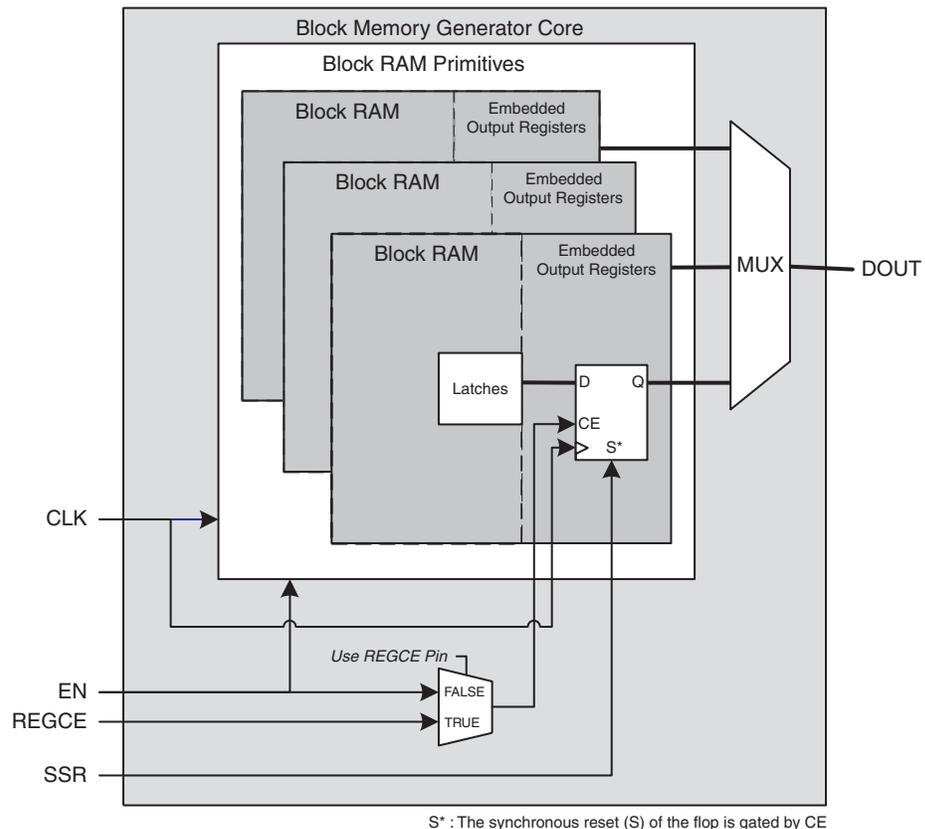


Figure 31: Virtex-5 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled

**Virtex-4 FPGA: Memory with Primitive Output Registers without SSR**

When Register Port [A | B] Output of Memory Primitives is selected and the corresponding Use SSR [A | B] Pin (set/reset pin) is unselected, a memory core that registers the output of the block RAM primitives for the selected port(s) using the output registers embedded in Virtex-4 FPGA architecture is generated. The output of any multiplexing that may be required to combine multiple primitives is not registered in this configuration, as shown in Figure 32.

- | Port A  | or | Port B  |
|---|----|---|
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core                  |    | <input type="checkbox"/> Register Port B Output of Memory Core                  |
| <input type="checkbox"/> Use SSRA Pin (set/reset pin)                           |    | <input type="checkbox"/> Use SSRB Pin (set/reset pin)                           |

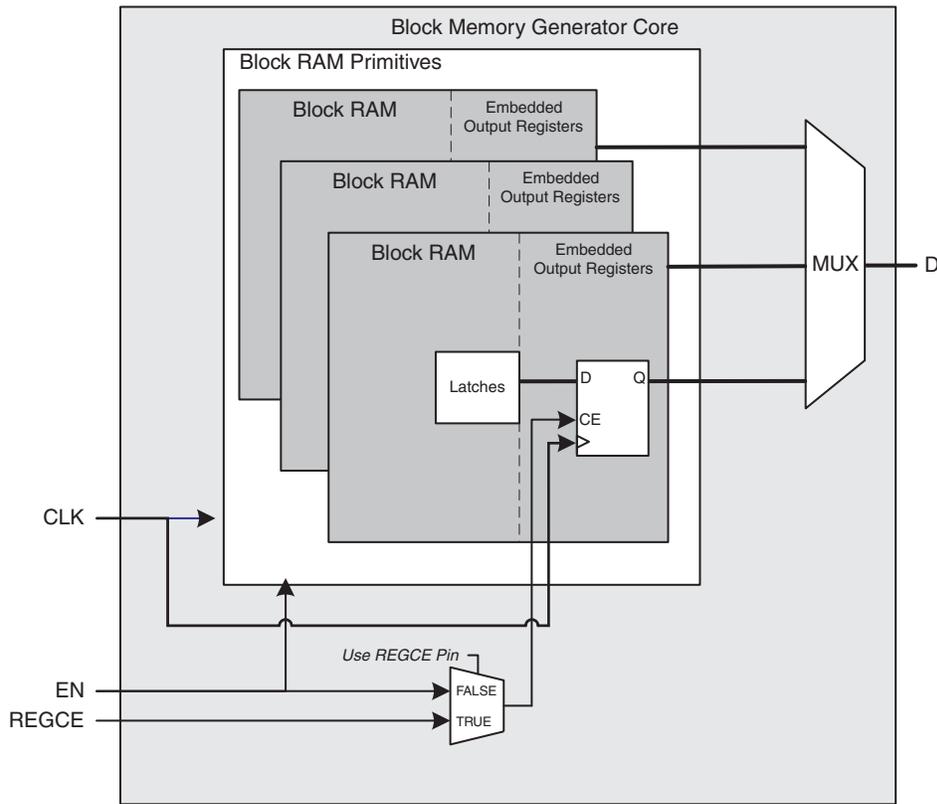
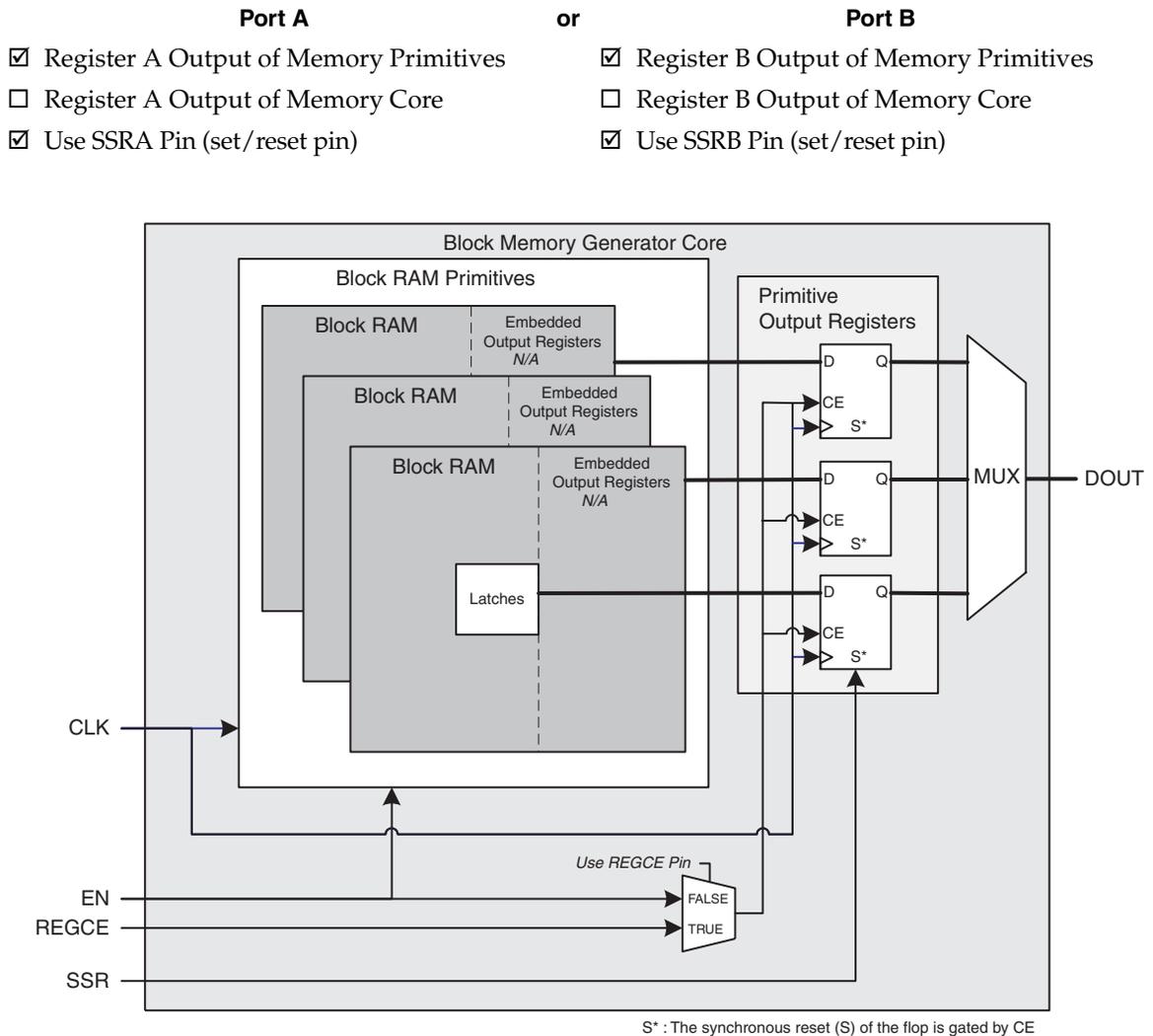


Figure 32: Virtex-4 Block Memory Generated with only Register Port [A | B] Output of Memory Primitives Enabled

**Virtex-4 FPGA: Memory with Primitive Output Registers with SSR**

If either Use SSRA Pin (set/reset pin) or Use SSRB Pin (set/reset pin) is selected from the Output Reset section of the Port Options screen(s), the Virtex-4 embedded block RAM registers cannot be used for the corresponding port(s). The primitive output registers are built from FPGA fabric, as shown in Figure 33.



**Figure 33: Virtex-4 Block Memory Generated with Register Output of Memory Primitives and Use SSR[AIB] Pin Options Enabled**

### Virtex-5 and Virtex-4 FPGA: Memory with Core Output Registers

When only Register Port [A|B] Output of Memory Core is selected, the Virtex-5/Virtex-4 device's embedded registers are disabled for the selected ports in the generated core, as shown in Figure 34.

- |   |    |   |
|---|----|---|
| <b>Port A</b>   | or | <b>Port B</b>   |
| <input type="checkbox"/> Register Port A Output of Memory Primitives      |    | <input type="checkbox"/> Register Port B Output of Memory Primitives      |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Core |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Core |

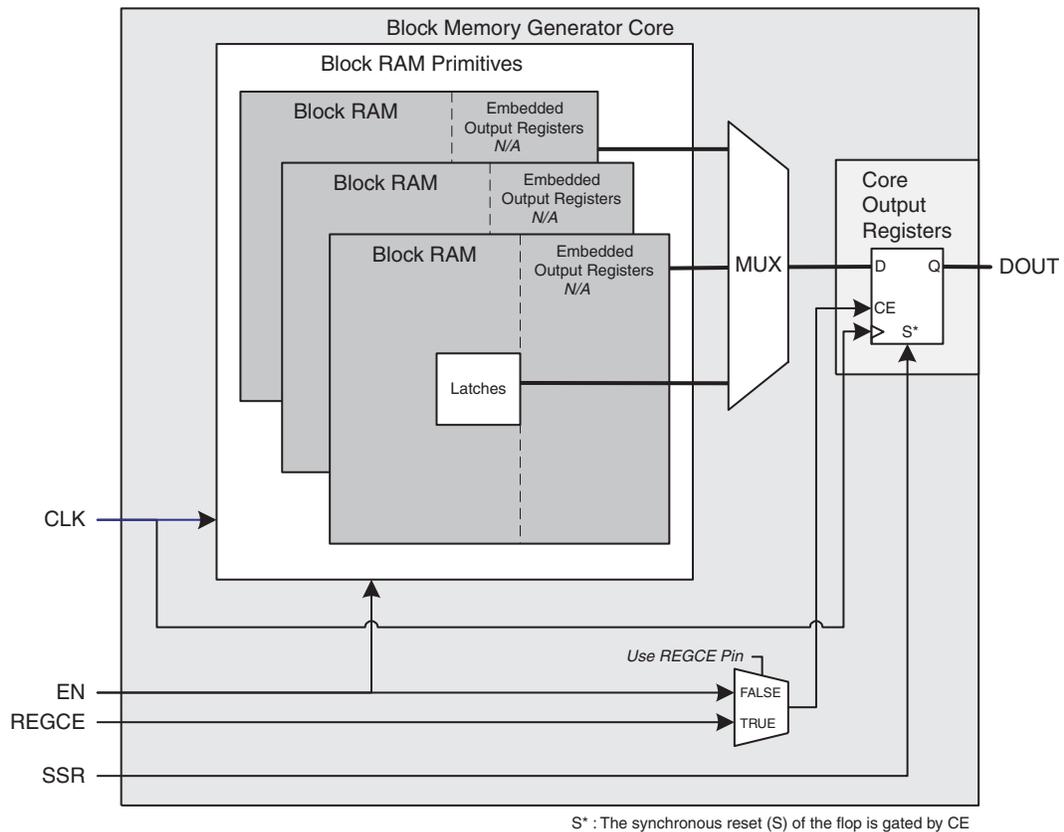


Figure 34: Virtex-4 or Virtex-5 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Core Enabled

**Virtex-5 and Virtex-4 FPGA: Memory with No Output Registers**

If neither of the output registers is selected for ports A or B, output of the memory primitives is driven directly from the RAM primitive latches. In this configuration, as shown in Figure 35, there are no additional clock cycles of latency, but the clock-to-out delay for a read operation can impact design performance.

- |  |           |  |
|--|-----------|--|
| <b>Port A</b>  | <b>or</b> | <b>Port B</b>  |
| <input type="checkbox"/> Register Port A Output of Memory Primitives |           | <input type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core       |           | <input type="checkbox"/> Register Port B Output of Memory Core       |

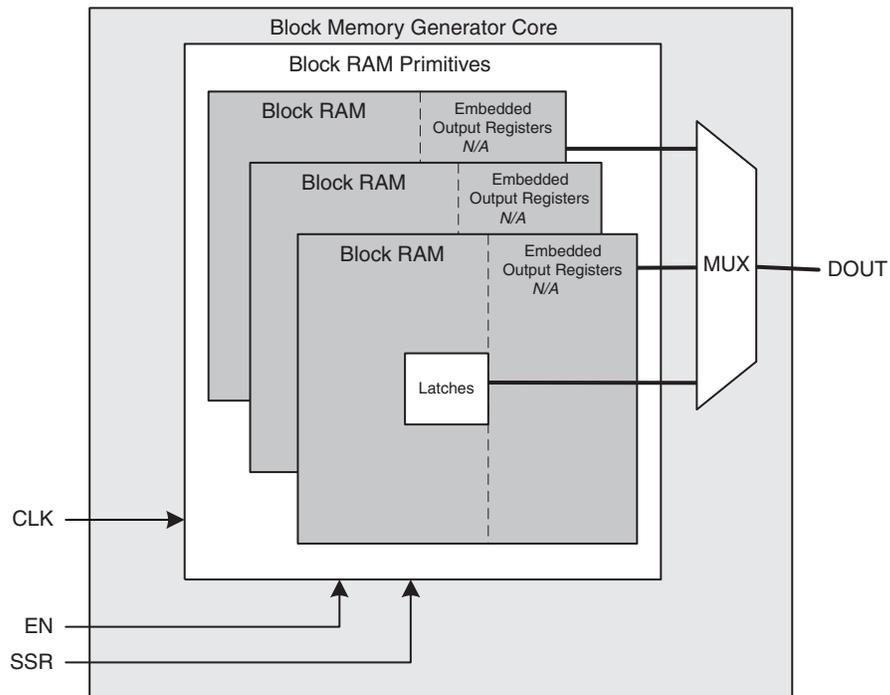


Figure 35: Virtex-4 or Virtex-5 Block Memory Generated with No Output Registers Enabled

## Spartan-3A DSP FPGA: Output Register Configurations

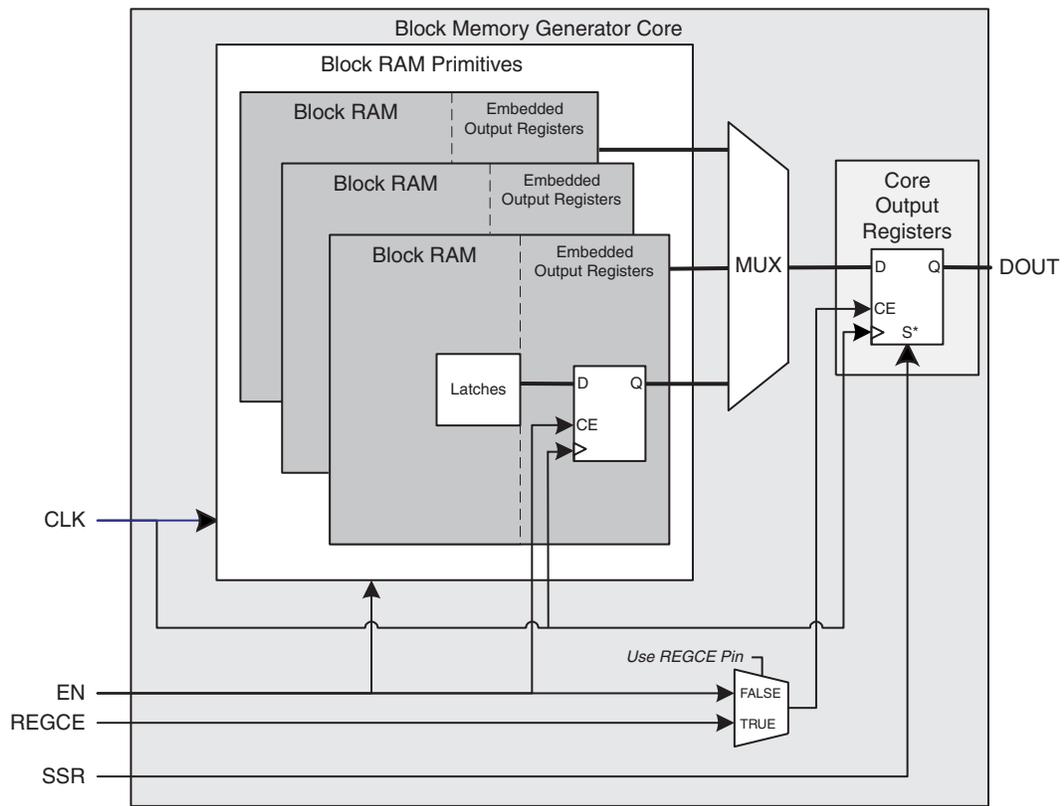
To tailor register options for Spartan-3A DSP device configurations, two selections each, for port A and port B, are provided on screen 4 of the CORE Generator GUI in the Optional Output Registers section. The embedded output registers for the corresponding port(s) are enabled when Register Port [A | B] Output of Memory Primitives is selected. Similarly, registers at the output of the core for the corresponding port(s) are enabled by selecting Register Port [A | B] Output of Memory Core. [Figures 36 through 41](#) illustrate the Spartan-3A DSP output register configurations.

When only Register Port [A | B] Output of Memory Primitives and the corresponding Use SSR[A | B] Pin (set/reset pin) is selected, the User Reset Behavior from RAMB16BWER Primitive becomes available. Selecting this option forces the core to use the Spartan-3A DSP embedded output registers, but changes the behavior of the core. For detailed information, see the sections that follow.

**Spartan-3A DSP FPGA: Memory with Primitive and Core Output Registers**

With both Register Port [A | B] Output of Memory Primitives and the corresponding Register Port [A | B] Output of Memory Core selected, a memory core is generated with both the Spartan-3A DSP embedded output registers and a register on the output of the core for the selected port(s), as shown in Figure 36. This configuration may improve performance when building a large memory construct.

- |   |           |   |
|---|-----------|---|
| <b>Port A</b>   | <b>or</b> | <b>Port B</b>   |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |           | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Core       |           | <input checked="" type="checkbox"/> Register Port B Output of Memory Core       |



S\* : The synchronous reset (S) of the flop is gated by CE

**Figure 36: Spartan-3A DSP Block Memory Generated with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Enabled**

### Spartan-3A DSP FPGA: Memory With Primitive Output Registers – Without SSR Pin

When Register Port [A | B] Output of Memory Primitives is selected, and the corresponding Use SSR Pin (set/reset pin) is not selected, a memory core that registers the output of the block RAM primitives for the selected port using the output registers embedded in Spartan-3A DSP FPGA architecture is generated. The output of any multiplexing that may be required to combine multiple primitives is not registered in this configuration (Figure 37).

- | Port A  | or | Port B  |
|---|----|---|
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core                  |    | <input type="checkbox"/> Register Port B Output of Memory Core                  |
| <input type="checkbox"/> Use SSRA Pin (set/reset pin)                           |    | <input type="checkbox"/> Use SSRB Pin (set/reset pin)                           |

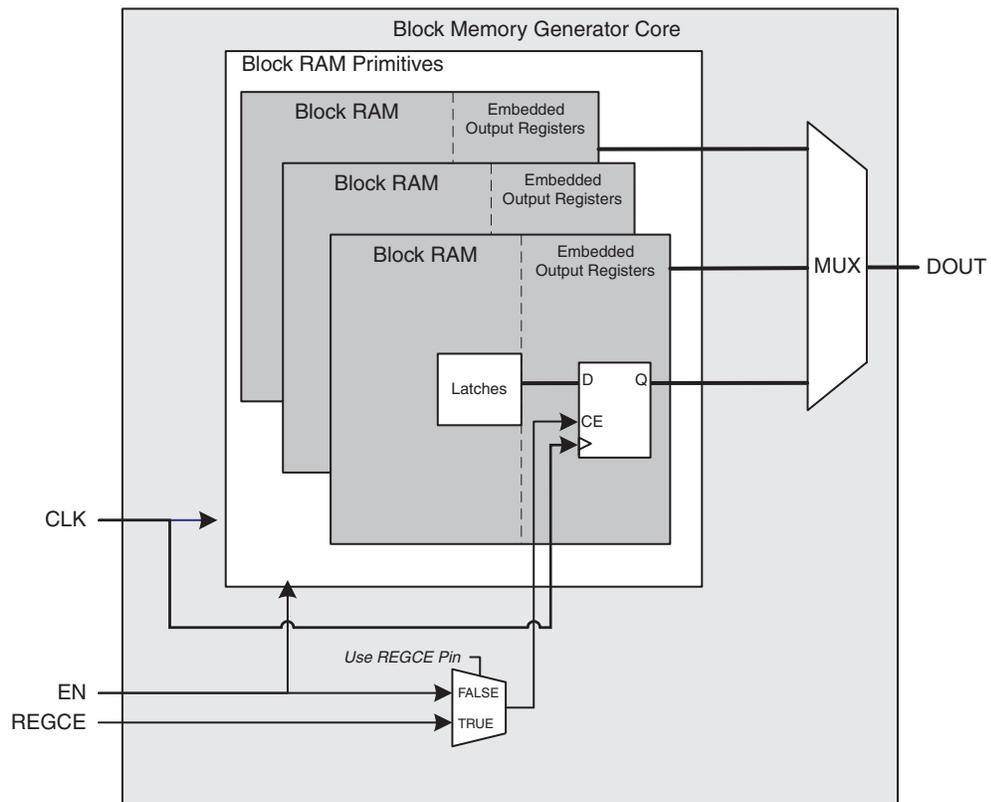


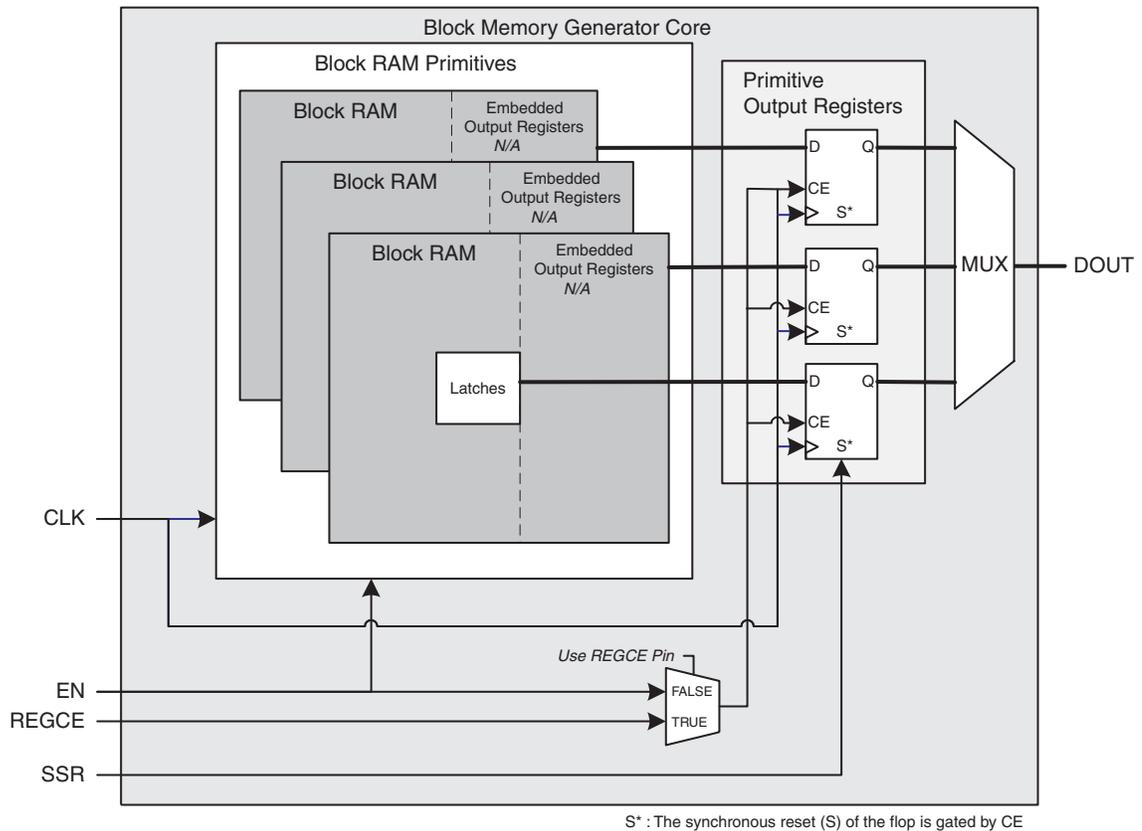
Figure 37: Spartan-3A DSP Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled (No SSR)

**Spartan-3A DSP FPGA: Memory with Primitive Output Registers, SSR – Without RAMB16BWER Reset Behavior**

If Use SSRA Pin (set/reset pin) or Use SSRB Pin (set/reset pin) is selected, and Use Reset Behavior from RAMB16BWER Primitive is *not* selected, the Spartan-3A DSP embedded block RAM registers cannot be used. The primitive output registers are built from FPGA fabric, as illustrated in Figure 38.

**Note:** This behavior is the same as that of Spartan-3A, Virtex-5, Virtex-4, and Virtex-II devices.

- | Port A  | or | Port B  |
|---|----|---|
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core                  |    | <input type="checkbox"/> Register Port B Output of Memory Core                  |
| <input checked="" type="checkbox"/> Use SSRA Pin (set/reset pin)                |    | <input checked="" type="checkbox"/> Use SSRB Pin (set/reset pin)                |
| <input type="checkbox"/> Use Reset Behavior from RAMB16BWER Primitive           |    | <input type="checkbox"/> Use Reset Behavior from RAMB16BWER Primitive           |



**Figure 38: Spartan-3A DSP Block Memory Generated with Register Port [AIB] Output of Memory Primitives, Use SSR[AIB] Pin Options (With SSR), without RAMB16BWER Reset Behavior**

### Spartan-3A DSP FPGA: Memory with Primitive Output Registers, SSR, and RAMB16BWER Reset Behavior

When Register Port [A | B] Output of Memory Primitives, Use SSRA Pin (set/reset pin) or Use SSRB Pin (set/reset pin), and Use Reset Behavior from RAMB16BWER Primitive are selected, the Spartan-3A DSP embedded registers are enabled for the selected port in the generated core, as displayed in Figure 39.

If Use Reset Behavior from RAMB16BWER Primitive is selected, the Spartan-3A DSP’s embedded output registers are used, but the reset behavior of the core changes. Figure 39 illustrates how the memory is constructed in this case. The functional differences between this and other implementations are that the SSR[A | B] input resets *both* the embedded output registers *and* the block RAM output latches. If EN and REGCE are held high, the output value is set to the reset value for two clock cycles following a reset. In addition, the synchronous reset for both the latches and the embedded output registers are gated by the EN input to the core, independent of the state of REGCE. This differs from all other configurations of the Block Memory Generator where SSR is typically gated by REGCE.

- | Port A   | or | Port B   |
|--|----|--|
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives  |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives  |
| <input type="checkbox"/> Register Port A Output of Memory Core                   |    | <input type="checkbox"/> Register Port B Output of Memory Core                   |
| <input checked="" type="checkbox"/> Use SSRA Pin (set/reset pin)                 |    | <input checked="" type="checkbox"/> Use SSRB Pin (set/reset pin)                 |
| <input checked="" type="checkbox"/> Use Reset Behavior from RAMB16BWER Primitive |    | <input checked="" type="checkbox"/> Use Reset Behavior from RAMB16BWER Primitive |

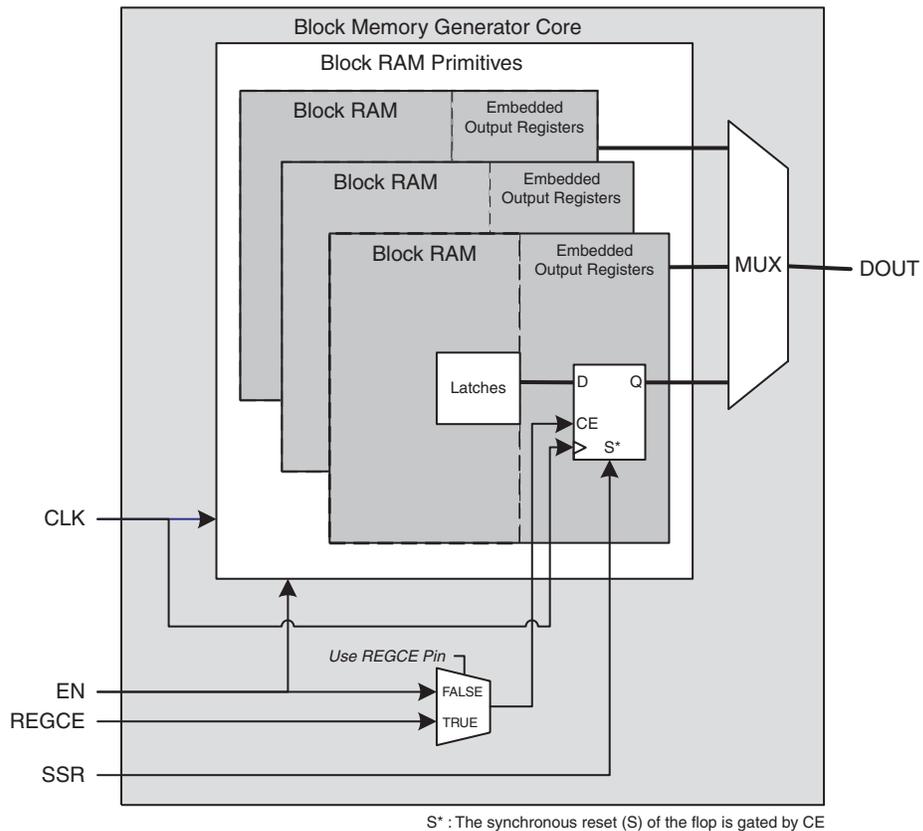
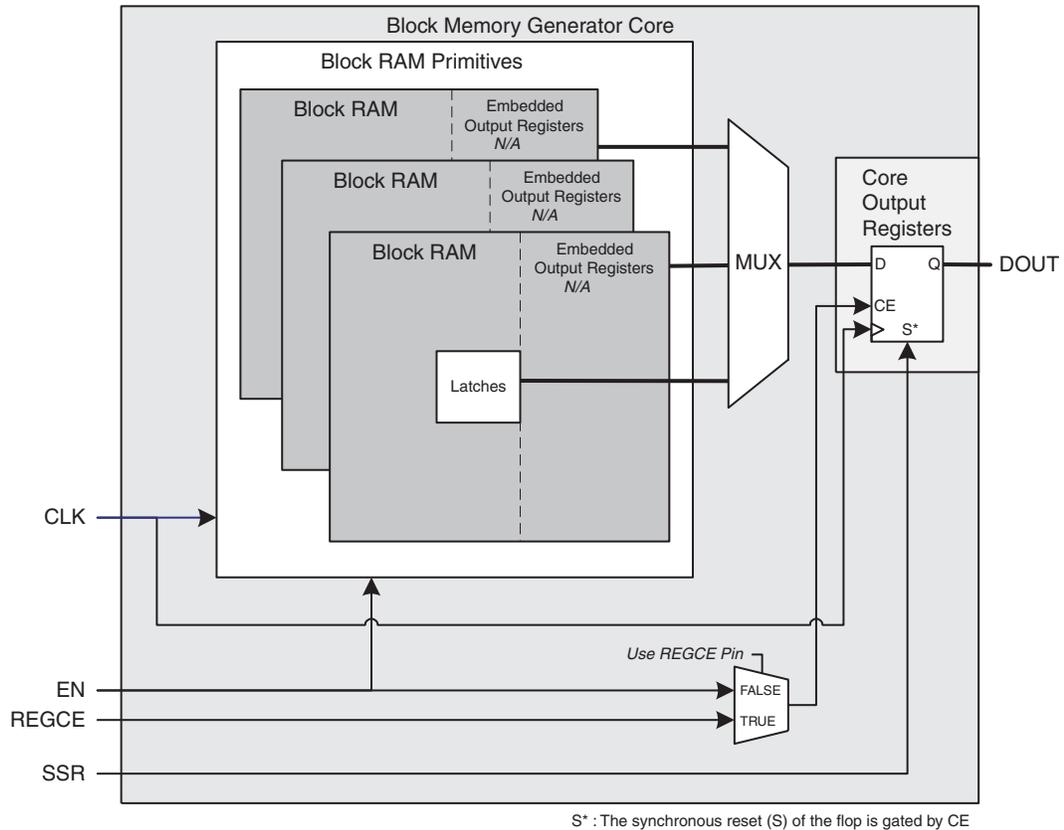


Figure 39: Spartan-3A DSP Block Memory Generated with Register Port [A|B] Output of Memory Primitives, Use SSR[A|B] Pin Options (With SSR), and RAMB16BWER Reset Behavior Enabled

**Spartan-3A DSP FPGA: Memory with Core Output Registers**

When Register Port [A | B] Output of Memory Core is selected, the Spartan-3A DSP FPGA embedded registers are disabled in the generated core, as illustrated in **Figure 40**.

- |   |           |   |
|---|-----------|---|
| <b>Port A</b>   | <b>or</b> | <b>Port B</b>   |
| <input type="checkbox"/> Register Port A Output of Memory Primitives      |           | <input type="checkbox"/> Register Port B Output of Memory Primitives      |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Core |           | <input checked="" type="checkbox"/> Register Port B Output of Memory Core |
| <input checked="" type="checkbox"/> Use SSRA Pin (set/reset pin)          |           | <input checked="" type="checkbox"/> Use SSRB Pin (set/reset pin)          |



**Figure 40: Spartan-3A DSP Block Memory Generated with Register Port [AIB] Output of Memory Core Enabled**

### Spartan-3A DSP FPGA: Memory with No Output Registers

If no output registers are selected for port A or B, output of the memory primitive is driven directly from the RAM primitive latches. In this configuration, as shown in Figure 41, there are no additional clock cycles of latency, but the clock-to-out delay for a read operation can impact design performance.

- |  |           |  |
|--|-----------|--|
| <b>Port A</b>  | <b>or</b> | <b>Port B</b>  |
| <input type="checkbox"/> Register Port A Output of Memory Primitives |           | <input type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core       |           | <input type="checkbox"/> Register Port B Output of Memory Core       |

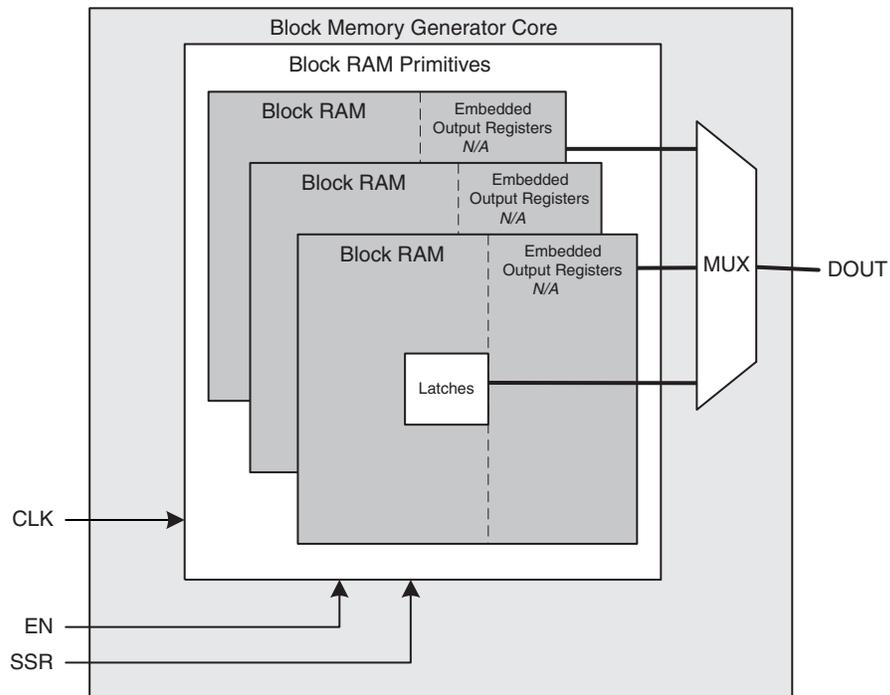


Figure 41: Spartan-3A DSP Block Memory Generated with No Output Port Registers Enabled

### Virtex-II FPGA: Output Register Configurations

To tailor register options for Virtex-II FPGA architectures, two selections each, for port A and port B, are provided in the CORE Generator GUI on screen 4 in the Optional Output Registers section. For implementing registers on the outputs of the individual block RAM primitives, Register Output of Memory Primitives is selected. In the same way, registering the output of the core is enabled by selecting Register Port [A | B] Output of Memory Core. Four implementations are available for each port. Figures 42 through 45 illustrate the Virtex-II FPGA output register configurations.

### Virtex-II FPGA: Memory with Primitive and Core Output Registers

With Register Port [A | B] Output of Memory Primitives and the corresponding Register Port [A | B] Output of Memory Core selected, a memory core is generated with registers on the outputs of the individual RAM primitives and on the core output, as displayed in Figure 42. Selecting this configuration may provide improved performance for building large memory constructs.

- | Port A  | or | Port B  |
|---|----|---|
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Core       |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Core       |

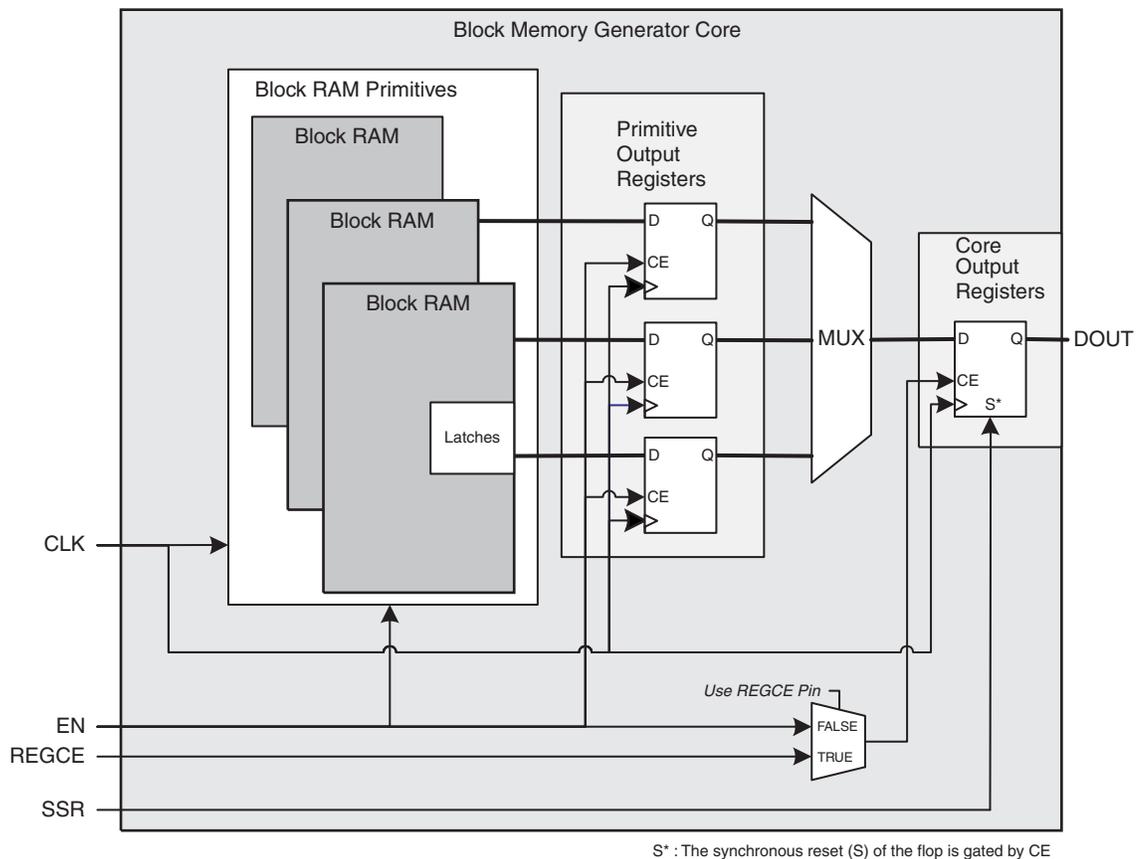
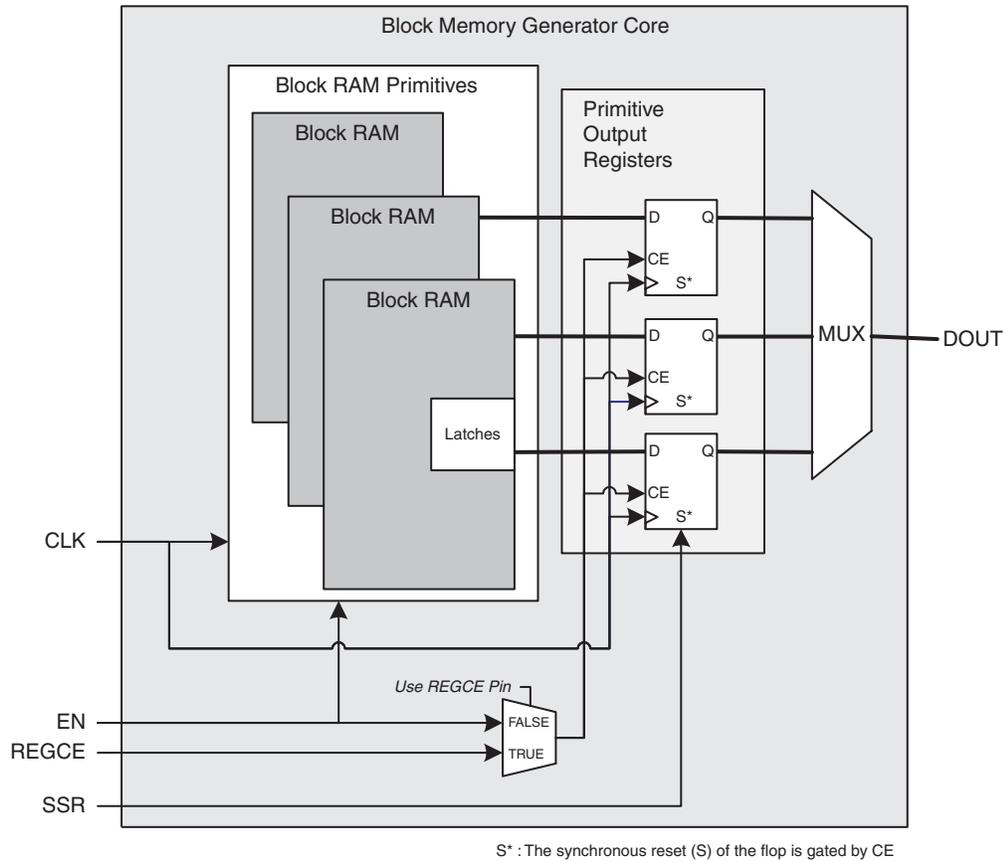


Figure 42: Virtex-II Block Memory Generated with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Options Enabled

**Virtex-II FPGA: Memory with Primitive Output Registers**

When Register Port [A | B] Output of Memory Primitives is selected, a core that only registers the output of the RAM primitives is generated. Note that the output of any multiplexing required to combine multiple primitives are not registered in this configuration, as shown in Figure 43.

- |   |    |   |
|---|----|---|
| <b>Port A</b>   | or | <b>Port B</b>   |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Primitives |    | <input checked="" type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core                  |    | <input type="checkbox"/> Register Port B Output of Memory Core                  |



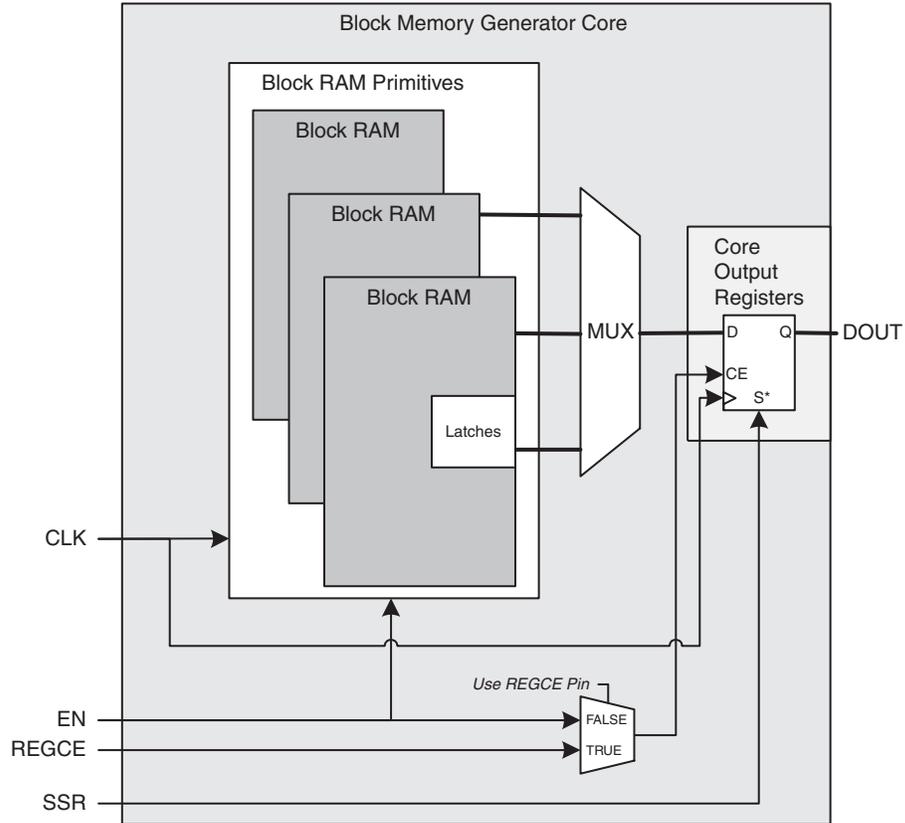
S\* : The synchronous reset (S) of the flop is gated by CE

**Figure 43: Virtex-II Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled**

**Virtex-II FPGA: Memory with Core Output Registers**

Figure 44 illustrates a memory configured with Register Port [A | B] Output of Memory Core selected.

- |   |           |   |
|---|-----------|---|
| <b>Port A</b>   | <b>or</b> | <b>Port B</b>   |
| <input type="checkbox"/> Register Port A Output of Memory Primitives      |           | <input type="checkbox"/> Register Port B Output of Memory Primitives      |
| <input checked="" type="checkbox"/> Register Port A Output of Memory Core |           | <input checked="" type="checkbox"/> Register Port B Output of Memory Core |



S\* : The synchronous reset (S) of the flop is gated by CE

Figure 44: Virtex-II Block Memory Generated with Register Port [A|B] Output of Memory Core Enabled

### Virtex-II FPGA: Memory with No Output Registers

When no output register options are selected for either port A or port B, the output of the memory primitive is driven directly from the memory latches. In this configuration, there are no additional clock cycles of latency, but the clock-to-out delay for a read operation can impact design performance. See [Figure 45](#).

- |  |           |  |
|--|-----------|--|
| <b>Port A</b>  | <b>or</b> | <b>Port B</b>  |
| <input type="checkbox"/> Register Port A Output of Memory Primitives |           | <input type="checkbox"/> Register Port B Output of Memory Primitives |
| <input type="checkbox"/> Register Port A Output of Memory Core       |           | <input type="checkbox"/> Register Port B Output of Memory Core       |

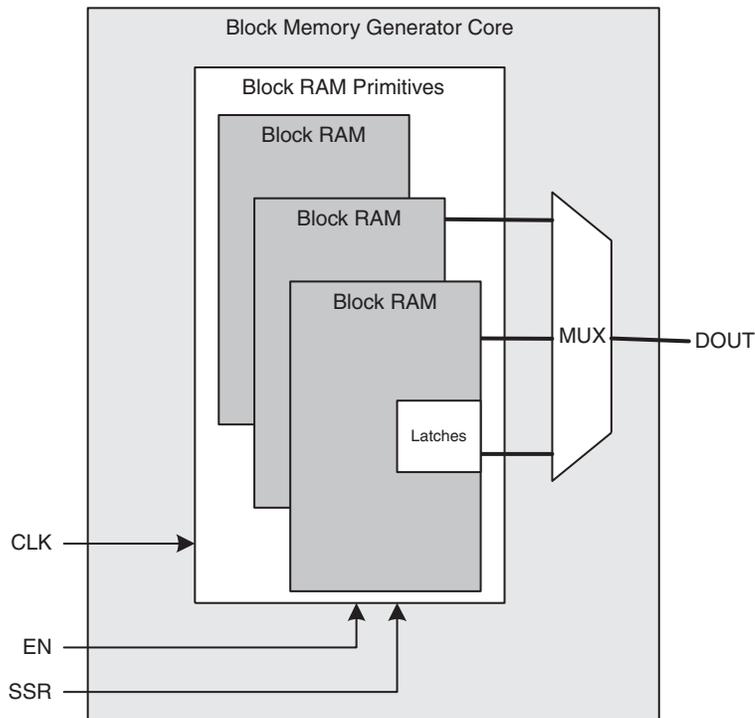


Figure 45: Virtex-II Block Memory Generated with No Output Registers Enabled

## Verification

The Block Memory Generator core and the simulation models delivered with it are rigorously verified using advanced verification techniques, including a constrained random configuration generator and a cycle-accurate bus functional model.

## Resource Utilization and Performance

The resource utilization and performance of the Block Memory Generator core is highly dependent on user selections, such as algorithm, optional output registers, and memory size.

## Ordering Information

The Block Memory Generator core is free of charge and is included with the Xilinx ISE CORE Generator system. Updates to the core are bundled with ISE IP Updates, which are accessible from the [Xilinx Download Center](#). To order Xilinx software, please contact your local Xilinx sales representative. Information about additional Xilinx LogiCORE IP modules is available on the [Xilinx IP Center](#).

## Related Information

Xilinx products are not intended for use in life-support appliances, devices, or systems. Use of a Xilinx product in such application without the written consent of the appropriate Xilinx officer is prohibited.

## Revision History

Date	Version	Revision
1/11/06	1.0	Initial Xilinx release
4/12/06	2.0	Updated for Virtex-5 support
7/13/06	3.0	Updated primitives information in Table 1, replaced GUI screens, ISE version, release date.
9/21/06	4.0	Minor updates for v2.2 release
11/15/06	4.5	Updated for the v2.3 release
2/15/07	5.0	Updated for v2.4 release, added support for ECC.
4/02/07	5.5	Added support for Spartan-3A DSP devices.
8/08/07	6.0	Updated core to v2.5; Xilinx tools v9.2i.
10/10/07	6.5	Updated core to v2.6.
3/24/08	7.0	Updated core to version 2.7; ISE tools 10.1.
9/19/08	8.0	Updated core to version 2.8.

## Notice of Disclaimer

Xilinx is providing this information (collectively, the "Information") to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.