Block Memory Generator v3.3

DS512 September 16, 2009 **Product Specification**

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 and memory depths from 2 to 9M words (limited only by memory resources on selected part)
- Supports configurable port aspect ratios for dual-port configurations and read-to-write aspect ratios in Virtex®-6, Virtex-5 and Virtex-4 FPGAs
- Optimized algorithms for minimum block RAM resource utilization or low power utilization
- Configurable memory initialization
- Supports individual write enable per byte in Virtex-6, Virtex-5, Virtex-4, Spartan®-6, and Spartan-3A/XA DSP 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) for Virtex-6 and Virtex-5 devices, and associated error injection pins in Virtex-6 to insert single and double bit errors
- Supports pipelining of DOUT bus for improved performance in specific configurations
- Smaller primitive configurations in Spartan-6 devices with the introduction of new 9K primitives.
- Lower data widths for Virtex-6 devices in SDP mode.
- Choice of reset priority for output registers between priority of SR (Set Reset) or CE (Clock Enable) in Spartan-6 and Virtex-6 families.
- Asynchronous reset in Spartan-6 devices.

LogiCORE™ IP Facts	
Core Specifics	
Supported Device Family(1)	Virtex-6, Virtex-5, Virtex-4, Spartan-6, Spartan-3E/XA, Spartan-3/XA, Spartan-3A/3AN/3A DSP
Block RAM	Varied, based on core parameters
DCM	None
BUFG	None
IOBs/ Transceivers	None
PPC	None
IOB-FF/TBUFs	None
Provided with Core	
Documentation	Product Specification Migration Guide(2)
Design File Formats	NGC Netlist
Design Tool Requirements	
Xilinx Implementation Tools	ISE® v11.3
Simulation	Mentor Graphics® ModelSim®: v6.4b and above VHDL Structural Verilog Structural VHDL Behavioral(3) Verilog Behavioral(3)
Synthesis	XST
Support	
Provided by Xilinx, Inc.	

1. See Table 1, page 2 for more details about supported devices.
2. The Migration Guide provides instructions for converting designs that contain instances of either Legacy LogiCORE IP 6.x Single or Dual Port Block Memory, or older versions of LogiCORE Block Memory Generator to blocks of the latest version of the LogiCORE Block Memory Generator.
3. Behavioral models do not precisely model collision behavior. See "Simulation Models," page 31 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-6, 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," page 52.

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

Supported Devices

Table 1 shows the families and sub-families supported by the Block Memory Generator.

Table 1: **Supported FPGA Families and Sub-Families**

FPGA Family	Sub-Family
Spartan-3	
XA Spartan-3	
Spartan-3E	
XA Spartan-3E	
Spartan-3A	
XA Spartan-3A	
Spartan-3AN	
XA Spartan-3AN	
Spartan-3A DSP	
XA Spartan-3A DSP	
Spartan-6	LX/LXT
Virtex-4	LX/FX/SX
Virtex-4 QPro	LX/FX/SX

Table 1: **Supported FPGA Families and Sub-Families *(Cont'd)***

FPGA Family	Sub-Family
Virtex-5	LXT/FXT/SXT/TXT
Virtex-5 QPro	LXT/FXT/SXT
Virtex-6	CXT/HXT/LXT/SXT
Virtex-6 -1L	LXT/SXT

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," page 16 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-6, 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-6, Virtex-5, Virtex-4, Spartan-6, 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-6, Virtex-5, Virtex-4, Spartan-6, and Spartan-3A DSP embedded block RAM registers as well as registers implemented in the FPGA fabric. See "Output Register Configurations," page 57 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," 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 Set/Reset Pin

The core provides optional set/reset pins (RSTA and RSTB) for each port that 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

Simple Dual-port RAM memories support the FPGA Hamming Error Correction Capability (ECC) of the Virtex-6 and 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. Figure 1 through Figure 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.

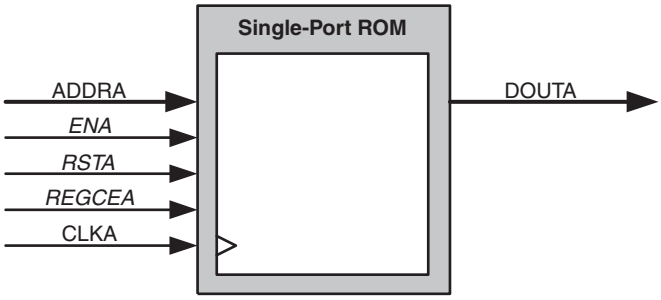

Figure 1: **Single-port ROM**

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

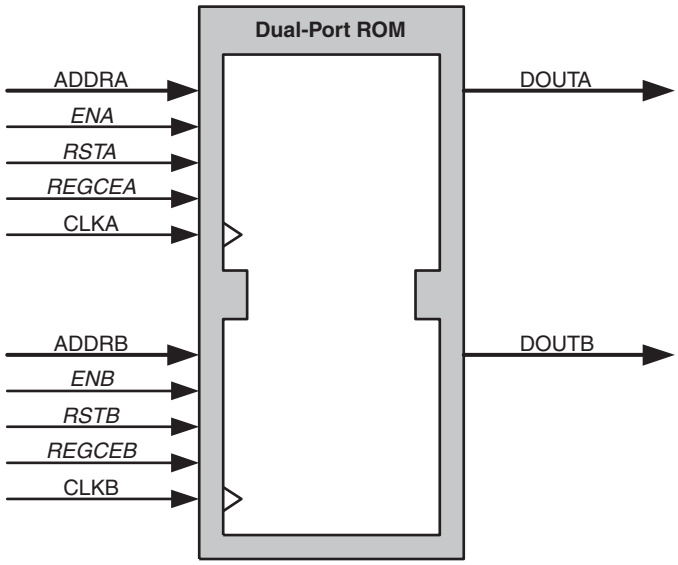

Figure 2: **Dual-port ROM**

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

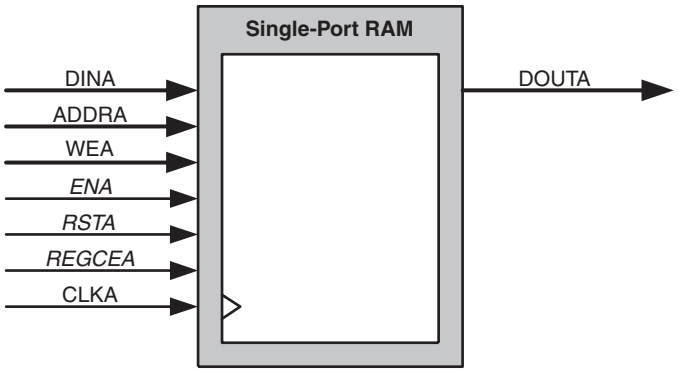

Figure 3: **Single-port RAM**

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.

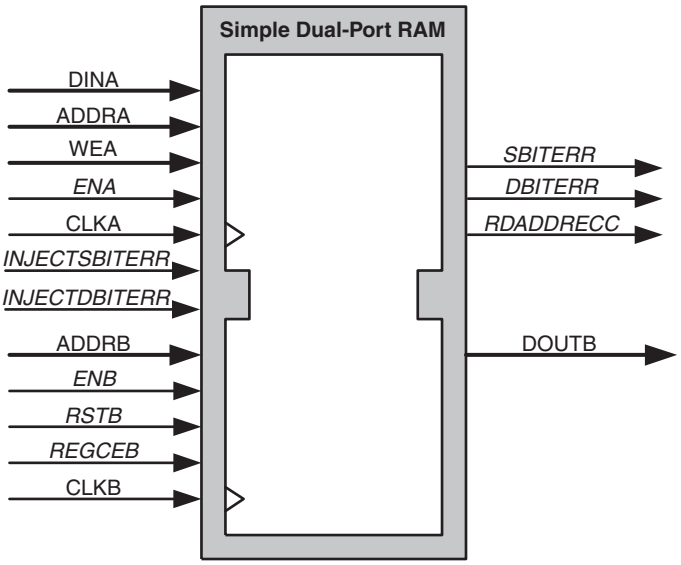

Figure 4: **Simple Dual-port RAM**

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.

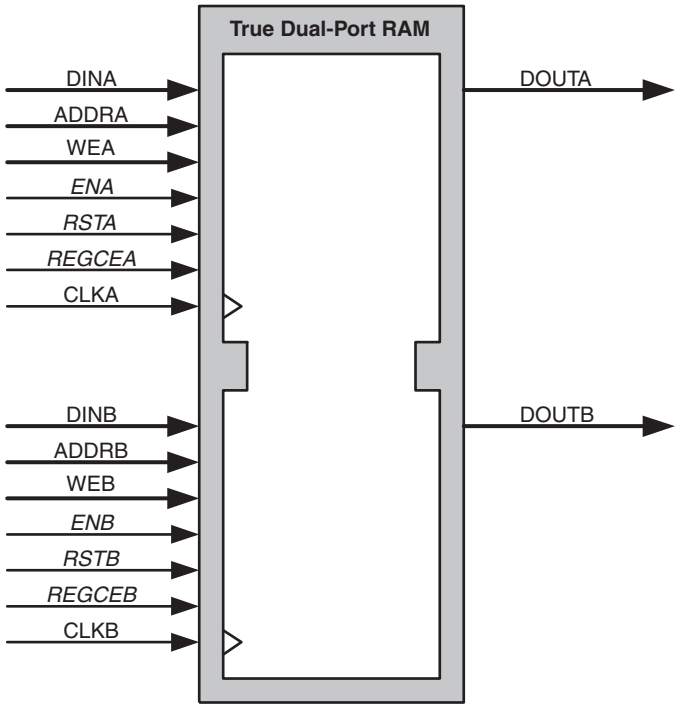

Figure 5: **True Dual-port RAM**

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.

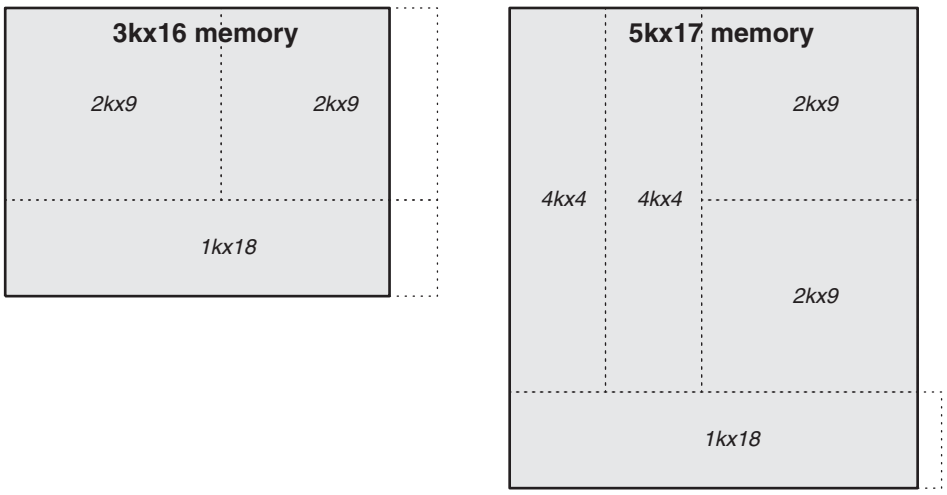

Figure 6: **Examples of the Minimum Area Algorithm**

Note: In Spartan-6 devices, two 9K block RAMs are used for one 1Kx18.

In 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.

In 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.

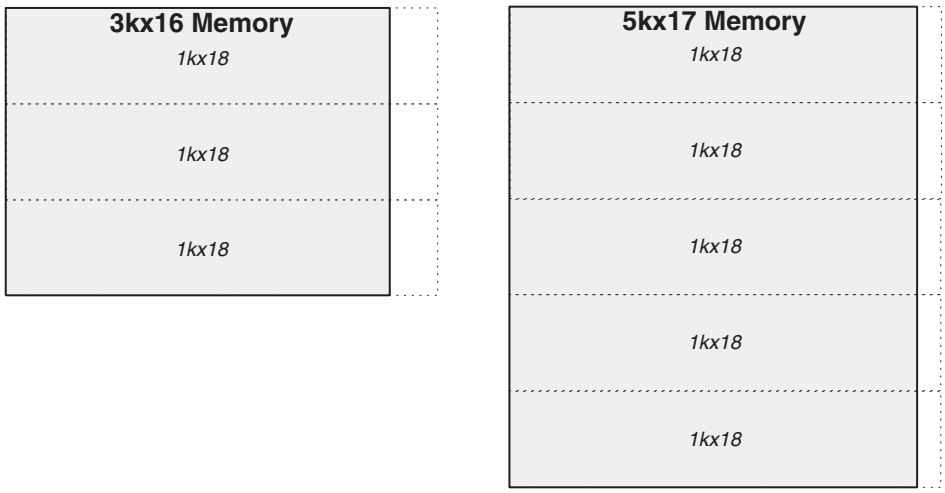

Figure 7: **Examples of the Low Power Algorithm**

Note: In Spartan-6 devices, two 9K block RAMs are used for one 1Kx18.

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.

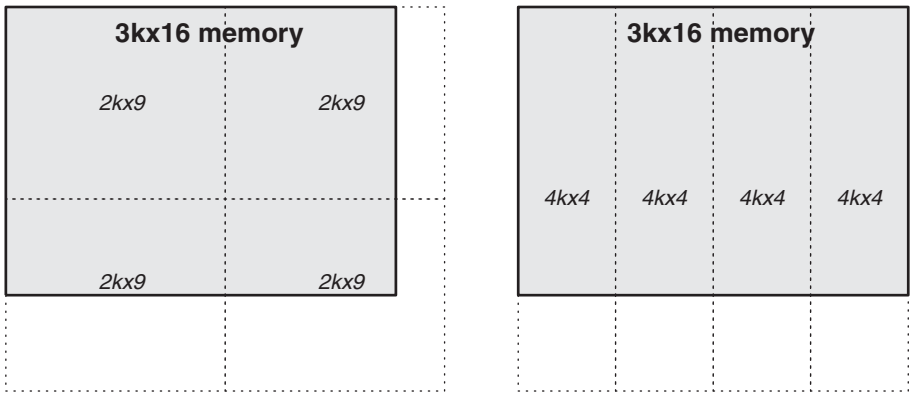

Figure 8: **Examples of the Fixed Primitive Algorithm**

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, 256x72 and 256x36 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 mem-

ory structures to enhance performance. Table 2 shows the primitives used to construct a memory given the specified architecture and primitive selection.

Table 2: **Memory Primitives Used Based on Architecture**

Architecture	Primitive Selection	Primitives Used
Spartan-6 FPGA	16kx1	8kx1,16kx1
Spartan-6 FPGA	8kx2	4kx2,8kx2
Spartan-6 FPGA	4kx4	2kx4,4kx4
Spartan-6 FPGA	2kx9	1kx9,2kx9
Spartan-6 FPGA	1kx18	512x18,1kx18
Spartan-6 FPGA	512x36	512x36
Spartan-6 FPGA	256x72	256x72 (SP RAM/ROM configurations only)
Spartan-6 FPGA	256x36	256x36 (SP RAM/ROM and SDP configurations only)
Spartan-3(1) FPGA	16kx1	16kx1
Spartan-3(1) FPGA	8kx2	8kx2
Spartan-3(1) FPGA	4kx4	4kx4
Spartan-3(1) FPGA	2kx9	2kx9
Spartan-3(1) FPGA	1kx18	1kx18
Spartan-3(1) FPGA	512x36	512x36
Spartan-3(1) FPGA	256x72	256x72 (Single Port configurations only)
Virtex-6 FPGA	16kx1	64x1, 32kx1, 16kx1
Virtex-6 FPGA	8kx2	16kx2, 8kx2
Virtex-6 FPGA	4kx4	4kx4, 8kx4
Virtex-6 FPGA	2kx9	2kx9, 4kx9
Virtex-6 FPGA	1kx18	1kx18, 2kx18
Virtex-6 FPGA	512x36	512x36 (SP RAM/ROM and SDP configurations only), 1kx36
Virtex-6 FPGA	256x72	512x72 (SP RAM/ROM and SDP configurations only)
Virtex-5 FPGA	16kx1	64kx1, 32kx1, 16kx1
Virtex-5 FPGA	8kx2	16kx2, 8kx2
Virtex-5 FPGA	4kx4	8kx4, 4kx4
Virtex-5 FPGA	2kx9	4kx9, 2kx9
Virtex-5 FPGA	1kx18	2kx18, 1kx18
Virtex-5 FPGA	512x36	1kx36
Virtex-5 FPGA	256x72	512x72 (Single and Simple Dual-port RAMs and Single Port ROMs only)
Virtex-4 FPGA	16kx1	32kx1, 16kx1
Virtex-4 FPGA	8kx2	8kx2
Virtex-4 FPGA	4kx4	4kx4
Virtex-4 FPGA	2kx9	2kx9

Table 2: **Memory Primitives Used Based on Architecture *(Cont'd)***

Architecture	Primitive Selection	Primitives Used
Virtex-4 FPGA	1kx18	1kx18
Virtex-4 FPGA	512x36	512x36
Virtex-4 FPGA	256x72	256x72 (Single Port configurations only)

1. 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," page 15. When using the byte write feature in Virtex-6, Virtex-5, Virtex-4, Spartan-6, 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 in the present or following clock cycles depending upon the output register stages of the core.

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," page 16. 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.

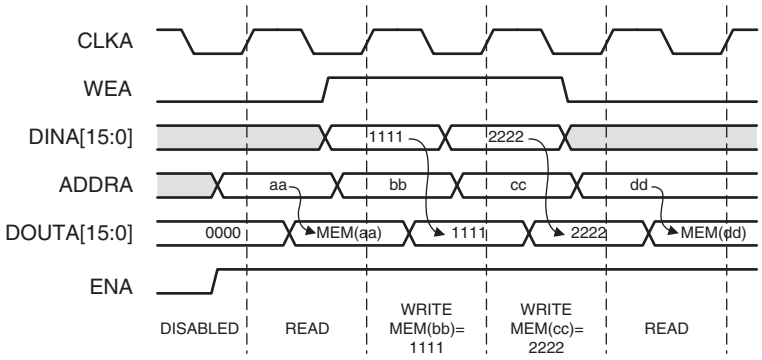

Figure 9: **Write First Mode Example**

Note: The WRITE_FIRST operation is affected by the optional byte-write feature in Virtex-6, Virtex-5, Virtex-4, Spartan-6 and Spartan-3A/3A DSP devices. It is also affected by the optional read-to-write aspect ratio feature in Virtex-6, Virtex-5 and Virtex-4 devices. For detailed information, see "Write First Mode Considerations," page 16.

- **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.

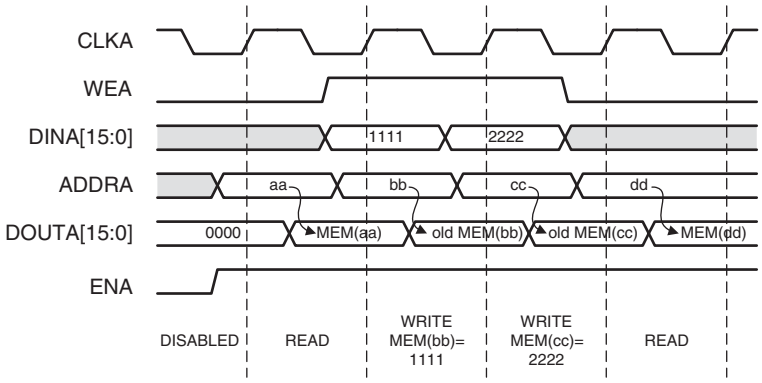

Figure 10: **Read First Mode Example**

- **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.

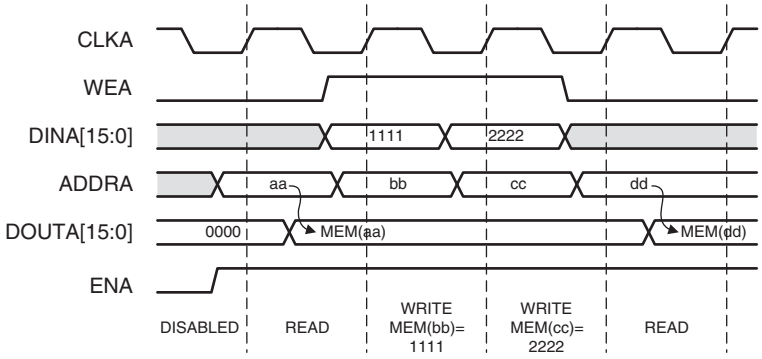

Figure 11: **No Change Mode Example**

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-6, Virtex-5 and Virtex-4 FPGA-based memories, all four data buses (DINA, DOUTA, DINB, and DOUTB) can have different widths, as described in "Virtex-6, Virtex-5 and Virtex-4 Read-to-Write Aspect Ratios," page 14.

The limitations of the data width aspect ratio feature (some of which are imposed by other optional features) are described in "Aspect Ratio Limitations," page 15. 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 ADDRA bus is 11 bits, while the ADDRB 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.

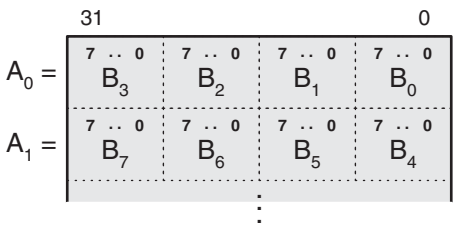

Figure 12: **Port Aspect Ratio Example Memory Map**

Virtex-6, Virtex-5 and Virtex-4 Read-to-Write Aspect Ratios

When implementing RAMs targeting Virtex-6, 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-6, 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 3 defines the four data-port widths and their respective depths for this example.

Table 3: **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 ADDRB width is determined by the larger of the B port depths (1024). For this reason, ADDRB is 10 bits wide. On port B, read operations utilize the entire ADDRB 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.

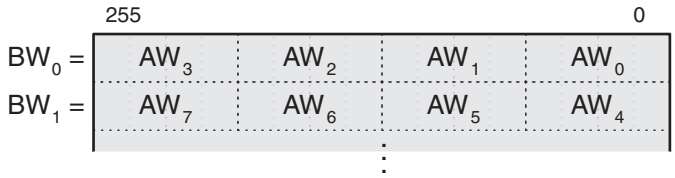

Figure 13: **Read-to-Write Aspect Ratio Example Memory Map**

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-6, Virtex-5, Virtex-4, Spartan-6, 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]` (`WEA` or `WEB`) 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," page 13. 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," page 16.

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.

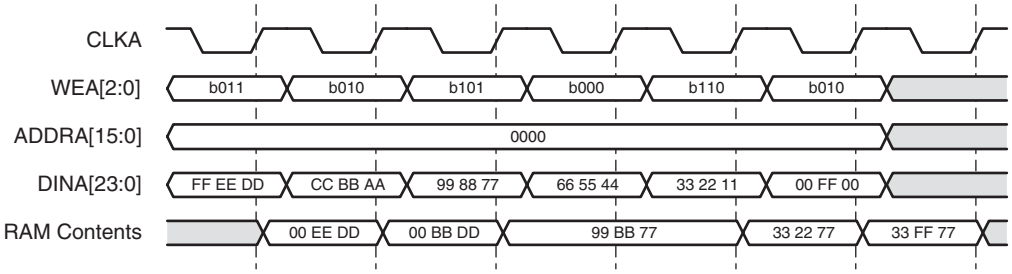

Figure 14: **Byte-write Example**

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-6, Virtex-5, Virtex-4, Spartan-6, and Spartan-3A/3A DSP devices or the read-to-write aspect ratio feature in Virtex-6, 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, is it 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 ADDRA = ADDRB = 0.

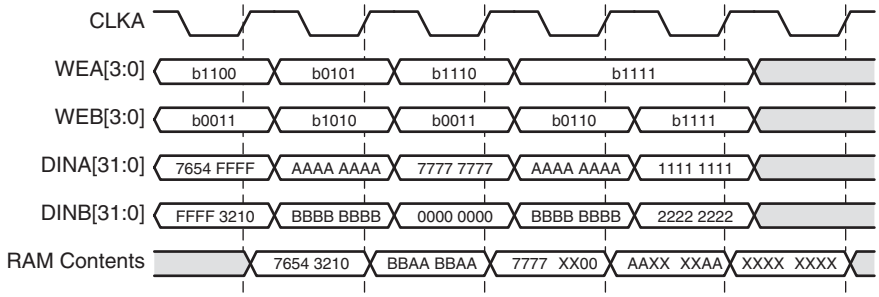

Figure 15: **Write-write Collision Example**

- 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 ADDRA = ADDRB = 0, port B is always reading, and all memory locations are initialized to 0. The RAM contents are never corrupted in write-read collisions.

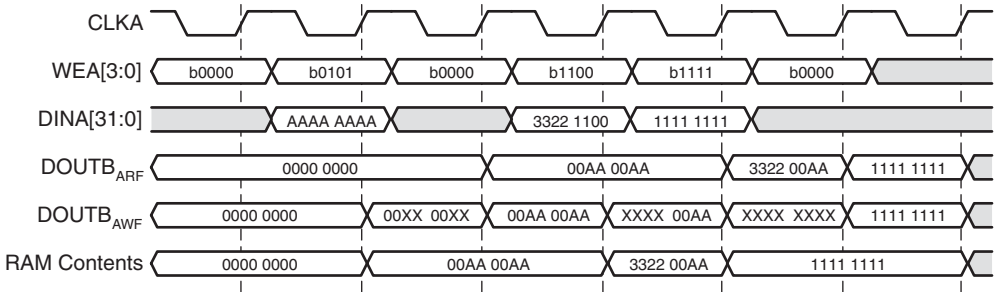

Figure 16: **Write-read Collision Example**

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," page 16. 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.

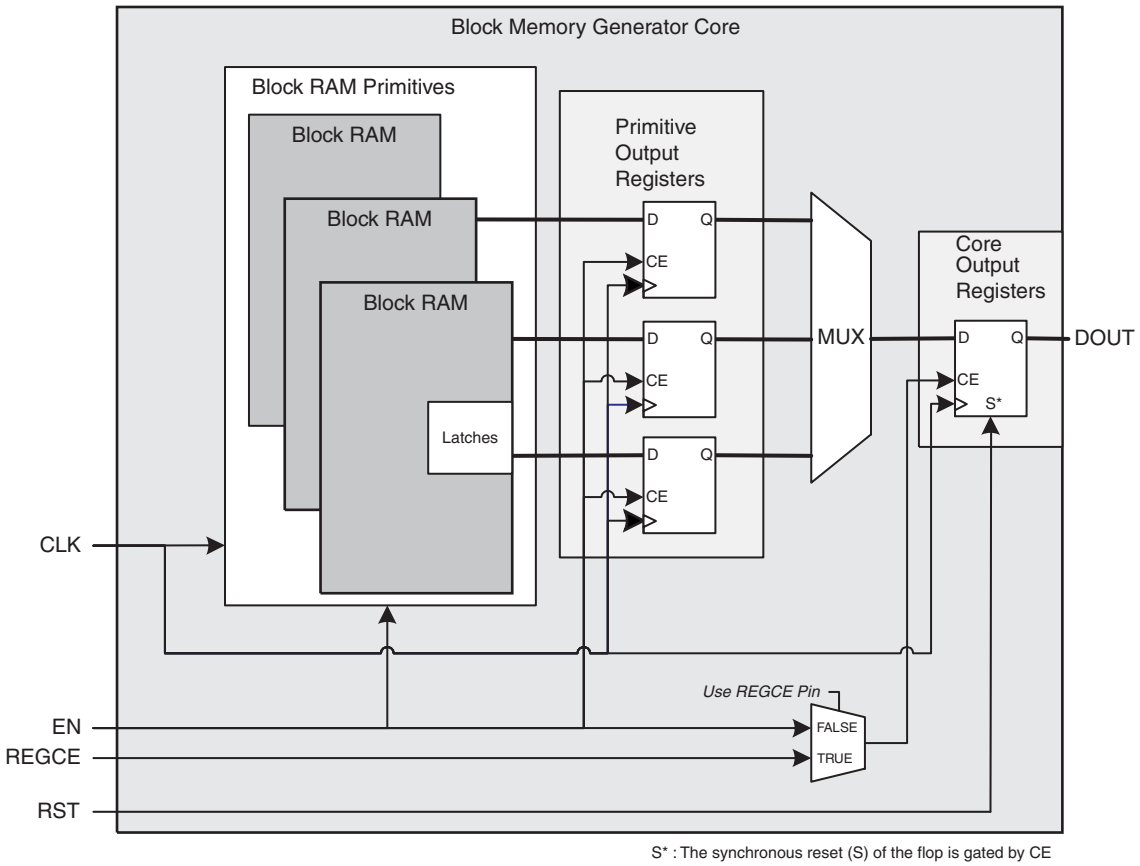

Figure 17: **Spartan-3 Block Memory: Register Port [A|B] Outputs of Memory Primitives and Memory Core Options Enabled**

For Virtex-6, Virtex-5, Virtex-4, Spartan-6, 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-6, Virtex-5 or Virtex-4 FPGA-based memory that has been configured using both output register stages for one of the ports.

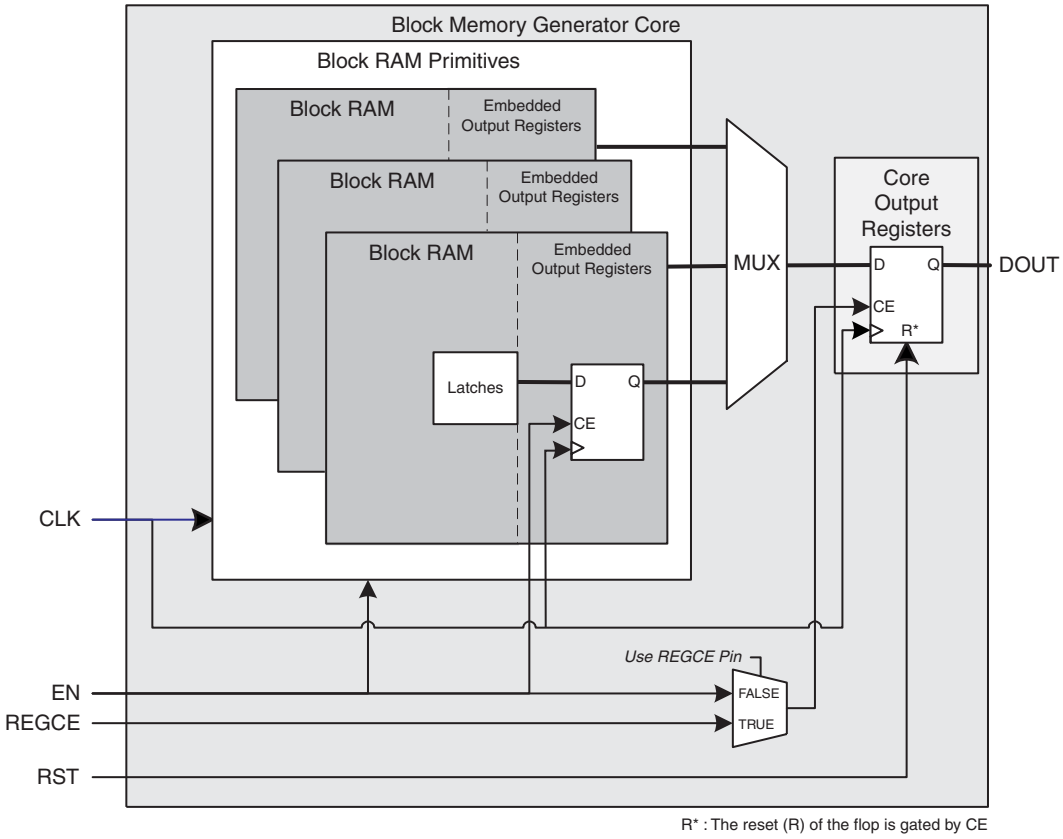

Figure 18: **Virtex-6, Virtex-5, and Virtex-4 Block Memory with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Options Enabled**

When using the Synchronous Reset Input (RST), 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-6, Virtex-5 and Virtex-4 FPGA configurations. To force the core to use the embedded output registers in Spartan-6 and Spartan-3A DSP devices, the Reset Behavior options are provided. For a complete description of the supported output options, see "Output Register Configurations," page 57.

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 "Optional Output Registers," page 37 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.

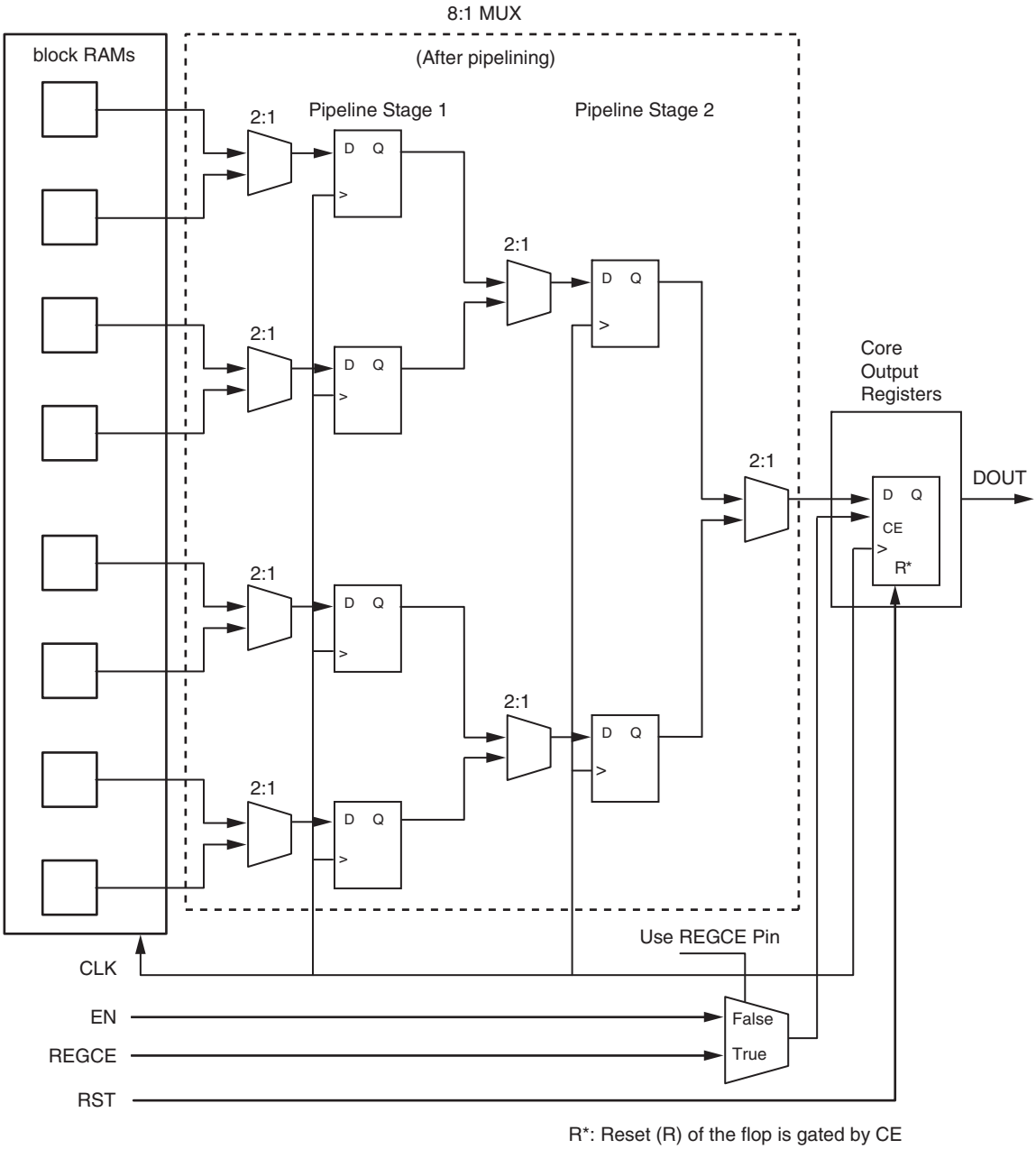

Figure 19: **Memory Configuration with 8:1 MUX and Two Pipeline Stages within the MUX**

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 independent of the EN signal.

Optional Set/Reset Pins

The set/reset pins (RSTA and RSTB) 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 RST 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 RST and EN when the Use REGCE Pin option is not selected.)

- For Virtex-4 FPGAs, if the option to use the 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 Virtex-6, Spartan-6, and Spartan-3A DSP FPGAs, the set/reset behavior differs when the reset behavior option is selected. However, this option saves resources by using the embedded output registers available in the Spartan-6 and Spartan-3A DSP primitives. See "Special Reset Behavior," page 23 for more information.

Memory Output Flow Control

The combination of the enable (EN), reset (RST), 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 RST and REGCE pins apply only to the last register stage.

Figure 20 depicts how RST 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.

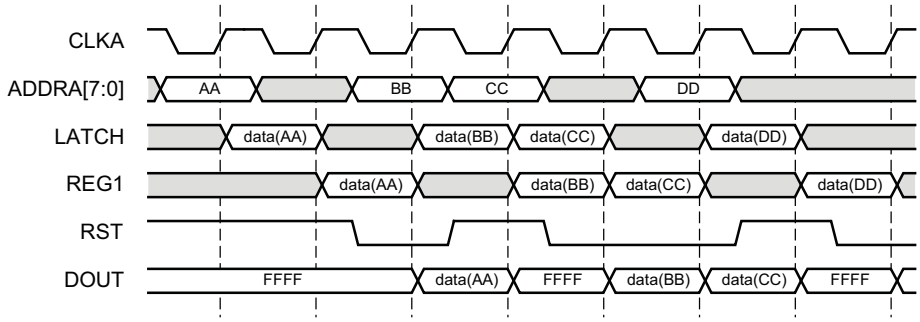

Figure 20: **Flow Control Using RST**

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 RST 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.

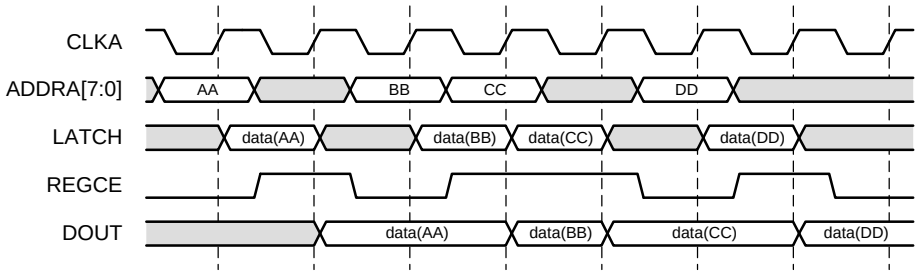

Figure 21: **Flow Control Using REGCE**

Reset Priority

For Spartan-6 and Virtex-6 devices, the Block Memory Generator core provides the option to choose the reset priority of the output stages of the Block Memory. In previous architectures such as Spartan-3A DSP and Virtex-5 devices, EN had a fixed priority over SSR (Synchronous Set Reset) for the memory latch, and REGCE (Register Clock Enable) had a fixed priority over SSR for embedded registers.

In Spartan-6 devices, when a user chooses the Reset Priority as CE, then the enable signal (ENA or ENB) has a priority over the reset signal (RSTA or RSTB) for the memory latch, and the CE signal (REGCEA or REGCEB) has a priority over the reset signal for the output registers. When Reset Priority is chosen as SR, then the reset signal has a priority over the enable signal and the CE signal, in the case of the latch and output registers respectively.

In Virtex-6 devices, reset priority can be set only for the output registers and not for the memory latch. When CE is the selected priority, then CE (REGCEA or REGCEB) has a priority over reset (RSTA or RSTB). When SR is the selected reset priority, then reset has a priority over CE.

Figure 22 illustrates the reset behavior when the Reset Priority option is set to CE. Here, the first reset operation occurs successfully because EN is high, but the second reset operation does not cause any change in output because EN is low.

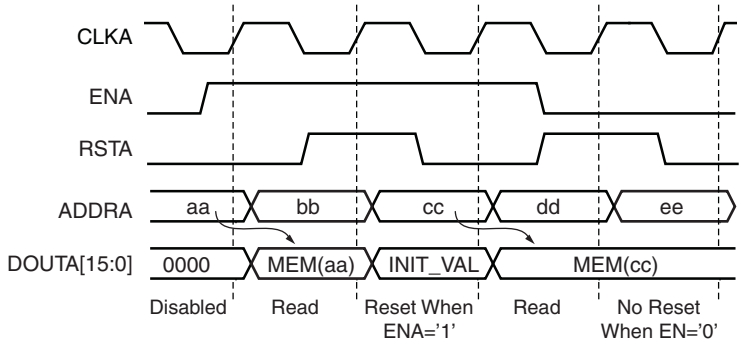

Figure 22: **Reset Behavior When Reset Priority is Set to CE**

Figure 23 illustrates the reset behavior when the Reset Priority option is set to SR. Here, reset is not dependent on enable and both reset operations occur successfully.

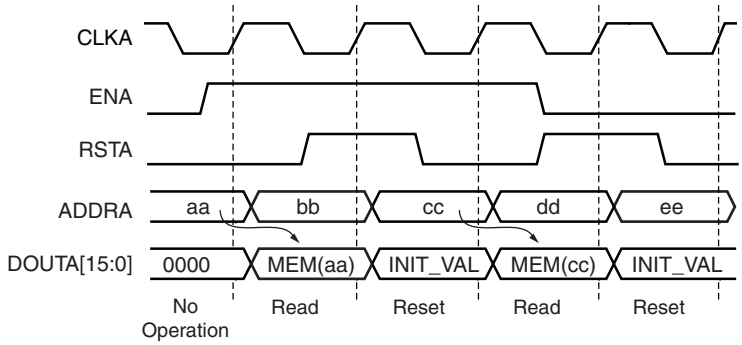

Figure 23: **Reset Behavior When Reset Priority is Set to SR**

Special Reset Behavior

For Spartan-6, Spartan-3A DSP and Virtex-6 devices, the Block Memory Generator provides the option to reset both the memory latch and the embedded primitive output register. This Reset Behavior option is available to users when they choose to have a primitive output register, but no core output register. When a user chooses the option to Reset the Memory Latch besides the primitive output register, then the reset value is asserted at the output for two clock cycles. However, when the user does not choose the option to Reset the Memory Latch in the presence of a primitive output register, the reset value is asserted at the output for only one clock cycle, since only the primitive output register is reset.

Note that the duration of reset assertion specified here is the minimum duration when the latch and register are always enabled, and the RST input is held high for only one clock cycle. If the enable signals are de-asserted or the RST input is held high for more than one clock cycle, the reset value may be asserted at the output for a longer duration.

In Virtex-6 devices, the latch and the embedded output register can be reset independently using two separate inputs (`RSTREG` and `RSTRAM`) that are connected to the primitive. So, if the user does not choose to reset the memory latch, only the embedded output register is reset.

In Spartan-6 and Spartan-3A DSP, the same reset signal (`RST`) is connected to both the latch and the embedded output register. So, if the user does not choose to reset the memory latch, the primitive output register needs to be constructed out of fabric to get the desired behavior. Thus in Spartan-6 and Spartan-3A DSP devices, by choosing the option to reset the memory latch, the reset behavior is modified slightly but resources are saved since the embedded register is used.

Figure 24 and Figure 25 illustrate the difference between the standard reset behavior similar to previous architectures obtained when the memory latch is not reset, and the special reset behavior in the new architectures, obtained when the memory latch is reset. Note that there is an extra clock cycle of latency in the data output because of the presence of the primitive output register.

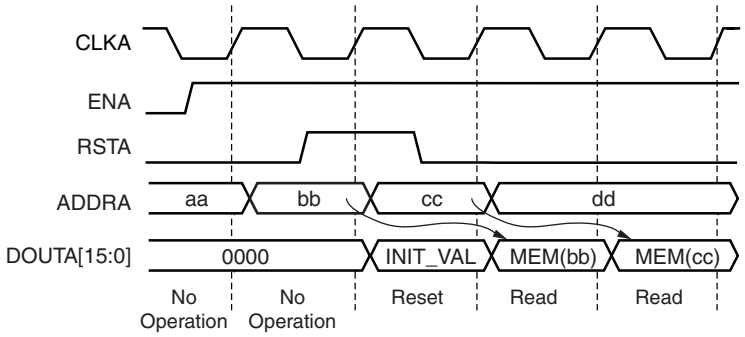

Figure 24: **Standard Reset Behavior Similar to Previous Architectures**

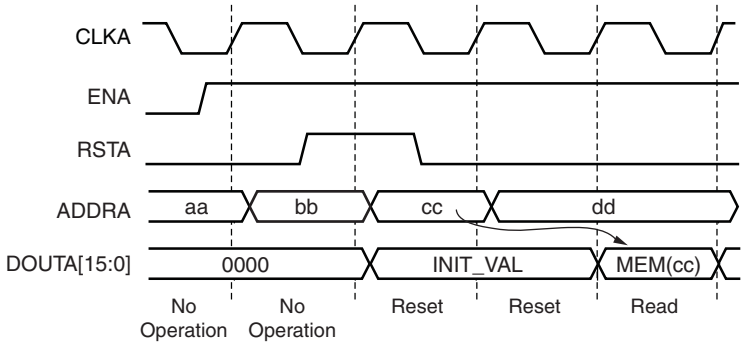

Figure 25: **Special Reset Behavior using the Reset Memory Latch Option**

The special reset behavior of Spartan-3A DSP devices also differs from standard reset behavior in that the reset of the latch and embedded output register is gated by the EN input to the core, independent of the state of REGCE. As shown in Figure 26, the enable input is low during the first reset, and therefore the reset value is not asserted at the output. However during the second reset, the ENA input is high, and the reset value is asserted at the output for two clock cycles."

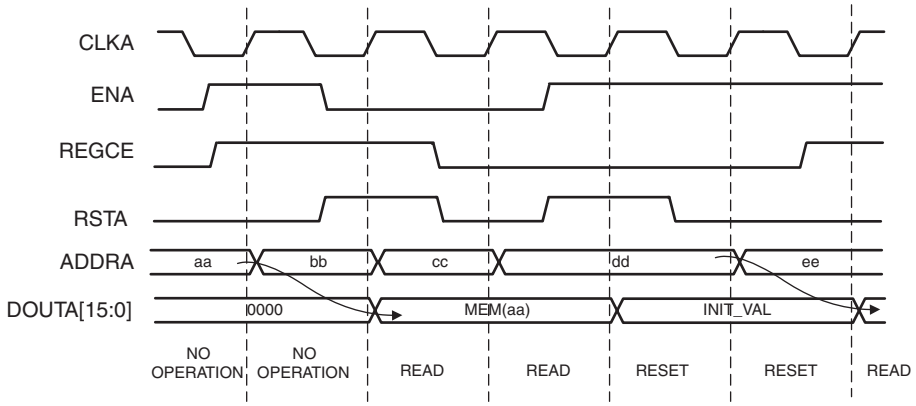

Figure 26: **Reset Gated by EN in Spartan-3A DSP Devices with the Reset Memory Latch Option**

In Spartan-6 and Virtex-6 devices, the reset of the memory latch is gated by EN, and the reset of the embedded register is gated by CE, similar to other architectures. As shown in Figure 27, both ENA and REGCEA are high at the time of the first reset, and the reset value is asserted at the output for two clock cycles. At the time of the second reset, ENA is high, but REGCEA is low; so the reset value does not

appear at the output. At the time of the third reset, only REGCEA is high; so the reset value is asserted at the output for only one clock cycle.

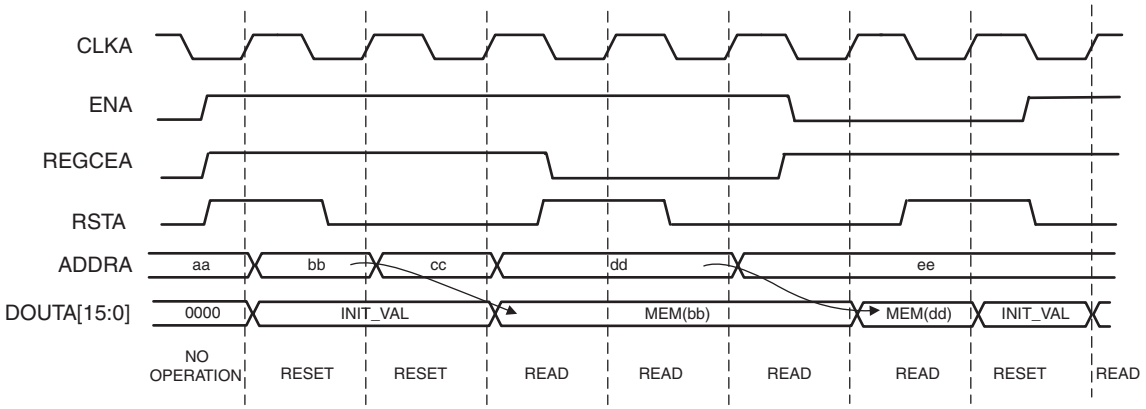

Figure 27: **Reset Gated by CE in Virtex-6 and Spartan-6 Devices with Reset Memory Latch Option**

Asynchronous Reset

The Spartan-6 device architecture provides the ability to asynchronously reset the memory latch and embedded output register. The Block Memory Generator core extends this capability to the core output registers and the primitive output registers constructed out of fabric. The GUI provides the option to choose between the two reset types: synchronous and asynchronous. On using an asynchronous reset, the reset value is asserted immediately when the reset input goes high, but is deasserted only at the following clock edge when reset is low and enable is high. Figure 28 illustrates an asynchronous reset operation in Spartan-6 devices.

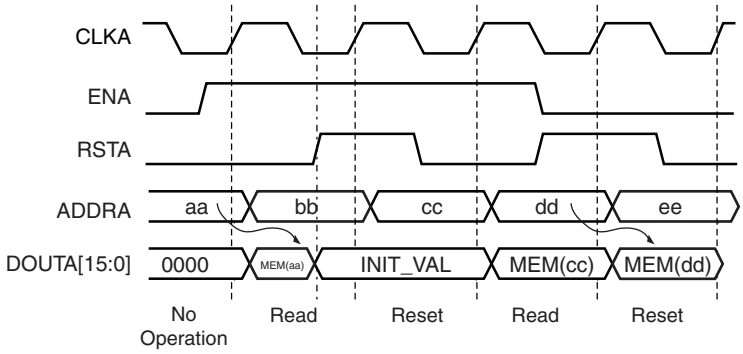

Figure 28: **Asynchronous Reset**

Controlling Reset Operations in Spartan-6 and Virtex-6 FPGAs

The reset operation in Spartan-6 and Virtex-6 devices is dependent on the following generics:

- **Use RST[A | B] Pin**: These options determine the presence or absence of RST pins at the output of the core.
- **Reset Memory Latch (for Port A and Port B)**: This option determines if the memory latch will be reset in addition to the embedded primitive output register for the respective port.
- **Reset Priority (for Port A and Port B)**: This option determines the priority of clock enable over reset or reset over clock enable for the respective port.

- **Reset Type**: This option determines if the reset is synchronous or asynchronous in Spartan-6 devices.

In addition to the above options, the options of output registers also affects reset functionality, since the option to reset the memory latch depends on these options. Table 4 lists the dependency of the reset behavior for Spartan-6 and Virtex-6 devices on these parameters. In these configurations, the core output register does not exist. The reset behavior detailed in Table 4 uses Port A as an example. The reset behavior for Port B is identical.

Table 4: **Control of Reset Behavior in Spartan-6 and Virtex-6 for Single Port**

Use RSTA Pin	Register Port A Output of Memory Primitives	Reset Memory Latch	Reset Priority for Port A	Reset Type (Spartan-6 Only)	S6 RESET BEHAVIOR	V6 RESET BEHAVIOR
0	X	X	X	X	No control over reset	No control over reset
1	0	(cannot be selected)	“CE”, “SR”	“SYNC”, “ASYNC”	Since no primitive output register exists, the reset applies only to the latch. Reset occurs synchronously or asynchronously depending on the Reset Type option, and is dependent or independent of the input enable signal based upon the Reset Priority. The reset value is asserted for only one clock cycle (only if input RST signal is high for only one clock and EN is high continuously).	Since no primitive output register exists, the reset applies only to the latch. For Virtex-6, the priority of SR cannot be set for the latch. Therefore, reset occurs synchronously when the enable input is ‘1’. The reset value is asserted for only one clock cycle (only if input RST signal is high for only one clock and EN is high continuously).
1	1	0	“CE”	“SYNC”, “ASYNC”	Fabric register used. Reset occurs synchronously or asynchronously depending on the Reset Type option, and is dependent or independent of the input enable signal based on the Reset Priority. Reset value asserted for one clock cycle (only if input RST signal is high for only one clock and REGCE is high continuously).	For Virtex-6 devices, the priority cannot be set for the latch. Therefore reset priority of "SR" is not supported.Reset occurs synchronously, and is dependent or independent of the input enable signal. Reset value asserted for one clock cycle (only if input RST signal is high for only one clock and REGCE is high continuously).

Table 4: **Control of Reset Behavior in Spartan-6 and Virtex-6 for Single Port *(Cont'd)***

Use RSTA Pin	Register Port A Output of Memory Primitives	Reset Memory Latch	Reset Priority for Port A	Reset Type (Spartan-6 Only)	S6 RESET BEHAVIOR	V6 RESET BEHAVIOR
1	1	1	"CE", "SR"	"SYNC", "ASYNC"	Both memory latch and embedded output register of primitive are reset. Reset occurs synchronously or asynchronously depending on the Reset Type option, and is dependent or independent of the input enable signal based upon the Reset Priority. Reset value is asserted for at least two clock cycles when enable inputs of both stages are '1', and may be more depending on the input RST and enable signals. If RST is asserted when the latch EN input is '1' and the register enable input is '0', the memory latch alone gets reset and this reset value gets output only when the register enable goes high.	For Virtex-6 devices, the priority cannot be set for the latch. Therefore reset priority of "SR" is not supported. Both memory latch and embedded output register of primitive are reset. Reset occurs synchronously at both these stages, and is dependent or independent of the input enable signal. Reset value is asserted for at least two clock cycles when enable inputs of both stages are '1', and may be more depending on the input RST and enable signals. If RST is asserted when the latch EN input is '1' and the register enable input is '0', the memory latch alone gets reset and this reset value gets output only when the register enable goes high.
1	1	0	"SR"	"SYNC", "ASYNC"	Fabric register used. Reset occurs synchronously or asynchronously when the RST input is '1', irrespective of the state of the enable input. However, the reset value will get deasserted synchronously only once the enable input is '1'.	Not applicable.
1	1	1	"SR"	"SYNC", "ASYNC"	Reset occurs synchronously or asynchronously when the RST input is '1', irrespective of the state of the enable input. However, the reset value will get deasserted synchronously when the latch and embedded register are enabled sequentially for at least one clock cycle each after the reset input is deasserted.	Not applicable.
1	X	X	"SR", "CE"	ASYNC	Reset occurs asynchronously when the RST input is '1'. Dependencies on the remaining options are as explained above.	Not applicable.

Built-in Error Correction Capability and Error Injection

For Virtex-6 and 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-6 and Virtex-5 FPGAs
- Simple Dual-port RAM memory type

When using ECC, the Block Memory Generator constructs the memory from special primitives available in Virtex-6 and 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.

The Virtex-6 and Virtex-5 ECC primitives calculate ECC for a 64-bit wide data input. If the data width chosen by a user is not an integral multiple of 64 (for example, there are spare bits in any ECC primitive), then a double bit error (DBITERR) may indicate that one or more errors have occurred in the spare bits. So, the accuracy of the DBITERR signal cannot be guaranteed in this case. For example, if the user's data width is 32, then 32 bits of the primitive are left spare. If two of the spare bits are corrupted, the DBITERR signal would be asserted even though the actual user data is not corrupt.

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

- Byte-write enable is not available
- All port widths must be identical
- For Virtex-5 devices, No Change Operating mode is supported, and for Virtex-6 devices, Read First Operating Mode is supported
- Use RST[A | B] Pin and the Output Reset Value options are not available
- Memory Initialization is not supported
- No Algorithm selection is available

Figure 29 illustrates a typical write and read operation for a Virtex-6 or Virtex-5 FPGA Block Memory Generator core in Simple Dual-port RAM with ECC enabled, and no additional output registers.

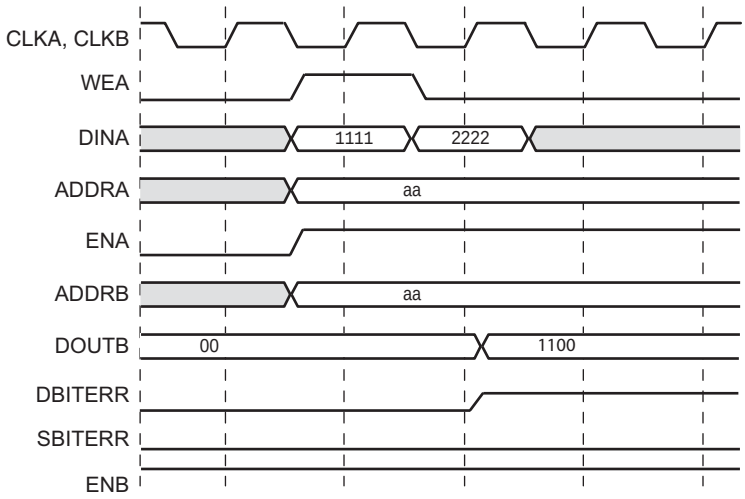

Figure 29: **Read and Write Operation with ECC in Virtex-6 or Virtex-5 FPGAs**

Error Injection

For 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.

Virtex-6 devices, however, support error injection through two new optional pins: INJECTSBITERR and INJECTDBITERR. Users can use these optional error injection pins as debug pins to inject single or double bit errors on specific locations during write operations. Then, the user can check the assertion of the SBITERR and DBITERR signals at the output of those addresses. The user has the option to have no error injection pins, or to have only one or both of the error injection pins.

The RDADDRECC output port indicates the address at which a SBITERR or DBITERR has occurred. The RDADDRECC port, the two error injection ports, and the two error output ports are optional and come into existence only when the ECC option is chosen. If the ECC feature is not selected by the user, the primitive's INJECTSBITERR and INJECTDBITERR ports are internally driven to '0', and the primitive's outputs SBITERR, DBITERR, and RDADDRECC are not connected externally.

Figure 30 shows the assertion of the SBITERR and DBITERR output signals when errors are injected through the error injection pins during a write operation.

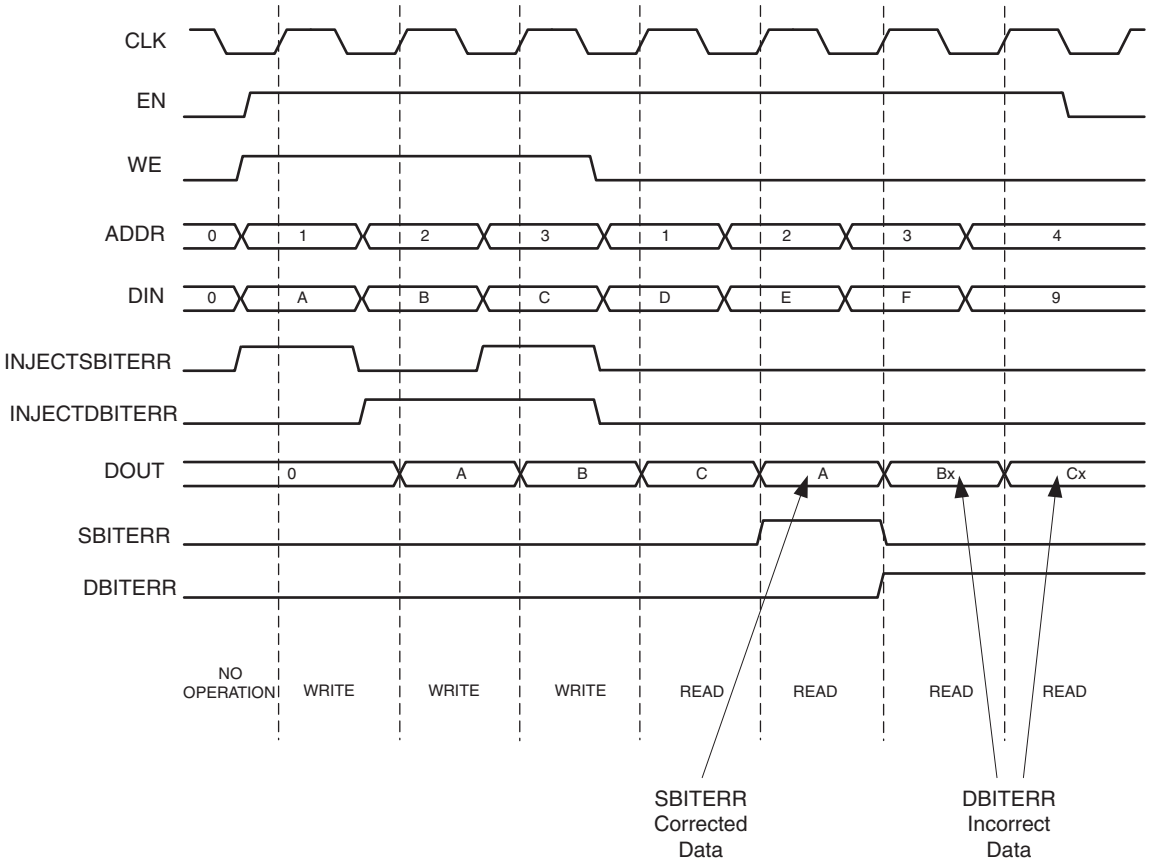

Figure 30: **Assertion of SBITERR and DBITERR Signals by Using Error Injection Pins**

When the INJECTSBITERR and INJECTDBITERR inputs are asserted together at the same time for the same address during a write operation (as in the case of address 3 in Figure 30), then the INJECTDBITERR input takes precedence, and only the DBITERR output is asserted for that address during a read operation. The data output for this address is not corrected.

Smaller Primitive Configurations in Spartan-6

The introduction of the new 9K primitive in Spartan-6 results in smaller memory configurations: 8kx1, 4kx2, 2kx4, 1kx9, 512x18 and 256x36 (this primitive is only supported in SP and SDP configurations). In previous Spartan families, extra-wide configurations were only supported in Single Port memory configurations. The 9K primitive, RAMB8BWER, allows the primitive to be configured in either the TDP or the SDP mode. The presence of the SDP mode of operation allows the extra-wide 256x36 configuration, even for Simple Dual Port memory configurations.

Lower Data Widths in Virtex-6 SDP Configurations

The Virtex-6 FPGA architecture with new SDP primitives supports lower data widths as compared to Virtex-5 FPGAs. In Virtex-5 devices, the RAMB18SDP primitive could only support a symmetric configuration with port widths of 36, and the RAMB36SDP primitive could only support a symmetric con-

figuration with port widths of 72. For Virtex-6 devices, new width combinations are possible for Port A and Port B, as shown in Table 5.

Table 5: **Data Widths Supported by Virtex-6 Device SDP Primitives**

Primitive	Read Port Width	Write Port Width	Read Port Width	Write Port Width
RAMB18 SDP Primitive	x1	x32	x32	x1
	x2	x32	x32	x2
	x4	x32	x32	x4
	x9	x36	x36	x9
	x18	x36	x36	x18
	x36	x36	-	-
RAMB36 SDP Primitive	x1	x64	x64	x1
	x2	x64	x64	x2
	x4	x64	x64	x4
	x9	x72	x72	x9
	x18	x72	x72	x18
	x36	x72	x72	x36
	x72	x72	-	-

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 6 defines the differences between the two functional simulation models.

Table 6: **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 7 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 ADDRB) 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 7: **Core Signal Pinout**

Name	Direction	Description
CLKA	Input	**Port A Clock:** Port A operations are synchronous to this clock. For synchronous operation, this must be driven by the same signal as CLKB.
ADDRA	Input	**Port A Address:** Addresses the memory space for port A read and write operations. Available in all configurations.
DINA	Input	**Port A Data Input:** Data input to be written into the memory via port A. Available in all RAM configurations.
DOUTA	Output	**Port A Data Output:** Data output from read operations via port A. Available in all configurations except Simple Dual-port RAM.
ENA	Input	**Port A Clock Enable:** Enables read, write, and reset operations via port A. Optional in all configurations.
WEA	Input	**Port A Write Enable:** Enables write operations via port A. Available in all RAM configurations.
RSTA	Input	**Port A Set/Reset:** Resets the Port A memory output latch or output register. Optional in all configurations.
REGCEA	Input	**Port A Register Enable:** Enables the last output register of port A. Optional in all configurations with port A output registers.
CLKB	Input	**Port B Clock:** 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	**Port B address:** Addresses the memory space for port B read and write operations. Available in dual-port configurations.
DINB	Input	**Port B Data Input:** Data input to be written into the memory via port B. Available in True Dual-port RAM configurations.
DOUTB	Output	**Port B Data Output:** Data output from read operations via Port B. Available in dual-port configurations.
ENB	Input	**Port B Clock Enable:** Enables read, write, and reset operations via Port B. Optional in dual-port configurations.
WEB	Input	**Port B Write Enable:** Enables write operations via Port B. Available in Dual-port RAM configurations.
RSTB	Input	**Port B Set/Reset:** Resets the Port B memory output latch or output register. Optional in all configurations.
REGCEB	Input	**Port B Register Enable:** Enables the last output register of port B. Optional in dual-port configurations with port B output registers.
SBITERR	Output	**SIngle Bit Error**: Flags the presence of a single-bit error in memory which has been auto-corrected on the output bus.
DBITERR	Output	**Double Bit Error**: Flags the presence of a double-bit error in memory. Double-bit errors cannot be auto-corrected by the built-in ECC decode module.

Table 7: **Core Signal Pinout *(Cont'd)***

Name	Direction	Description
INJECTSBITERR	Input	**Inject Single Bit Error:** Available only for Virtex-6 ECC configurations
INJECTDBITERR	Input	**Inject Double Bit Error:** Available only for Virtex-6 ECC configurations
RDADDRECC	Output	**Read Address for ECC Error output:** Available only for Virtex-6 ECC configurations

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 maximum possible customization options in the Block Memory Generator GUI screens. The actual GUI screens with enabled options will depend on the user configuration.

CORE Generator Parameter Screens

The Block Memory Generator GUI includes five main screens:

- "Block Memory Generator Main Screen"
- "Port Options Screen"
- "Output Registers and Memory Initialization Screen"
- "Reset Options 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

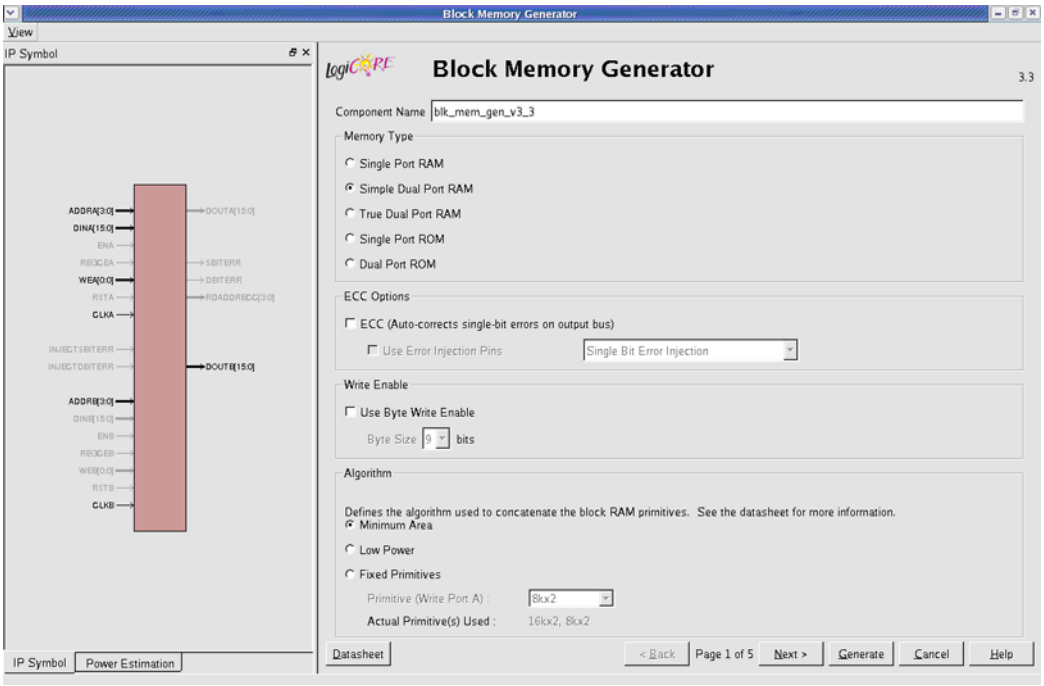

Figure 31: **Block Memory Generator Main Screen**

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-6 or Virtex-5 devices, and when the Simple dual-port RAM memory type is selected, the ECC options become available. Selecting ECC enables built-in Hamming error correction for the Virtex-6 and Virtex-5 FPGA architecture.

For the Virtex-6 FPGA, the Use Error Injection Pins option is available for selection once the ECC option is selected. It provides the user the choice to have error injection pins. On choosing this option, additional options are available to have only Single Bit Error Injection using the `INJECTSBITERR` pin, or only Double Bit Error Injection using the `INJECTDBITERR` pin, or both Single and Double Bit Error

Injection using both pins. See "Built-in Hamming Error Correction Capability," 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
- For Virtex-5, No Change Operating mode is supported, and for Virtex-6, Read First Operating Mode is supported
- Use RST[A|B] Pin and the Output Reset Value options are not available
- Memory Initialization is not supported
- No algorithm selection is available

Write Enable

When targeting Virtex-6, Virtex-5, Virtex-4, Spartan-6, 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 Options Screen

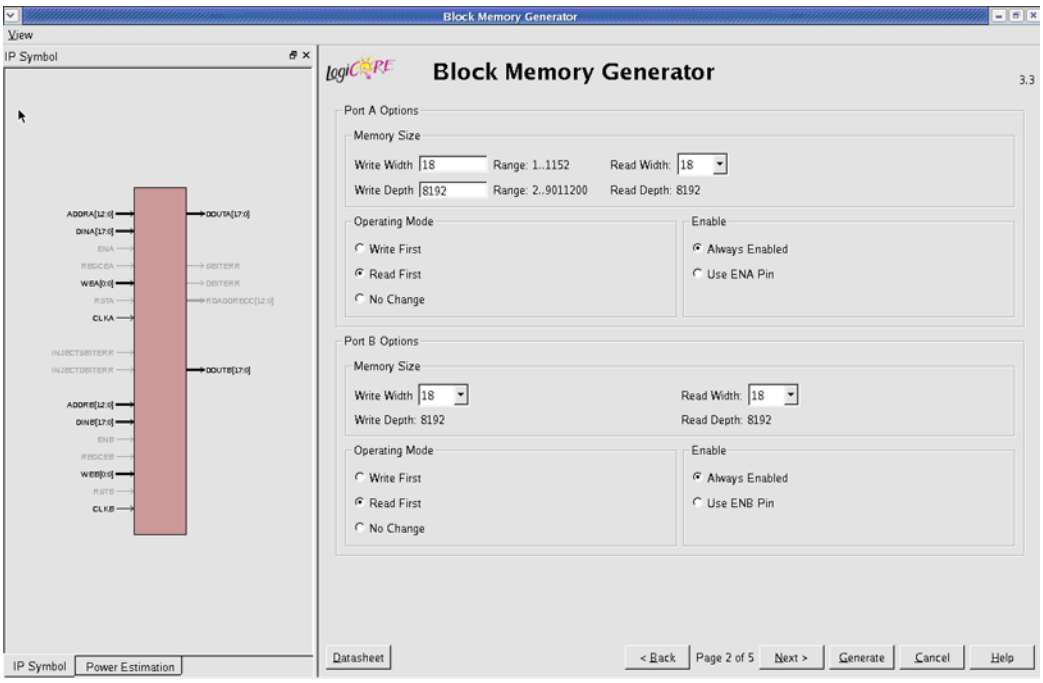

Figure 32: **Port A Options**

Port A Options

Memory Size

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

Specify the port A operating mode.

- READ_FIRST
- WRITE_FIRST
- NO_CHANGE

Enable

Select the enable type:

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

Port B Options Screen

Memory Size

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

Operating Mode

Specify the port B write mode.

- READ_FIRST
- WRITE_FIRST
- NO_CHANGE

Enable

Select the enable type:

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

Output Registers and Memory Initialization Screen

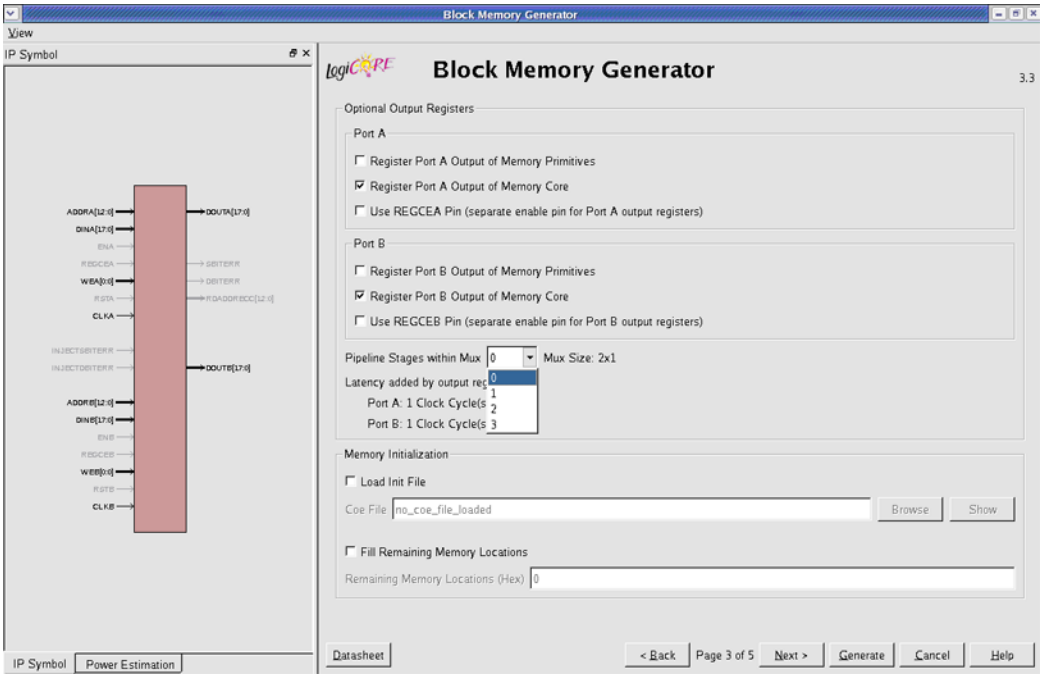

Figure 33: **Output Registers and Memory Initialization Screen**

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-6, Virtex-5 or Virtex-4 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 Spartan-6 and Spartan-3A DSP, either the primitive embedded registers or fabric registers from FPGA slices are used, depending upon the Reset Behavior option chosen by the user. For other architectures, the registers in the FPGA slices are used. Note that in Virtex-4 devices, the use of the RST input prevents the core from using the embedded output registers. See "Output Register Configurations," page 57 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.
- **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.

Reset Options Screen

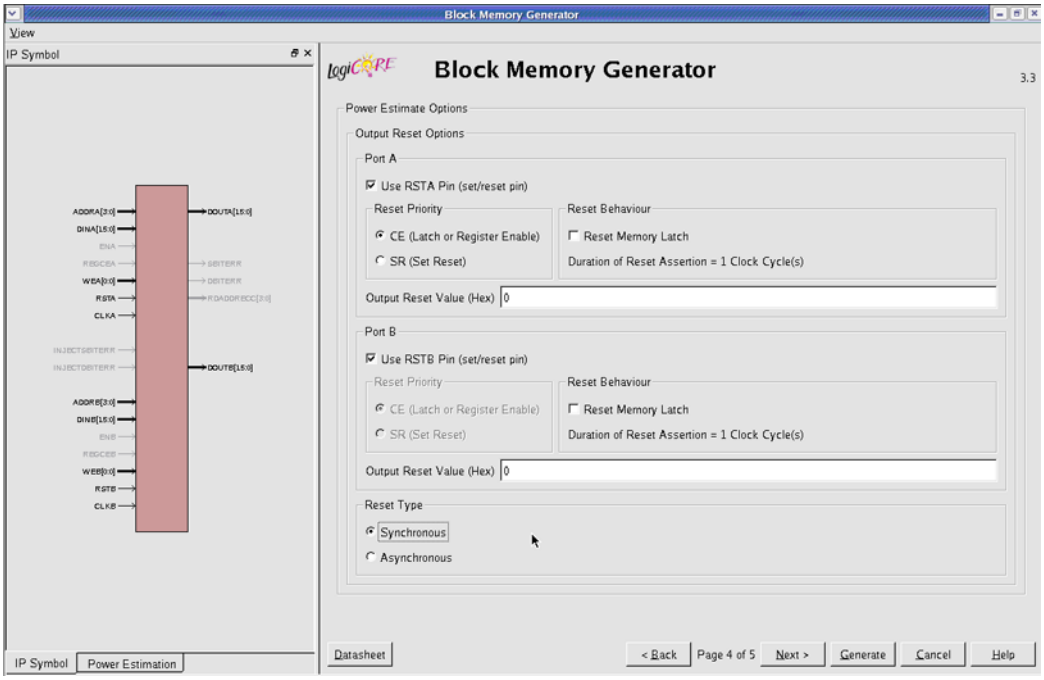

Figure 34: **Reset Options Screen**

Port [A|B] Output Reset Options

- **Use RST[A|B] Pin**: Choose whether a set/reset pin (RST[A|B]) is needed.
- **Reset Priority**: The Reset Priority option for each port is available only when the Use RST Pin option of the corresponding port is chosen. The user can set the reset priority to either CE or SR. For more information on the reset priority feature, see "Reset Priority," page 22.
- **Reset Behavior**: The Reset Behavior (Reset Memory Latch) options for each port are available only when the Use RST Pin option and the Register Output of Memory Primitives option of the corresponding port are chosen, and the Register Memory Core option of the corresponding port is not chosen.

  The Reset Memory Latch option modifies the behavior of the reset and changes the duration for which the reset value is asserted. The minimum duration of reset assertion is displayed as information in the GUI based upon the choice for this option. For more information on the Reset Memory Latch option, see "Special Reset Behavior," page 23.
- **Output Reset Value (Hex)**: Specify the reset value of the memory output latch and output registers. These values are with respect to the read port widths.

Reset Type

The Reset Type option is available only for Spartan-6 devices and when either or both of Use RSTA Pin or Use RSTB Pin option are chosen. The user can set the reset type to either Synchronous or Asynchronous. For more information on this option, see "Asynchronous Reset," page 25.

Simulation Model Options and Information Screen

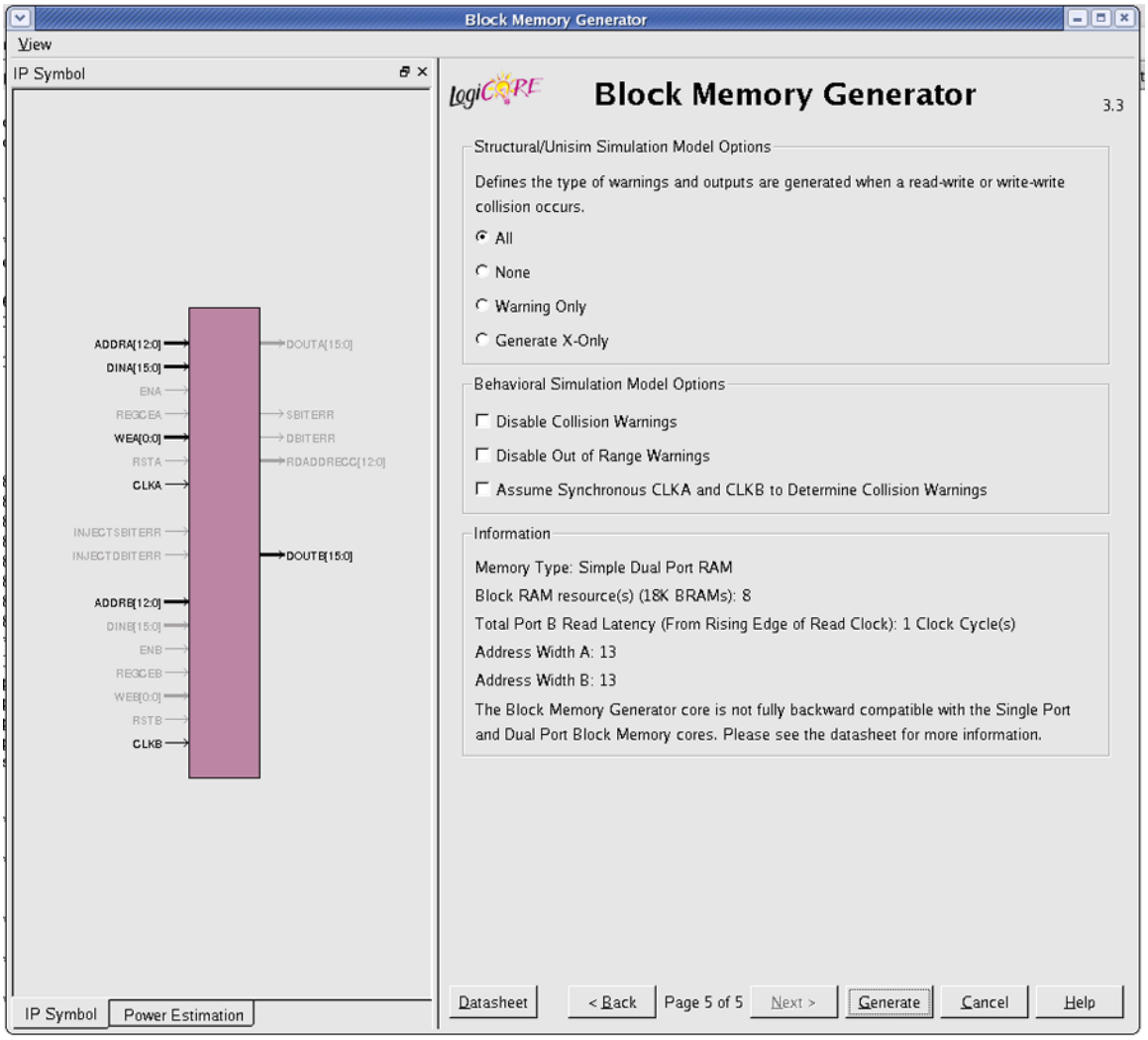

Figure 35: **Simulation Model Options and Information Screen**

Power Estimate Options

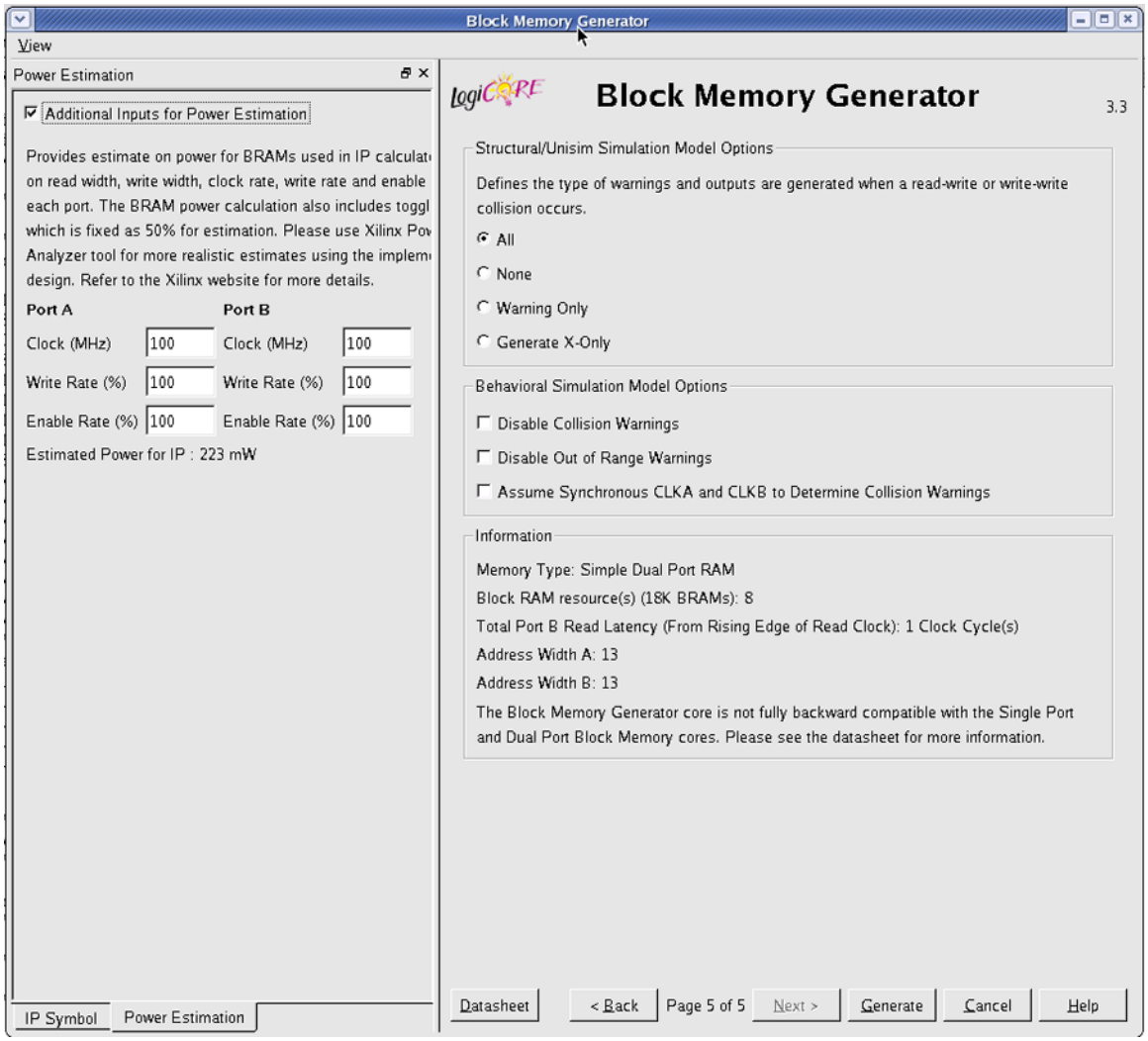

Figure 36: **Power Estimate Options Screen**

The "Power Estimation" tab on the left side of the GUI screen shown in Figure 36 provides a rough estimate of power consumption for the core based on the configured read width, write width, clock rate, write rate and enable rate of each port. The power consumption calculation assumes a toggle rate 50%. More accurate estimates can be obtained on the routed design using the XPower Analyzer tool. See www.xilinx.com/power for more information on the XPower Analyzer.

The screen has an option to provide "Additional Inputs for Power Estimation" apart from configuration parameters. The following parameters can be entered by the user for power calculation:

- **Clock Frequency [A | B]**: The operating clock frequency of the two ports A and B respectively.
- **Write Rate [A | B]**: Write rate of ports A and B respectively.
- **Enable Rate [A | B]**: Average access rate of port A and B respectively.

Structural/UNISIM Simulation Model Options

Select the type of warning messages and outputs generated by the structural simulation model in the event of collisions. For the options of ALL, WARNING_ONLY and GENERATE_X_ONLY, the collision detection feature will be enabled in the UniSim models to handle collision under any condition.

The NONE selection is intended for designs that have no collisions and clocks (Port A and Port B) that are never in phase or within 3000 ps in skew. If NONE is selected, the collision detection feature will be disabled in the models, and the behavior during collisions is left for the simulator to handle. So, the output will be unpredictable if the clocks are in phase or from the same clock source or within 3000 ps in skew, and the addresses are the same for both ports. The option NONE is intended for design with clocks never in phase.

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**: Reports the exact number of 9K, 18K and 36K block RAM primitives which will be used to construct the core.
- **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 9K, 18K and 36K block RAM blocks to be 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 of the number of 18K block RAMs 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-4 FPGA in the -10 speed grade (4VLX60-FF1148-10), Virtex-5 FPGA in the -1 speed grade (5VLX30-FF324-1), Virtex-6 FPGA in the -1 speed grade (XC6VLX365T-FF1759-1) and a Spartan-6 FPGA in the -2 speed grade (XC6SLX150T-FGG484-2). 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. It is recommended that users register their inputs to the core for better performance. The following examples highlight the use of embedded registers in Virtex-4, Virtex-5, Virtex-6 and Spartan-6 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 8 through Table 11 define performance data for single-primitive memories.

Table 8: **Single Primitive Examples - Virtex-6 FPGAs**

Memory Type	Options	Width x Depth	Resource Utilization						Performance (MHz)
			Block RAMs			Shift Regs	FFs	LUTs(1)	
			36K	16K	8K				
True Dual-port RAM	No Output Registers	36x512	1	0	0	0	0	0	325
		9x2k	0	1	0	0	0	0	325
	Embedded Output Registers	36x512	1	0	0	0	0	0	450
		9x2k	0	1	0	0	0	0	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.

Table 9: **Single Primitive Examples - Virtex-5 FPGAs**

Memory Type	Options	Width x Depth	Resource Utilization						Performance (MHz)
			Block RAMs			Shift Regs	FFs	LUTs(1)	
			36K	16K	8K				
True Dual-port RAM	No Output Registers	36x512	1	0	0	0	0	0	300
		9x2k	0	1	0	0	0	0	325
	Embedded Output Registers	36x512	1	0	0	0	0	0	450
		9x2k	0	1	0	0	0	0	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.

Table 10: **Single Primitive Examples - Virtex-4 FPGAs**

Memory Type	Options	Width x Depth	Resource Utilization				Performance (MHz)
			Block RAMs 16K	Shift Regs	FFs	LUTs(1)	Virtex-4
True Dual-port RAM	No Output Registers	36x512	1	0	0	0	300
		9x2k	1	0	0	0	325
	Embedded Output Registers	36x512	1	0	0	0	400
		9x2k	1	0	0	0	400

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 11: **Single Primitive Examples - Spartan-6 FPGAs**

Memory Type	Options	Width x Depth	Resource Utilization						Performance (MHz)
			Block RAMs			Shift Regs	FFs	LUTs(1)	
			36K	16K	8K				
True Dual-port RAM	No Output Registers	36x512	0	1	0	0	0	0	200
		9x2k	0	1	0	0	0	0	225
	Embedded Output Registers	36x512	0	1	0	0	0	0	275
		9x2k	0	1	0	0	0	0	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.

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.

In Virtex-6, Virtex-5, Virtex-4, and Spartan-6 architectures, the embedded block RAM may be utilized, reducing the FPGA fabric resources required to create the registers.

Table 12: **Virtex-6 Device Output Register Examples**

Memory Type	Width x Depth	Output Register Options	Block RAM			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k		1	3	0	0	3	18	325
		Primitive	1	3	0	3	3	18	450
		Core	1	3	0	0	20	18	325
		Primitive, Core	1	3	0	3	20	18	450

Table 12: **Virtex-6 Device Output Register Examples *(Cont'd)***

Memory Type	Width x Depth	Output Register Options	Block RAM			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
True Dual-port RAM	17x5k		1	3	0	0	6	36	300
		Primitive	1	3	0	6	6	36	450
		Core	1	3	0	0	40	36	300
		Primitive, Core	1	3	0	6	40	36	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.

Table 13: **Virtex-5 Device Output Register Examples**

Memory Type	Width x Depth	Output Register Options	Block RAM			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k		1	3	0	0	3	18	300
		Primitive	1	3	0	3	3	18	450
		Core	1	3	0	0	20	18	300
		Primitive, Core	1	3	0	3	20	18	450
True Dual-port RAM	17x5k		1	3	0	0	6	36	300
		Primitive	1	3	0	6	6	36	450
		Core	1	3	0	0	40	36	300
		Primitive, Core	1	3	0	6	40	36	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.

Table 14: **Virtex-4 Device Output Register Examples**

Memory Type	Width x Depth	Output Register Option	Block RAMs 16K	Shift Regs	FFs	LUTs(1)	Performance (MHz)
Single-port RAM	17x5k	-	5	0	3	30	275
		Primitive	5	3	3	30	400
		Core	5	0	20	30	275
		Primitive, Core	5	2	22	32	400
True Dual-port RAM	17x5k	-	5	0	6	60	275
		Primitive	5	6	6	148	375
		Core	5	0	40	142	250
		Primitive, Core	5	6	40	148	375

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 15: **Spartan-6 Device Output Register Examples**

Memory Type	Width x Depth	Output Register Options	Block RAM			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k		0	5	0	0	3	19	175
		Primitive	0	5	0	3	3	19	250
		Core	0	5	0	0	20	19	175
		Primitive, Core	0	5	0	3	20	19	225
True Dual-port RAM	17x5k		0	5	0	0	6	38	175
		Primitive	0	5	0	4	6	38	250
		Core	0	5	0	0	40	38	175
		Primitive, Core	0	5	0	4	40	38	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.

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 16: **Virtex-6 Device Aspect Ratio**

Memory Type	Width x Depth	Data Width Aspect Ratio	Block RAMs			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k	1:1	2	3	0	0	6	36	300
		1:8(2)	8	1	0	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. Read port is 136x640; write port is 17x5k.

Table 17: **Virtex-5 Device Aspect Ratio**

Memory Type	Width x Depth	Data Width Aspect Ratio	Block RAMs			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k	1:1	2	3	0	0	6	36	300
		1:8(2)	8	1	0	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. Read port is 136x640; write port is 17x5k.

Table 18: **Virtex-4 Device Aspect Ratio**

Memory Type	Width x Depth	Data Width Aspect Ratio	Block RAM 16K	Shift Regs	FFs	LUTs(1)	Performance (MHz)
Single Port	17x5k	1:1	5	0	6	60	275
		1:8(2)	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. Read port is 136x640; write port is 17x5k.

Algorithm

The differences between the minimum area, low power and fixed primitive algorithms are discussed in detail in "Selectable Memory Algorithm," page 3. Table 19 shows examples of the resource utilization and the performance difference between them for two selected configurations for Virtex-6 FPGA architectures.

Table 19: **Memory Algorithm Examples Virtex-6 Devices**

Memory Type	Width x Depth	Algorithm Type	Block RAM			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k	Minimum area	1	3	0	0	3	18	325
		Fixed Primitive using 18x1k block RAM	2	1	0	0	3	19	300
		Low power	0	5	0	0	3	37	275
	36x4k	Minimum area	4	0	0	0	0	0	325
		Fixed Primitive using 36x512 block RAM	4	0	0	0	2	38	275
		Low power	4	0	0	0	3	76	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.

Table 20 shows examples of the resource utilization and the performance difference between them for two selected configurations for Virtex-5 FPGA architecture.

Table 20: **Memory Algorithm Examples Virtex-5 Devices**

Memory Type	Width x Depth	Algorithm Type	Block RAM			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k	Minimum area	1	3	0	0	3	18	300
		Fixed Primitive using 18x1k block RAM	2	1	0	0	3	20	300
		Low power	0	5	0	0	3	39	275
	36x4k	Minimum area	4	0	0	0	0	0	300
		Fixed Primitive using 36x512 block RAM	4	0	0	0	2	40	275
		Low power	0	8	0	0	3	80	250

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 21 shows examples of the resource utilization and the performance difference between them for two selected configurations for Virtex-4 FPGA architecture.

Table 21: **Memory Algorithm Examples Virtex-4 Devices**

Memory Type	Width x Depth	Algorithm Type	Resource Utilization				Performance (MHz)
			Block RAM	Shift Regs	FFs	LUTs(1)	
Single-port RAM	17x5k	Minimum area	5	0	3	30	275
		Fixed Primitive using 18x1k block RAM	5	0	3	57	225
		Low power	5	0	3	57	225
	36x4k	Minimum area	8	0	1	36	275
		Fixed Primitive using 36x512 block RAM	8	0	3	152	225
		Low power	8	0	3	152	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 22 shows examples of the resource utilization and the performance difference between them for two selected configurations for Spartan-6 FPGA architecture.

Table 22: **Memory Algorithm Examples Spartan-6 Devices**

Memory Type	Width x Depth	Algorithm Type	Block RAM			Shift Regs	FFs	LUTs(1)	Performance (MHz)
			36K	16K	8K				
Single-port RAM	17x5k	Minimum area	0	5	0	0	3	19	175
		Fixed Primitive using 18x1k block RAM	0	5	0	0	3	37	175
		Low power	0	0	10	0	4	57	150
	36x4k	Minimum area	0	8	0	0	1	18	175
		Fixed Primitive using 36x512 block RAM	0	8	0	0	3	76	150
		Low power	0	0	16	0	4	170	125

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.

- "Low Power Designs": Provides information on the Low Power algorithm and methods that can be followed by the user to optimize power consumption in block RAM designs.
- "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.

Low Power Designs

The Block Memory Generator core also supports a Low Power implementation algorithm. When this option is selected, the configuration of the core is optimized to minimize dynamic power consumption. This contrasts with the Minimum Area algorithm, which optimizes the core implementation with the sole purpose of minimizing resource utilization.

The Low Power algorithm reduces power through the following mechanisms:

- Minimizing the number of block RAMs enabled for a write or read operation for a given memory size.
- Unlike the Minimum Area algorithm, smaller block RAM blocks are not grouped to form larger blocks in the Low Power algorithm. For example, two 9K block RAMs are not combined to form an 18K block RAM in Spartan-6 devices, and two 18K block RAMs are not combined to form a 36K block RAM in Virtex-5 devices.
- The "Always Enabled" option is not available to the user for the Port A and Port B enable pins. This prevents the block RAMs from being enabled at all times.
- The NO_CHANGE mode is set as the default operating mode.

Table 23 and Table 24 compare power consumption and resource utilization for Low Power and Minimum Area block memory implementations targeted to Virtex-5 devices and the Spartan-3 family of devices. Estimated power consumption values were obtained using the XPE Spreadsheets from the Power Solutions web page on Xilinx.com:

www.xilinx.com/products/design_resources/power_central/index.htm

XPE is a pre-implementation power estimation tool appropriate for estimating power requirements in the early stages of a design. For more accurate power consumption estimates and power analysis, the Xilinx Power Analyzer tool (XPA) available in ISE can be run on designs after place and route.

Power data shown in the following tables was collected assuming a 50% toggle rate and 50% write rate. A frequency of 300 MHz was specified for Virtex-5 devices, and a frequency of 150 MHz was specified for the Spartan-3 family of devices.

Note: Data shown in Table 23 is preliminary and subject to change. Data shown is based on Virtex-5 FPGA XPE spreadsheet v11.1. Always use the latest version of each target architecture XPE spreadsheet to get the most accurate estimate.

Table 23: **Power-Resource benchmarking for Virtex-5(1)**

Memory Configuration	Memory Type	Operating Mode	Dynamic Power Consumption for Single Read/Write (mW)		Block RAM Resource Utilization (Number of 18K Block RAMs)	
			Minimum Area	Low Power	Minimum Area	Low Power
2kx36	TDP	Write First/ Read First	39	21	4	4
		No Change	34	19	4	4
8kx12	TDP	Write First/ Read First	51	11	6	8
		No Change	45	10	6	8

Table 23: **Power-Resource benchmarking for Virtex-5(1) *(Cont'd)***

Memory Configuration	Memory Type	Operating Mode	Dynamic Power Consumption for Single Read/Write (mW)		Block RAM Resource Utilization (Number of 18K Block RAMs)	
			Minimum Area	Low Power	Minimum Area	Low Power
8kx72	TDP	Write First/ Read First	148	43	32	32
		No Change	129	38	32	32
8kx72	SP	Write First/ Read First	74	21	32	32(2)
		No Change	65	19	32	32(2)

1. Assumptions: 50% toggle rate and 50% write rate; 300MHz frequency.
2. Use of 512x72 extra-wide primitive in a Single Port configuration.

Note: Data shown in Table 24 is preliminary and subject to change. Data shown is based on Spartan-3 FPGA XPE spreadsheet v11.1. Always use the latest version of each target architecture XPE spreadsheet to get the most accurate estimate

Table 24: **Power-Resource benchmarking for Spartan-3 Family(1) (2)**

Memory Configuration	Memory Type	Dynamic Power Consumption for Single Read/Write (mW)		Block RAM Resource Utilization (Number of 18K Block RAMs)	
		Minimum Area	Low Power	Minimum Area	Low Power
2kx36	TDP	32	13	4	4
8kx12	TDP	44	10	6	8
8kx72	TDP	69	30	32	32
8kx72	SP	38	15	32	32(3)

1. All Spartan-3 and derivative families have similar power estimates.
2. Assumptions: 50% toggle rate and 50% write rate; 150 MHz frequency.
3. Use of 256x72 extra-wide primitive in a Single Port configuration.

Besides using the Low Power algorithm, the following design considerations are recommended for power optimizations:

- The Low Power algorithm disables the "Always Enabled" option for the Port A and Port B enable pins, and the user is forced to have these pins at the output (the "Use EN[A | B] Pin" option). These pins must not be permanently tied to '1' if it is desired that power be conserved. Each port's enable pin must be asserted high only when that port of the block RAM needs to be accessed.
- Use of output registers improves performance, but also increases power consumption. Even if used in the design, the output registers should be disabled when the block RAMs are not being accessed.
- The user can choose the operating mode even in the Low Power algorithm based upon design requirements; however it is recommended that the default operating mode of the Low Power algorithm (NO_CHANGE mode) be used. This mode results in lower power consumption as compared to the WRITE_FIRST and READ_FIRST modes.

Compatibility with Older Memory Cores

The Block Memory Generator Migration Kit can be used to migrate from legacy memory cores (Dual Port Block Memory and Single Port Block Memory cores) and older versions of the Block Memory Generator core to the latest version of the Block Memory Generator core.

For information about using the Migration Kit, see the Block Memory Generator Migration Kit Product Page.

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 ADDRB 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 37.

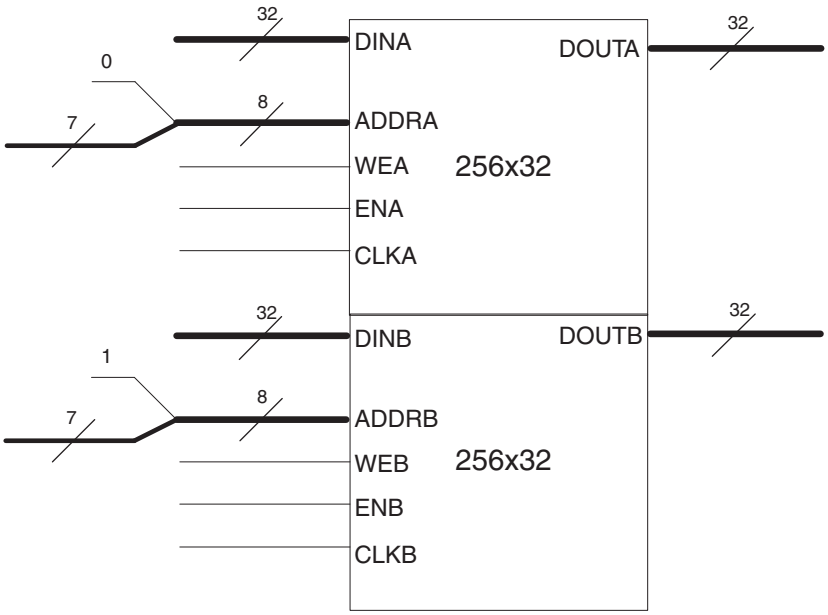

Figure 37: **Construction of Two Independent 128x32 Memories using a Single 256x32 Memory**

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 25 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 25: **SIM Parameters**

	SIM Parameter	Type	Values	Description
1	C_FAMILY	String	"Virtex4" "Virtex5" "Spartan-3"	Target device family
2	C_XDEVICEFAMILY	String	"Virtex4" "Virtex5" "Spartan-3" "Spartan-3A" "Spartan-3A DSP"	Finest resolution target family derived from the parent C_FAMILY
3	C_ELABORATION_DIR	String		Elaboration Directory
4	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 ROM	Type of memory
5	C_ALGORITHM	Integer	0 (selectable primitive), 1 (minimum area), 2 (low power)	Type of algorithm
6	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
7	C_BYTE_SIZE	Integer	9, 8	Defines size of a byte: 9 bits or 8 bits
8	C_SIM_COLLISION_CHECK	String	None, Generate_X, All, Warnings_only	Defines warning collision checks in structural/unisim simulation model
9	C_COMMON_CLOCK	Integer	0, 1	Determines whether to optimize behavioral models for read/write accesses and collision checks by assuming clocks are synchronous. It is recommended to set this option to "0" when both the clocks are not synchronous, in order to have the models function properly.

Table 25: **SIM Parameters *(Cont'd)***

	SIM Parameter	Type	Values	Description
10	C_DISABLE_WARN_BHV_COLL	Integer	0, 1	Disables the behavioral model from generating warnings due to read-write collisions
11	C_DISABLE_WARN_BHV_RANGE	Integer	0, 1	Disables the behavioral model from generating warnings due to address out of range
12	C_LOAD_INIT_FILE	Integer	0, 1	Defined whether to load initialization file
13	C_INIT_FILE_NAME	String	"…"	Name of initialization file (MIF format)
14	C_USE_DEFAULT_DATA	Integer	0, 1	Determines whether to use default data for the memory
15	C_DEFAULT_DATA	String	"…"	Defines a default data for the memory
16	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
17	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
18	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
19	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
20	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
21	C_WRITE_WIDTH_A	Integer	1 to 1152	Defines width of write port A
22	C_READ_WIDTH_A	Integer	1 to 1152	Defines width of read port A
23	C_WRITE_DEPTH_A	Integer	2 to 9011200	Defines depth of write port A
24	C_READ_DEPTH_A	Integer	2 to 9011200	Defines depth of read port A
25	C_ADDRA_WIDTH	Integer	1 to 24	Defines the width of address A
26	C_WRITE_MODE_A	String	Write_First, Read_first, No_change	Defines the write mode for port A
27	C_HAS_ENA	Integer	0, 1	Determines whether port A has an enable pin
28	C_HAS_REGCEA	Integer	0, 1	Determines whether port A has an enable pin for its output register
29	C_HAS_RSTA	Integer	0, 1	Determines whether port A has reset pin
30	C_INITA_VAL	String	"…"	Defines initialization/power-on value for port A output

Table 25: **SIM Parameters *(Cont'd)***

	SIM Parameter	Type	Values	Description
31	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
32	C_WEA_WIDTH	Integer	1 to 128	Defines width of WEA pin for port A
33	C_WRITE_WIDTH_B	Integer	1 to 1152	Defines width of write port B
34	C_READ_WIDTH_B	Integer	1 to 1152	Defines width of read port B
35	C_WRITE_DEPTH_B	Integer	2 to 9011200	Defines depth of write port B
36	C_READ_DEPTH_B	Integer	2 to 9011200	Defines depth of read port B
37	C_ADDRB_WIDTH	Integer	1 to 24	Defines the width of address B
38	C_WRITE_MODE_B	String	Write_First, Read_first, No_change	Defines the write mode for port B
39	C_HAS_ENB	Integer	0, 1	Determines whether port B has an enable pin
40	C_HAS_REGCEB	Integer	0, 1	Determines whether port B has an enable pin for its output register
41	C_HAS_RSTB	Integer	0, 1	Determines whether port B has reset pin
42	C_INITB_VAL	String	"…"	Defines initialization/power-on value for port B output
43	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
44	C_WEB_WIDTH	Integer	1 to 128	Defines width of WEB pin for port B
45	C_USE_ECC	Integer	0,1	For Virtex-6 and Virtex-5 FPGAs only. Determines ECC options: • 0 = No ECC • 1 = ECC
46	C_RST_TYPE	String	(["SYNC", "ASYNC"] : "SYNC")	Type of Reset – synchronous or asynchronous. This parameter applies only for Spartan-6 devices.
47	C_RST_PRIORITY_A	String	(["CE", "SR"] : "CE")	In the absence of output registers, this selects the priority between the RAM ENA and the RSTA pin. When using output registers, this selects the priority between REGCEA and the RSTA pin.

Table 25: **SIM Parameters *(Cont'd)***

	SIM Parameter	Type	Values	Description
48	C_RSTRAM_A	Integer	([0,1] : 1)	Applicable for Spartan-3A DSP, Spartan-6 and Virtex-6 devices. If the value of this generic is 1, both the memory latch and the embedded primitive output register of Port A are reset. If this value is 0, then for Spartan-3A DSP and Spartan-6 devices, the primitive output register is built out of fabric, and only the output register is reset (the memory latch is not reset). If this value is 0 for Virtex-6 devices, then only the embedded output register of the primitive is reset (the memory latch is not reset). Setting this option to 1 results in the output reset value being asserted for two clock cycles.
49	C_RST_PRIORITY_B	String	(["CE", "SR"] : "CE")	In the absence of output registers, this selects the priority between the RAM ENB and the RSTB pin. When using output registers, this selects the priority between REGCEB and the RSTB pin.
50	C_RSTRAM_B	Integer	([0,1] : 1)	Applicable for Spartan-3A DSP, Spartan-6 and Virtex-6 devices. If the value of this generic is 1, both the memory latch and the embedded primitive output register of Port B are reset. If this value is 0, then for Spartan-3A DSP and Spartan-6 devices, the primitive output register is built out of fabric, and only the output register is reset (the memory latch is not reset). If this value is 0 for Virtex-6 devices, then only the embedded output register of the primitive is reset (the memory latch is not reset). Setting this option to 1 results in the output reset value being asserted for two clock cycles.
51	C_HAS_INJECTERR	Integer	([0,1,2,3] : 0)	For Virtex-6 FPGAs only. Determines the type of error injection: • 0 = No Error Injection • 1 = Single Bit Error Injection Only • 2 = Double Bit Error Injection Only • 3 = Both Single and Double Bit Error Injection

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. The configurations described in the sections that follow are separated into these sections:

- Virtex-6, Virtex-5, Virtex-4 Devices
- Spartan-6 and Spartan-3A DSP Devices
- Spartan-3 Devices and Implementations

Figure 38 shows the Optional Output Registers section of the Block Memory Generator GUI.

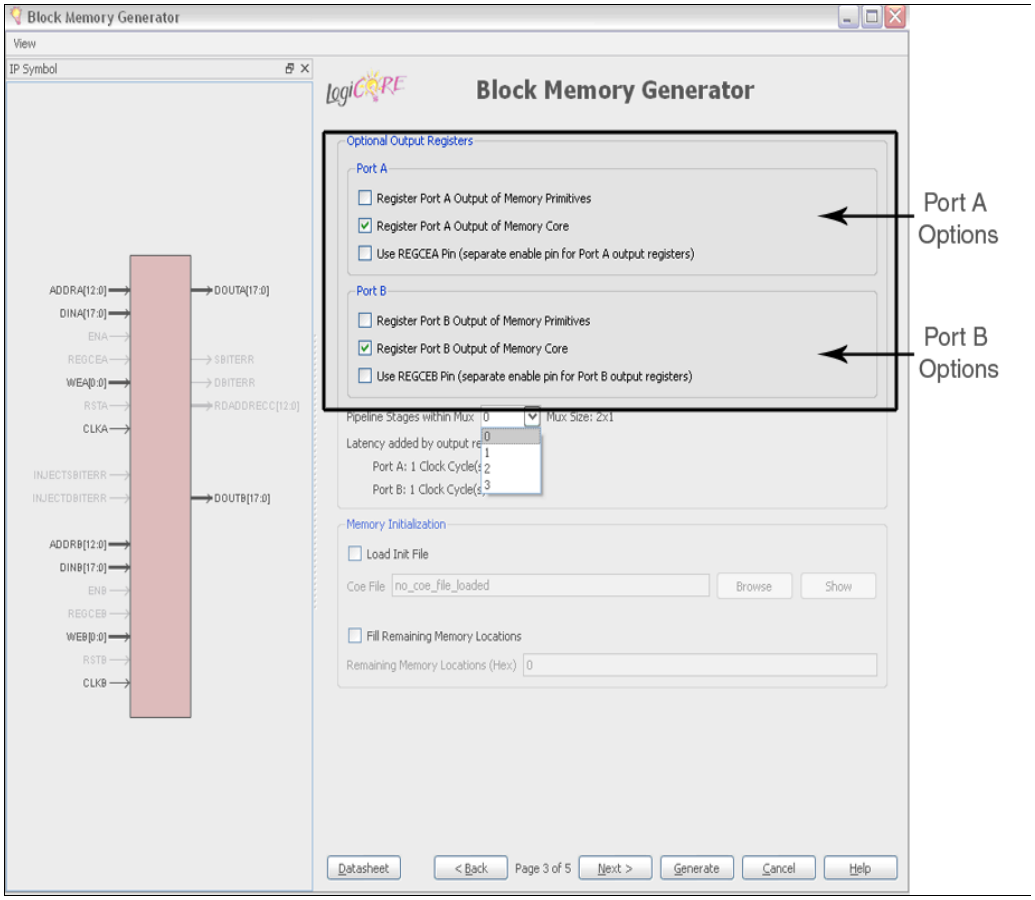

Figure 38: **Optional Output Registers Section**

Virtex-6, Virtex-5 and Virtex-4 FPGA: Output Register Configurations

To tailor register options for Virtex-6, Virtex-5 and Virtex-4 FPGA configurations, two selections for port A and two selections for port B are provided on Screen 3 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. Figure 39 through Figure 46 illustrate the Virtex-6, Virtex-5 and Virtex-4 FPGA output register configurations.

For Virtex-6, when only Register Port [A | B] Output of Memory Primitives and the corresponding Use RST[A | B] Pin are selected, the special reset behavior (option to reset the memory latch, in addition to the primitive output register) becomes available. For more information on this reset behavior, see "Special Reset Behavior," page 23.

VIrtex-6, Virtex-5 and Virtex-4 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 the Virtex-6, Virtex-5 or Virtex-4 FPGA embedded output registers and a register on the output of the core for the selected port(s), as shown in Figure 39. This configuration may provide improved performance for building large memory constructs.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☑ Register Port A Output of Memory Core		☑ Register Port B Output of Memory Core

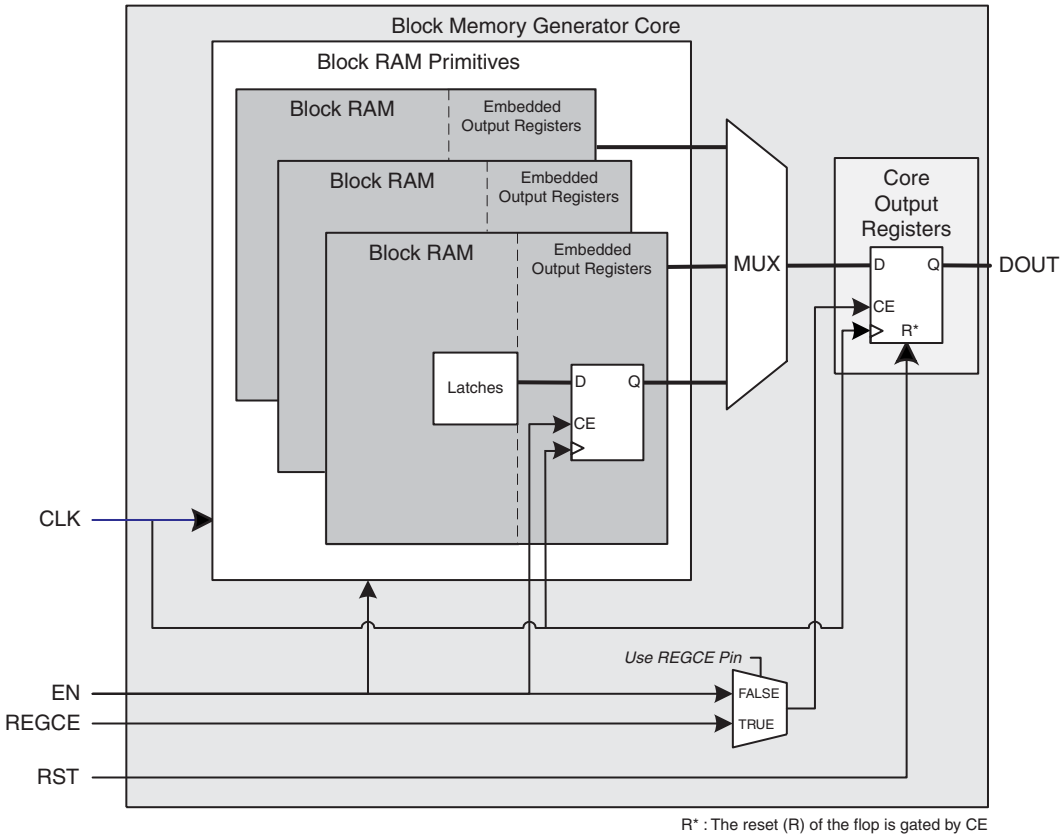

Figure 39: **Virtex-6, Virtex-5, or Virtex-4 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Enabled**

Virtex-6 FPGA: Memory with Primitive Output Registers and without Special Reset Behavior option

If Use RSTA Pin (set/reset pin) or Use RSTB Pin (set/reset pin) is selected, and the special reset behavior (to reset the memory latch besides the primitive output register) is not selected, then the input reset signal is only connected to the RSTREG pin of the Virtex-6 device block RAM primitive, as illustrated in Figure 40.

Note: This will result in reset similar to that of Spartan-3, Spartan-3A, Virtex-5 and Virtex-4 devices.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core
☑ Use RSTA Pin (set/reset pin)		☑ Use RSTB Pin (set/reset pin)
☐ Reset Memory Latch		☐ Reset Memory Latch

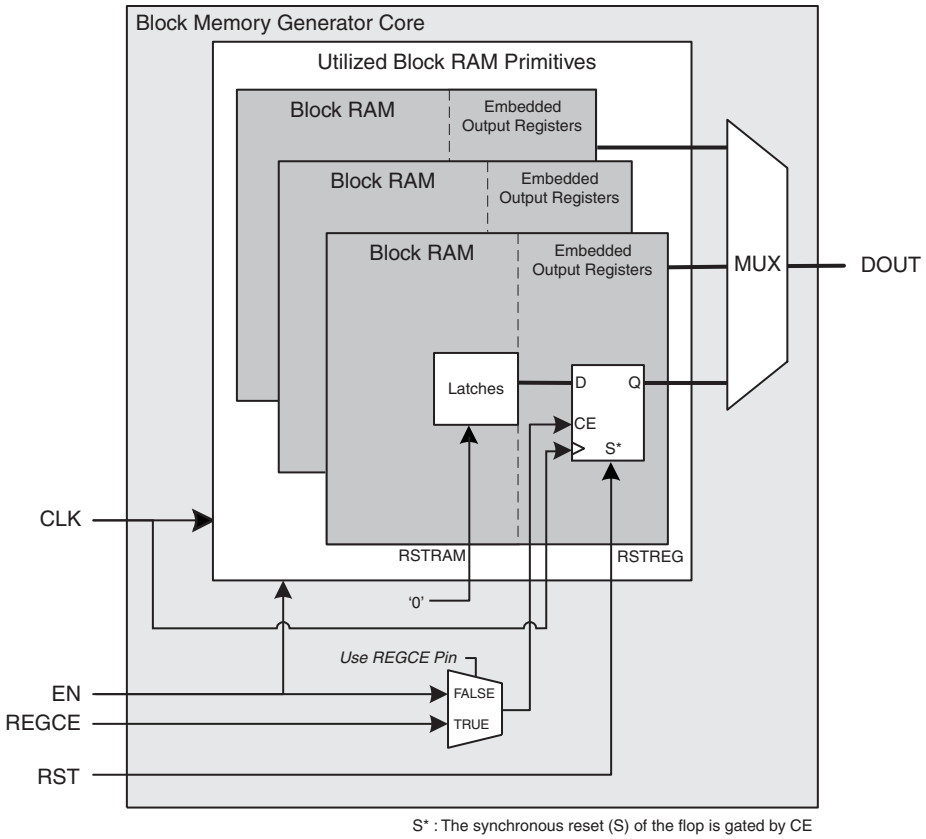

Figure 40: **Virtex-6 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled and without Special Reset Behavior**

Virtex-6 FPGA: Memory with Primitive Output Registers and with Special Reset Behavior option

When Register Port [A | B] Output of Memory Primitives, Use RSTA Pin (set/reset pin) or Use RSTB Pin (set/reset pin), and the special reset behavior (to reset the memory latch besides the primitive output register) are selected, then the input reset signal is connected to both the RSTRAM and RSTREG pins of the Virtex-6 device block RAM primitive, as illustrated in Figure 41.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core
☑ Use RSTA Pin (set/reset pin)		☑ Use RSTB Pin (set/reset pin)
☑ Reset Memory Latch		☑ Reset Memory Latch

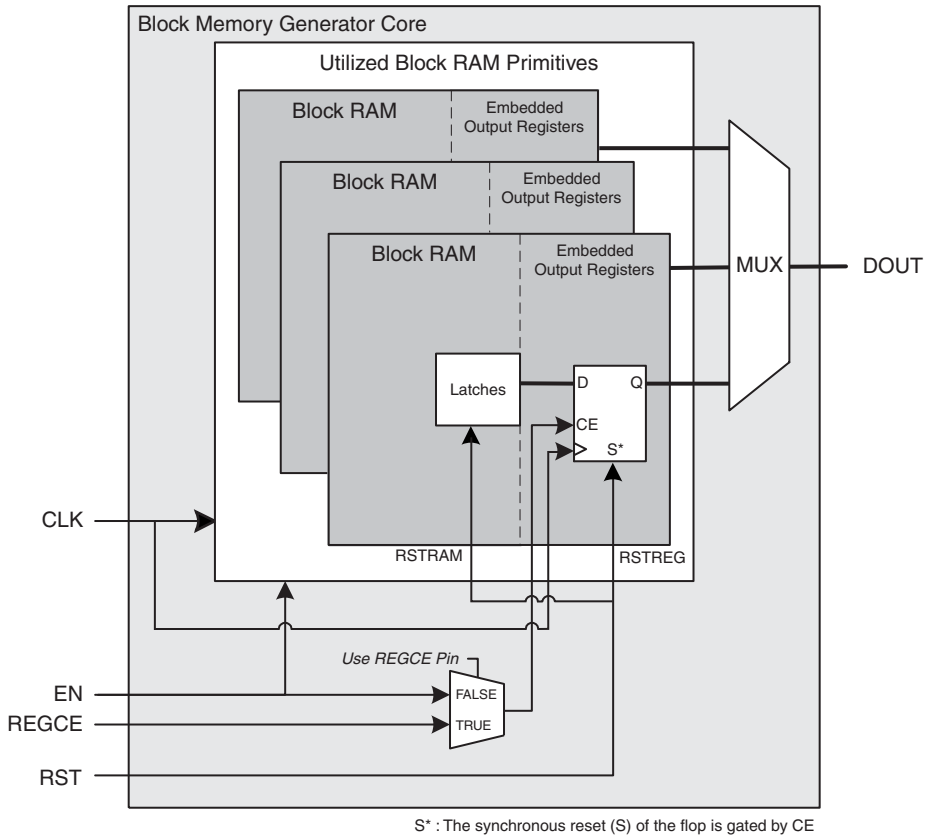

Figure 41: **Virtex-6 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled and with Special Reset Behavior**

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 42.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core

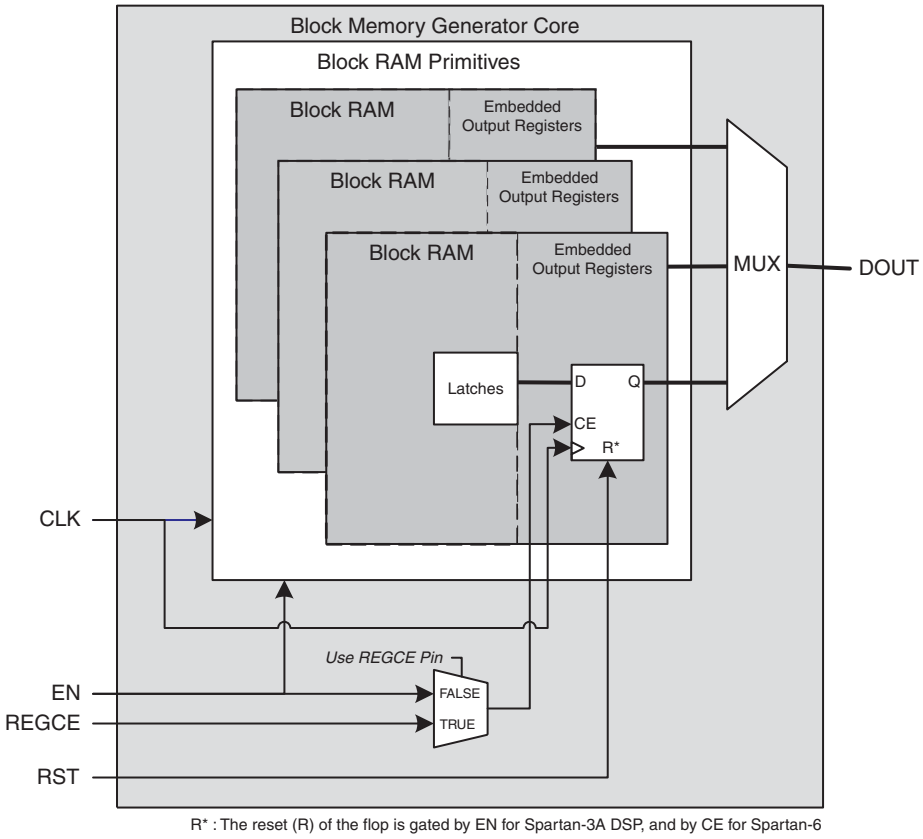

Figure 42: **Virtex-5 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled**

Virtex-4 FPGA: Memory with Primitive Output Registers without RST

When Register Port [A | B] Output of Memory Primitives is selected and the corresponding Use RST [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 43.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core
☐ Use RSTA Pin (set/reset pin)		☐ Use RSTB Pin (set/reset pin)

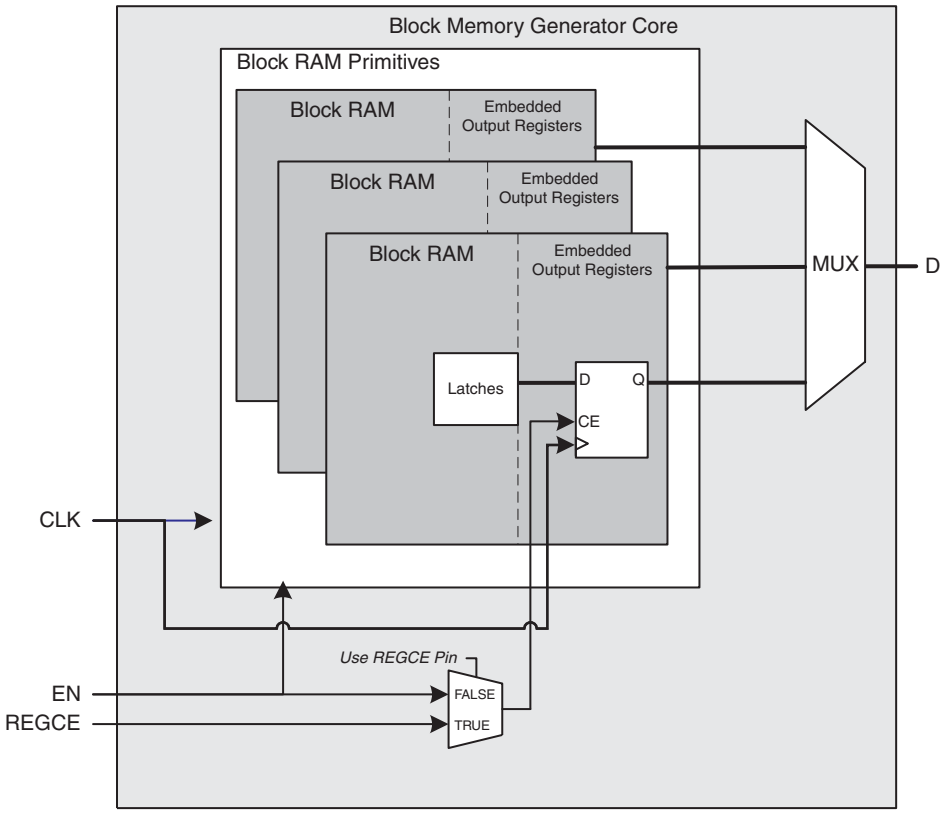

Figure 43: **Virtex-4 Block Memory Generated with only Register Port [A | B] Output of Memory Primitives Enabled**

Virtex-4 FPGA: Memory with Primitive Output Registers with RST

If either Use RSTA Pin (set/reset pin) or Use RSTB 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 44.

Port A	or	Port B
☑ Register A Output of Memory Primitives		☑ Register B Output of Memory Primitives
☐ Register A Output of Memory Core		☐ Register B Output of Memory Core
☑ Use RSTA Pin (set/reset pin)		☑ Use RSTB Pin (set/reset pin)

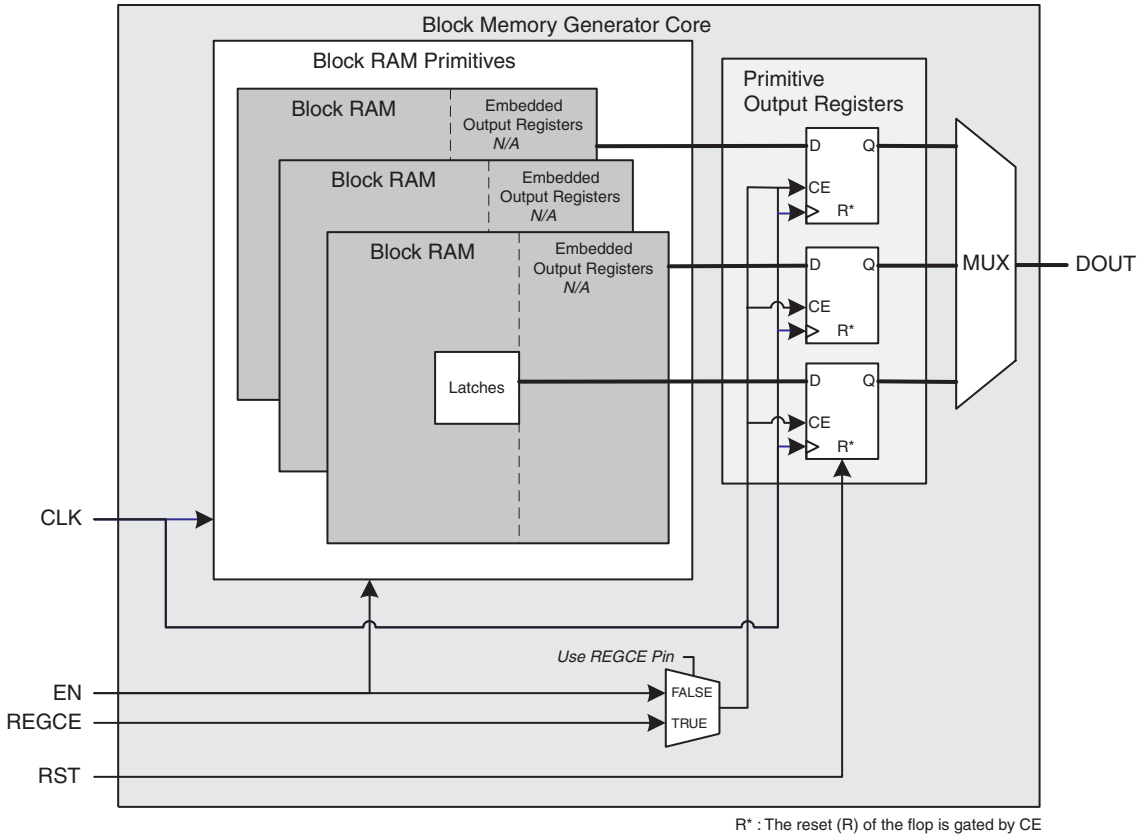

Figure 44: **Virtex-4 Block Memory Generated with Register Output of Memory Primitives and Use RST[A|B] Pin Options Enabled**

Virtex-6, 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-6/Virtex-5/Virtex-4 device's embedded registers are disabled for the selected ports in the generated core, as shown in Figure 45.

Port A	or	Port B
☐ Register Port A Output of Memory Primitives		☐ Register Port B Output of Memory Primitives
☑ Register Port A Output of Memory Core		☑ Register Port B Output of Memory Core

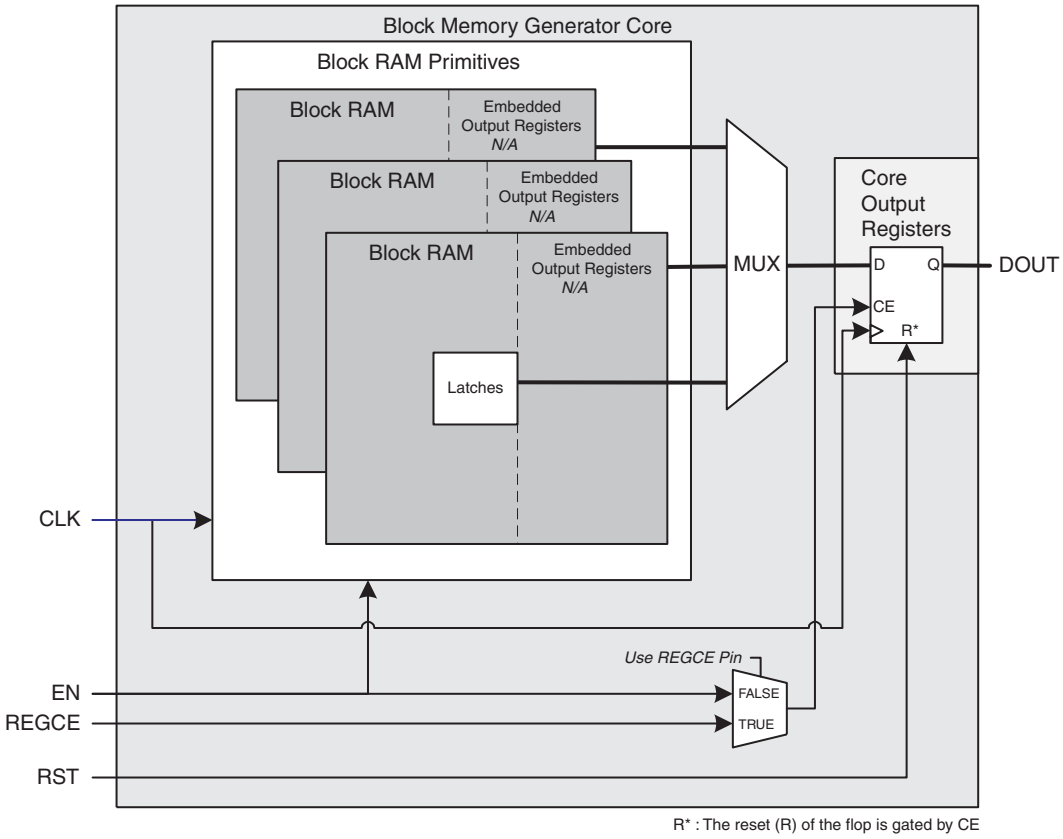

Figure 45: **Virtex-6, Virtex-5, or Virtex-4 FPGA Block Memory Generated with Register Port [A|B] Output of Memory Core Enabled**

Virtex-6, 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 46, there are no additional clock cycles of latency, but the clock-to-out delay for a read operation can impact design performance.

Port A	or	Port B
☐ Register Port A Output of Memory Primitives		☐ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core

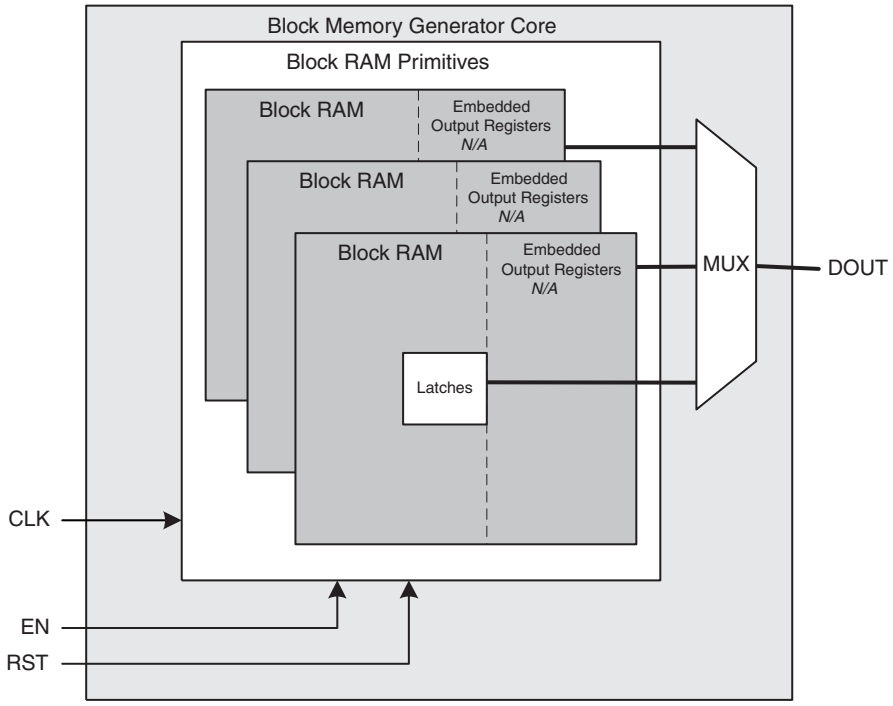

Figure 46: **Virtex-6, Virtex-5 or Virtex-4 Block Memory Generated with No Output Registers Enabled**

Spartan-6 or Spartan-3A DSP FPGA: Output Register Configurations

To tailor register options for Spartan-6 or Spartan-3A DSP device configurations, two selections for port A and two selections for port B are provided on screen 3 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. Figure 47 through Figure 52 illustrate the Spartan-6 or Spartan-3A DSP output register configurations.

When only Register Port [A | B] Output of Memory Primitives and the corresponding Use RST[A | B] Pin (set/reset pin) is selected, the special reset behavior (option to reset the memory latch besides the primitive output register) becomes available. This option is displayed as the **Reset Memory Latch** option on the Spartan-6 and Spartan-3A DSP GUI. Selecting this option forces the core to use the Spartan-6 or Spartan-3A DSP embedded output registers, but changes the behavior of the core. For detailed information, see the sections that follow.

Spartan-6 or 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 embedded output registers and a register on the output of the core for the selected port(s), as shown in Figure 47. This configuration may improve performance when building a large memory construct.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☑ Register Port A Output of Memory Core		☑ Register Port B Output of Memory Core

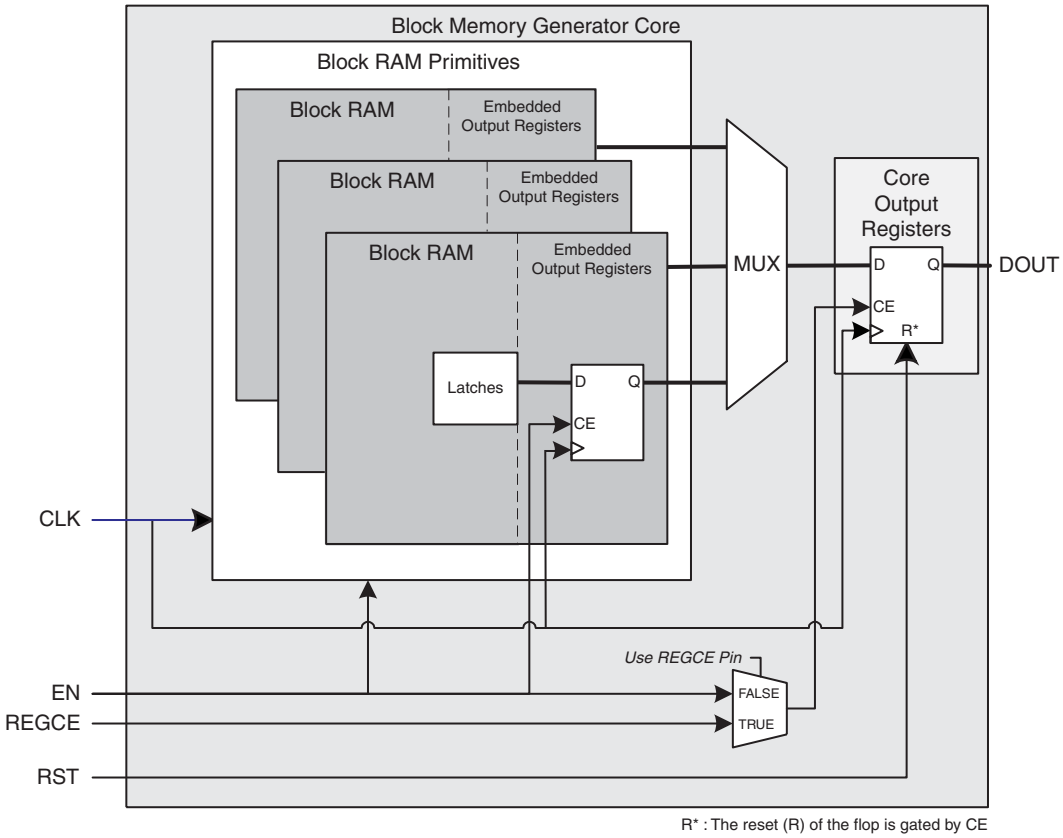

Figure 47: **Spartan-6 or Spartan-3A DSP Block Memory Generated with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Enabled**

Spartan-6 or Spartan-3A DSP FPGA: Memory With Primitive Output Registers – Without RST Pin

When Register Port [A|B] Output of Memory Primitives is selected, and the corresponding Use RST 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-6 and Spartan-3A DSP FPGA architectures is generated. The output of any multiplexing that may be required to combine multiple primitives is not registered in this configuration (Figure 48).

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core
☐ Use RSTA Pin (set/reset pin)		☐ Use RSTB Pin (set/reset pin)

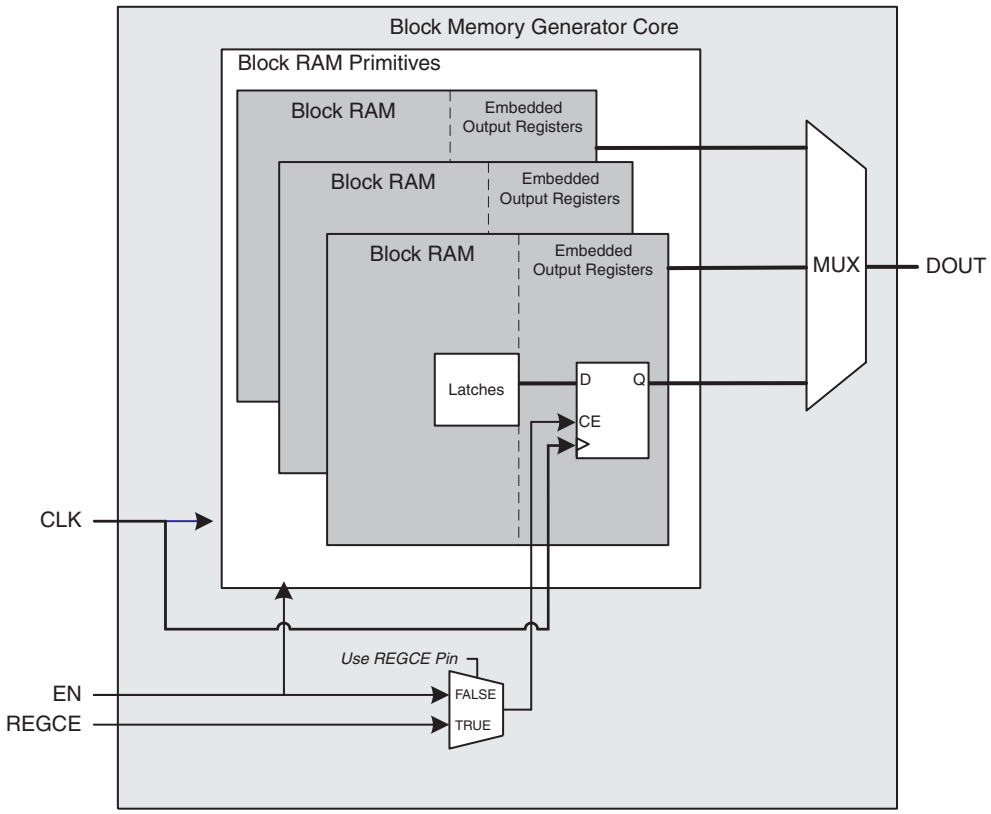

Figure 48: **Spartan-6 or Spartan-3A DSP Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled (No RST)**

Spartan-6 or Spartan-3A DSP FPGA: Memory with Primitive Output Registers and without Special Reset Behavior Option

If Use RSTA Pin (set/reset pin) or Use RSTB Pin (set/reset pin) is selected, and the Reset Behavior option (resets the memory latch in addition to the primitive output register) is not selected, the embedded block RAM registers of the Spartan-6 or Spartan-3ADSP device cannot be used.The primitive output registers are built from FPGA fabric, as illustrated in Figure 49.

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

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core
☑ Use RSTA Pin (set/reset pin)		☑ Use RSTB Pin (set/reset pin)
☐ Reset Memory Latch		☐ Reset Memory Latch

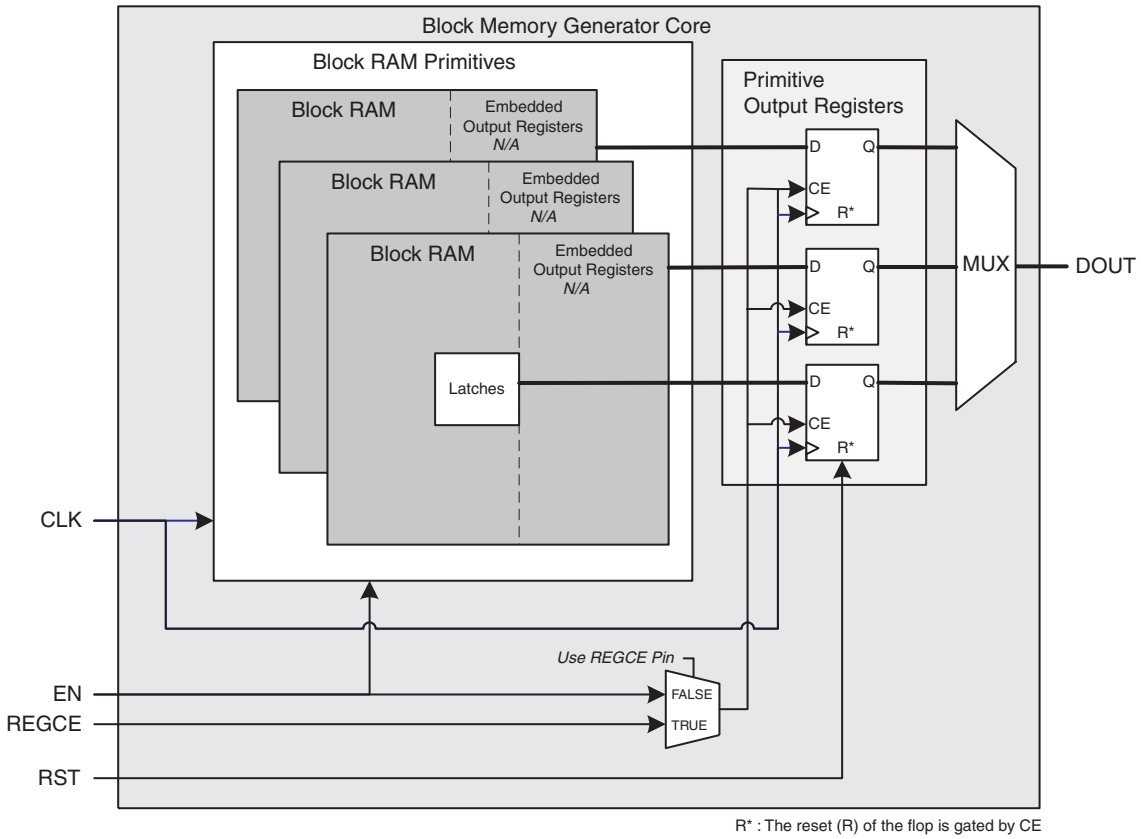

Figure 49: **Spartan-6 or Spartan-3A DSP Block Memory Generated with Register Port [A|B] Output of Memory Primitives, Use RST[A|B] Pin Options (With RST), without Special Reset Behavior**

Spartan-6 or Spartan-3A DSP FPGA: Memory with Primitive Output Registers and with Special Reset Behavior Option (Embedded Registers)

When Register Port [A | B] Output of Memory Primitives, Use RSTA Pin (set/reset pin) or Use RSTB Pin (set/reset pin), and the special reset behavior (resets the memory latch in addition to the primitive output register) are selected, the Spartan-6 or Spartan-3A DSP embedded registers are enabled for the selected port in the generated core, as displayed in Figure 50.

If the special reset behavior option is selected, the Spartan-6 or Spartan-3A DSP FPGA's embedded output registers are used, but the reset behavior of the core changes as described in "Special Reset Behavior," page 23. The functional differences between this and other implementations are that the RST[A | B] input resets *both* the embedded output registers *and* the block RAM output latches.

For Spartan-3ADSP devices, 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, as shown in Figure 50. This differs from all other configurations of the Block Memory Generator where RST is typically gated by REGCE.

For Spartan-6 devices, if REGCE is held high, the output value is set to the reset value for two clock cycles following a reset. Unlike Spartan-3A DSP devices, and similar to other architectures, reset is gated by REGCE.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core
☑ Use RSTA Pin (set/reset pin)		☑ Use RSTB Pin (set/reset pin)
☑ Reset Memory Latch		☑ Reset Memory Latch

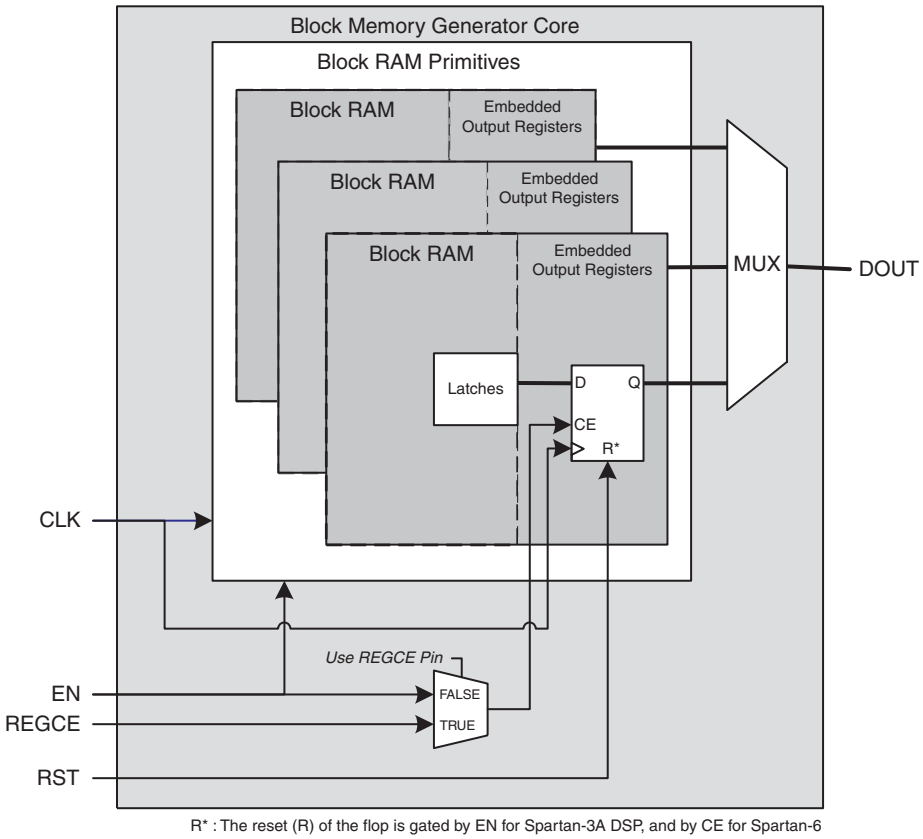

Figure 50: **Spartan-6 or Spartan-3A DSP Block Memory Generated with Register Port [A|B] Gated by EN in Spartan-3A DSP and by CE in Spartan-6 Output of Memory Primitives, Use RST[A|B] Pin Options (With RST), and Special Reset Behavior Enabled**

Spartan-6 or Spartan-3A DSP FPGA: Memory with Core Output Registers

When Register Port [A | B] Output of Memory Core is selected, the Spartan-6 or Spartan-3A DSP FPGA embedded registers are disabled in the generated core, as illustrated in Figure 51.

Port A	or	Port B
☐ Register Port A Output of Memory Primitives		☐ Register Port B Output of Memory Primitives
☑ Register Port A Output of Memory Core		☑ Register Port B Output of Memory Core
☑ Use RSTA Pin (set/reset pin)		☑ Use RSTB Pin (set/reset pin)

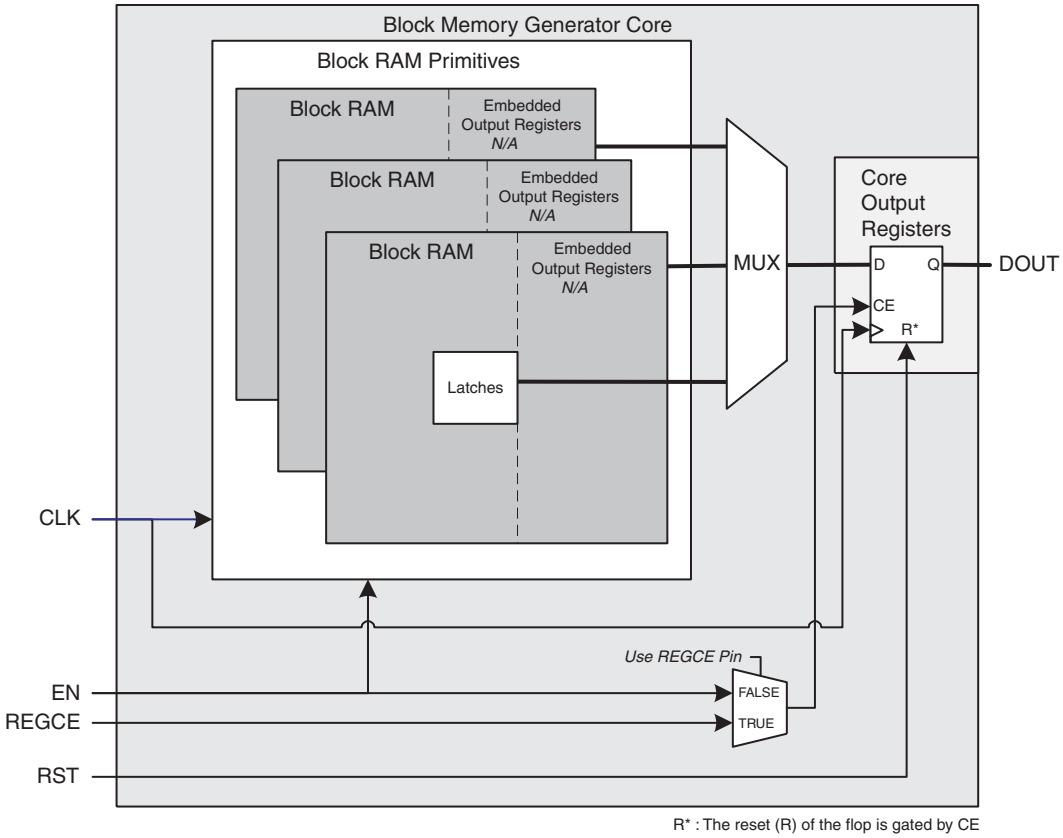

Figure 51: **Spartan-6 or Spartan-3A DSP Block Memory Generated with Register Port [A|B] Output of Memory Core Enabled**

Spartan-6 or 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 52, there are no additional clock cycles of latency, but the clock-to-out delay for a read operation can impact design performance.

Port A	or	Port B
☐ Register Port A Output of Memory Primitives		☐ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core

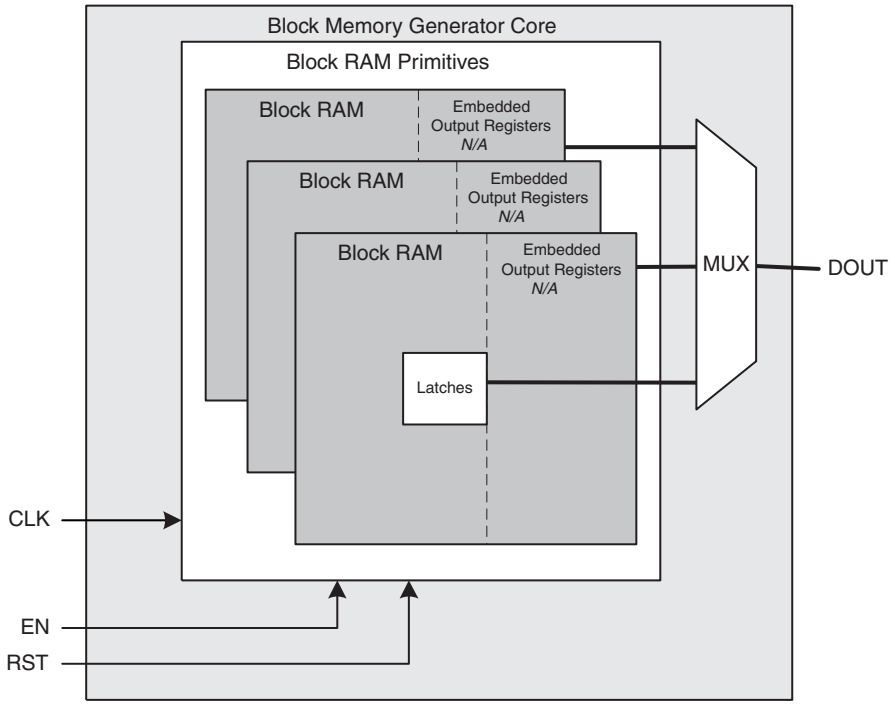

Figure 52: **Spartan-6 or Spartan-3A DSP Block Memory Generated with No Output Port Registers Enabled**

Spartan-3 FPGA: Output Register Configurations

To tailor register options for Spartan-3 FPGA architectures, two selections for port A and two selections for 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. Figure 53, Figure 54, Figure 55, and Figure 56 illustrate the Spartan-3 FPGA output register configurations.

Spartan-3 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 53. Selecting this configuration may provide improved performance for building large memory constructs.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☑ Register Port A Output of Memory Core		☑ Register Port B Output of Memory Core

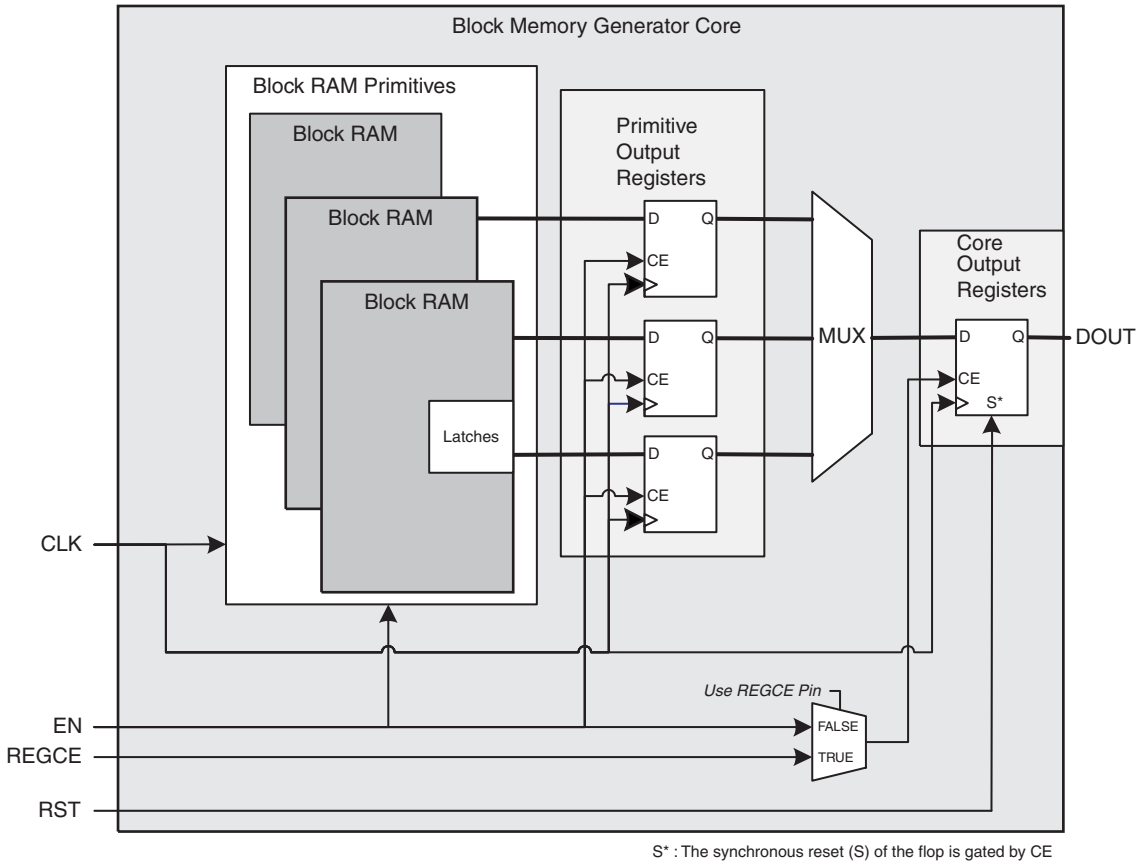

Figure 53: **Spartan-3 Block Memory Generated with Register Port [A|B] Output of Memory Primitives and Register Port [A|B] Output of Memory Core Options Enabled**

Spartan-3 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 54.

Port A	or	Port B
☑ Register Port A Output of Memory Primitives		☑ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core

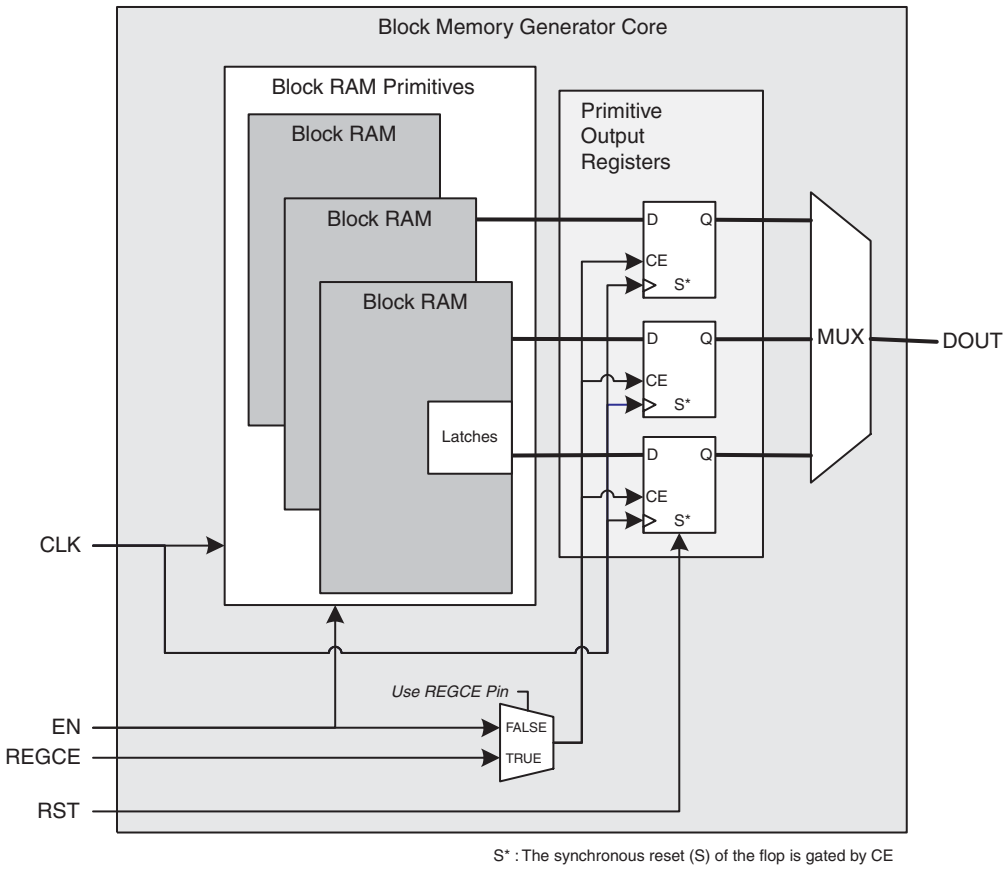

Figure 54: **Spartan-3 Block Memory Generated with Register Port [A|B] Output of Memory Primitives Enabled**

Spartan-3 FPGA: Memory with Core Output Registers

Figure 55 illustrates a memory configured with Register Port [A | B] Output of Memory Core selected.

Port A	or	Port B
☐ Register Port A Output of Memory Primitives		☐ Register Port B Output of Memory Primitives
☑ Register Port A Output of Memory Core		☑ Register Port B Output of Memory Core

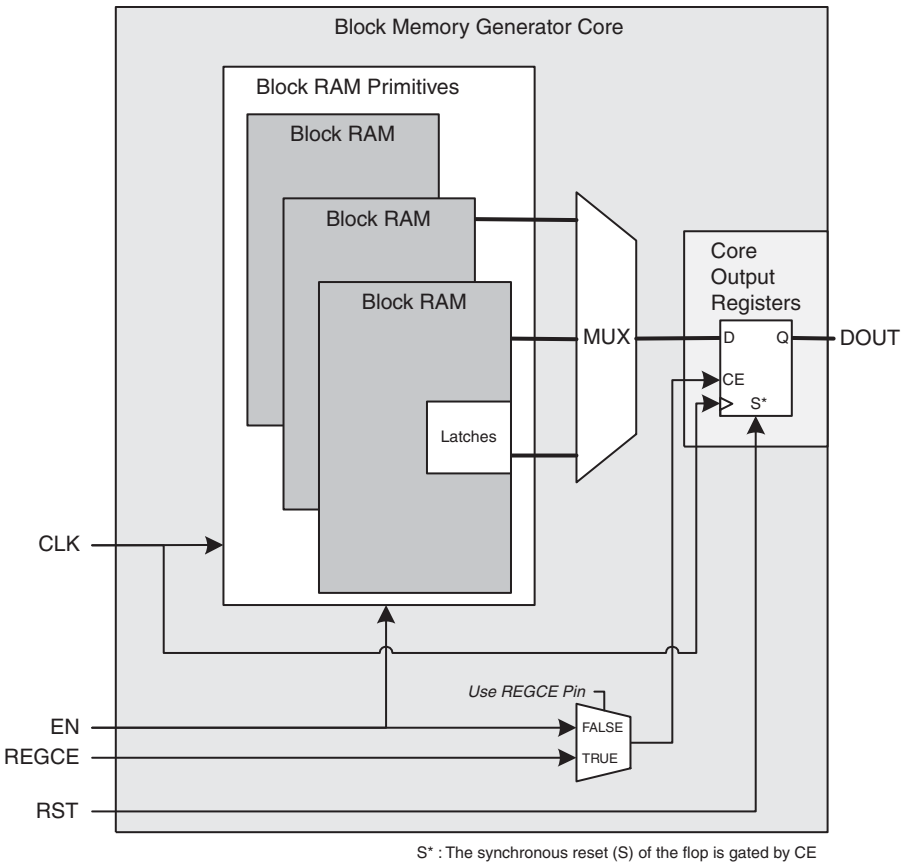

Figure 55: **Spartan-3 Block Memory Generated with Register Port [A|B] Output of Memory Core Enabled**

Spartan-3 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 56.

Port A	or	Port B
☐ Register Port A Output of Memory Primitives		☐ Register Port B Output of Memory Primitives
☐ Register Port A Output of Memory Core		☐ Register Port B Output of Memory Core

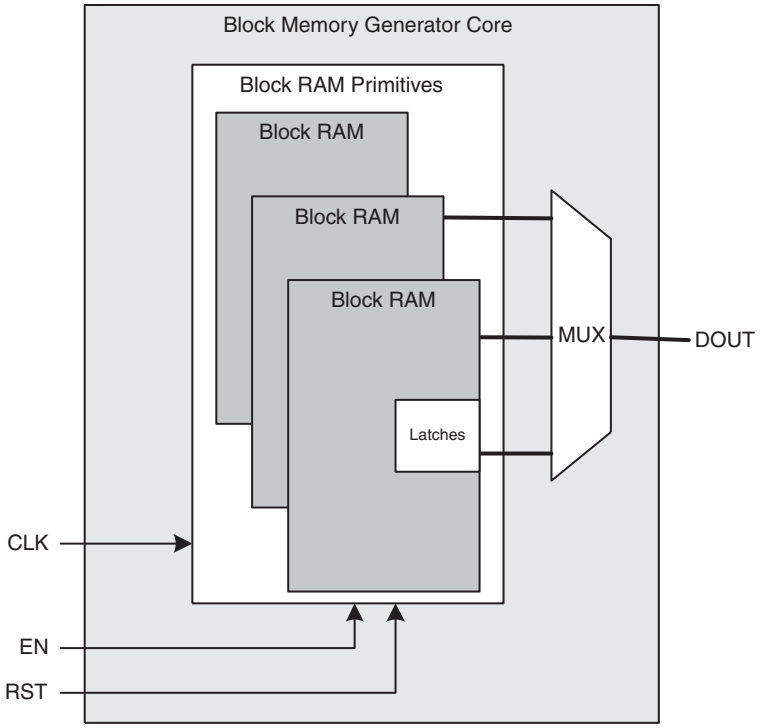

Figure 56: **Spartan-3 Block Memory Generated with No Output Registers Enabled**

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. See "Resource Utilization and Performance Examples," page 44 for more information.

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

The Block Memory Generator LogiCORE IP core is included free of charge with the Xilinx ISE Foundation Series Development software and is provided under the terms of the Xilinx End User License Agreement. The core can be generated using the ISE CORE Generator system v11.3 or higher.

For more information, please visit the Block Memory Generator core page.

Information about additional Xilinx LogiCORE modules is available at the Xilinx IP Center. For pricing and availability of other Xilinx LogiCORE modules and software, please contact your local Xilinx sales representative.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
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.

Date	Version	Description of Revisions
9/19/08	8.0	Updated core to version 2.8.
12/17/08	8.0.1	Early access documentation.
4/24/09	9.0	Updated core to version 3.1 and Xilinx tools to version 11.1.
6/24/09	10.0	Updated core to version 3.2 and Xilinx tools to version 11.2.
6/24/09	10.1	Updated "Resource Utilization and Performance Examples," page 44.
9/16/09	11.0	Updated core to version 3.3 and Xilinx tools to version 11.3.

Notice of Disclaimer