RAM Megafunction User Guide

101 Innovation Drive
San Jose, CA 95134
www.altera.com

UG-MF9404-3.0

Software Version:	8.1
Document Version:	3.0
Document Date:	December 2008

UG-MF9404-3.0

Contents

RAM Megafunction User Guide

RAM Megafunction User Guide

Introduction

Altera provides a diverse set of RAM modes to address the memory requirements of today's system-on-a-programmable-chip (SOPC) designs. Altera provides the RAM modes through parameterizable memory megafunctions that are optimized for Altera® device architecture. Using megafunctions instead of coding your own logic saves valuable design time and offers more efficient logic synthesis and device implementations.

There are several types of memory megafunctions, including the ALTSYNCRAM, ALTDPRAM, ALT3PRAM, and LPM_RAM_DQ megafunctions. The Quartus® II software automatically selects one of the megafunctions to implement the RAM functions. The selections depends on the target device, RAM modes, and features of the RAM functions.

You can use one of the following methods to implement the RAM function you desire:

- RAM MegaWizard plug-ins
- Memory inferring from HDL code

For more information about memory inferring from HDL code, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook.*

Altera recommends using the RAM MegaWizard plug-ins to configure and build your RAM function. The MegaWizard Plug-In Manager is a GUI that enables you to quickly set the options and ensure that the combination options are valid.

This user guide contains the following sections:

- Design Example with External ECC Implementation
- Ports and Parameters

RAM Modes and RAM MegaWizard Plug-Ins

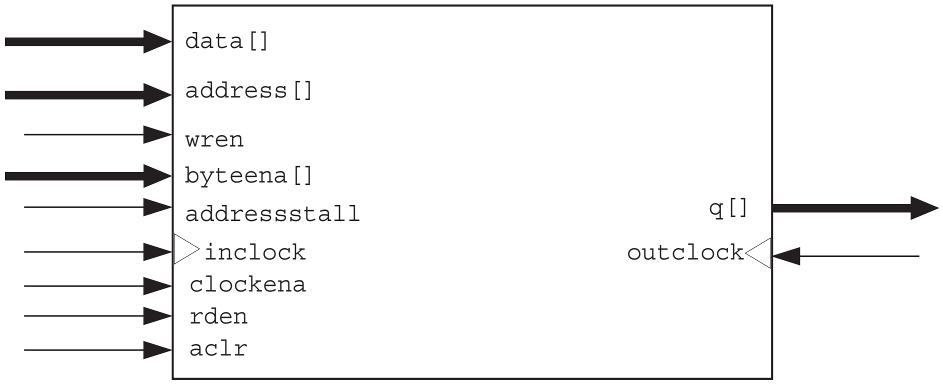
Altera provides the following RAM modes:

- Single-port RAM
- Simple dual-port RAM
- True dual-port RAM
- Tri-port RAM

In single-port RAM mode, the read operation and write operation share the same address at port A, and the data is read from the output of port A. The single-port RAM can be configured and built through the RAM:1-PORT MegaWizard Plug-In.

Figure 1 shows a block diagram for a typical single-port RAM.

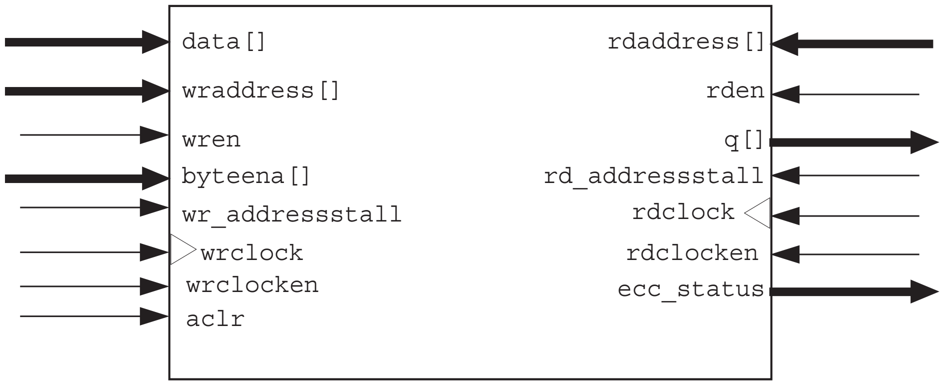
Figure 1. Single-Port RAM



In simple dual-port RAM mode, a dedicated address port is available for each read operation and write operation (one read port and one write port). Write operation uses write address of port A while read operation uses read address and output of port B. The simple dual-port RAM can be configured and built through the RAM:2-PORT MegaWizard Plug-In.

Figure 2 shows the block diagrams for a typical simple dual-port RAM.

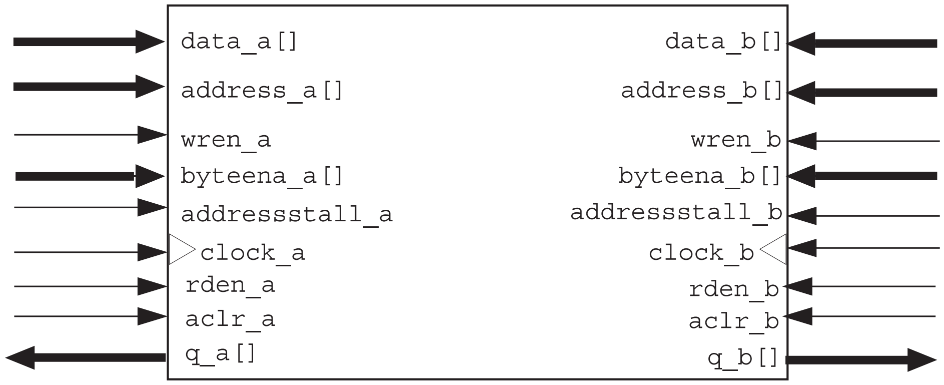
Figure 2. Simple Dual-Port RAM



In true dual-port RAM mode, two address ports are available and can be used for read operation or write operation (two read and write ports). In this mode, you can write to or read from the address of port A or port B, and the data read is shown at the output port with respect to the read address port. The true-dual port RAM also can be configured and built through the RAM:2-PORT MegaWizard Plug-In.

Figure 3 shows the block diagrams for a typical true dual-port RAM.

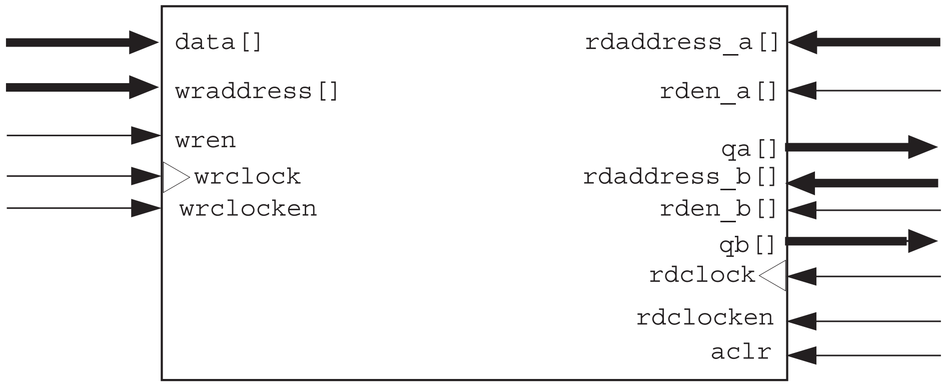
Figure 3. True Dual-Port RAM



In tri-port RAM mode, two read address ports and one write address port are available (two read ports and one write port). The tri-port RAM can be configured and built through the RAM:3-PORT MegaWizard Plug-In.

Figure 4 shows the block diagrams for a typical tri-port RAM.

Figure 4. Tri-Port RAM



Memory Block Types

Altera's embedded memory features the TriMatrix memory structure that provides three different sizes of embedded SRAM. Different devices support different sizes of the TriMatrix embedded memory blocks. Table 1 shows the TriMatrix memory blocks in different Altera devices.

Table 1. TriMatrix Embedded Memory Blocks in Altera Devices

Devices	Types of TriMatrix Memory Blocks
Stratix®, Stratix GX, Stratix II, Stratix II GX, Cyclone®, Cyclone II, and Arria® GX *(1)*	M512 blocks (512 bits) M4K blocks (4 Kbits) M-RAM blocks (512 Kbits)
Stratix III, Stratix IV, and Cyclone III *(2)*	MLAB blocks (640 bits) *(3)* M9K blocks (9 Kbits) M144K blocks (144 Kbits)

Notes to Table 1:

(1) Cyclone and Cyclone II devices have M4K blocks only.
(2) Cyclone III devices have M9K blocks only.
(3) For Stratix III devices, MLAB blocks are 320-bit in RAM mode and 640-bit in ROM mode.

For more information about TriMatrix memory blocks and the specification, refer to the *TriMatrix Embedded Memory Blocks* chapter in your target device handbook.

From the RAM MegaWizard plug-ins, you can implement the RAM in any of the supported TriMatrix memory blocks available based on your target device. You can also choose to implement the RAM using logic cells, or allow the software to automatically select the appropriate TriMatrix memory resource.

If you want to specifically select the TriMatrix memory block, first get more information about the TriMatrix memory features such as maximum performance, supported configurations (depth × width), byte enable, power-up condition, and write and read operation triggering, for the type of TriMatrix memory block you selected on your target device.

☞ In true-dual port RAM mode, the RAM cannot be implemented using logic cells, M512, or MLAB.

As compared to TriMatrix memory resources, using logic cells to implement your RAM functions reduces the design performance and utilize more area. This implementation is normally used when you use up all the TriMatrix memory resources. When using logic cells, the wizard provides you with two types of logic cell implementations:

- Default logic cell style—the write operation triggering (internally) on the rising edge of the write clock and having continuous read. This implementation uses less logic cells and is faster, but it is not fully compatible with the Stratix M512 emulation style.
- Stratix M512 emulation logic cell style—the write operation triggering (internally) on the falling edge of the write clock and read only on the rising edge of the read clock.

To get the proper implementation based on the RAM configuration you set, let the Quartus II software automatically choose the memory type. This gives the compiler the flexibility to place the memory function in any available memory resource based on the functionality and size.

☞ To identify the type of memory block the software selected to implement your RAM, refer to the fitter report after compilation.

When the RAM block type is set to **Auto**, the compiler favors larger block types that can support the memory capacity you require in a single RAM block. This gives the best performance and requires no logic elements (LEs) for glue logic. When you choose to implement the RAM with specific TriMatrix memory blocks, such as M9K, the compiler is still able to emulate wider and deeper memories than the block type supported natively. The compiler spans multiple RAM blocks with glue logic added in LEs as needed.

Write and Read Operations Triggering

The TriMatrix memory blocks vary slightly in its supported features and behaviors. One important variation is the different write and read operations triggering for different types of TriMatrix memory blocks.

Table 2 shows the write and read operations triggering for different TriMatrix memory blocks.

Table 2. Write and Read Operations Triggering for TriMatrix Memory Blocks

TriMatrix Memory Blocks	Write Operation	Read Operation
M144K	Rising clock edges	Rising clock edges
M9K	Rising clock edges	Rising clock edges
MLAB	Falling clock edges	Rising clock edges
M-RAM	Rising clock edges	Rising clock edges
M4K	Falling clock edges	Rising clock edges
M512	Falling clock edges	Rising clock edges

Understanding the write operation triggering is crucial to avoid potential write contentions that can result in unknown data storage at that location.

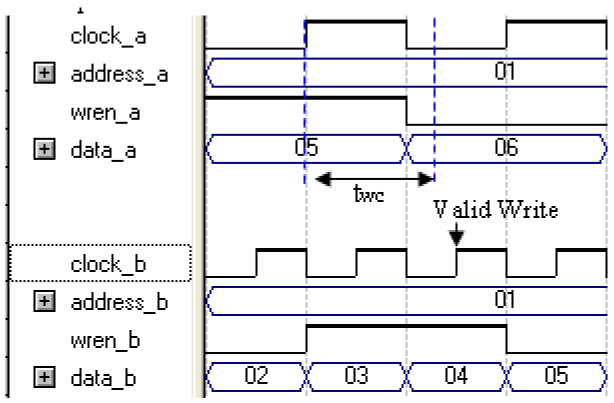
Table 3 shows the comparison of criteria of a valid write operation to the same address for memory block with write operation triggers at rising clock edges and falling clock edges.

Table 3. Valid Write Operation Criteria to the Same Address for Write Operation Triggers at Rising Clock Edges and Falling Clock Edges

Rising Clock Edges	Falling Clock Edges
The rising edge of the write clock for a port occurs following the maximum write cycle time interval and after the rising edge of write clock for another port.	The rising edge of the write clock for a port occurs following half of the maximum write cycle time interval and after the falling edge of the write clock for another port.

Example 1 shows a write operation for rising clock edges.

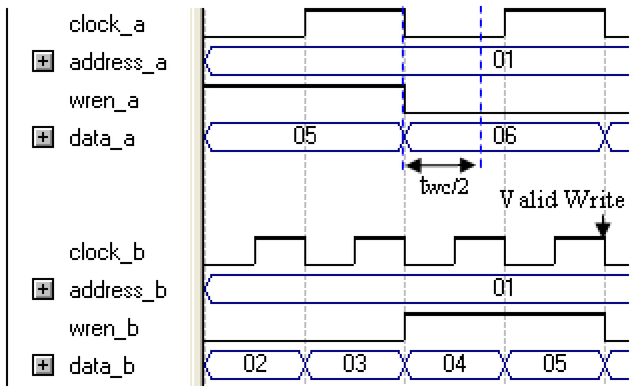
Example 1. Write Operation for Rising Clock Edges



Assuming t_{wc} the maximum write cycle time interval. Write operation of data 03 through port B does not meet the criteria and therefore causes write contention with the write operation at port A that result in unknown data at address 01. The write operation at the next rising edge is valid as it meets the criteria and the data 04 replaces the unknown data.

Example 2 shows a write operation for falling clock edges.

Example 2. Write Operation for Falling Clock Edges



Assuming $t_{wc/2}$ half the maximum write cycle time interval. Write operation of data 04 through port B does not meet the criteria and therefore causes write contention with the write operation at port A that result in unknown data at address 01. The next data 05 is latched at the next rising clock edge that meets the criteria and written into the memory block at the falling clock edge.

Regardless of the different write operation triggering, data, and addresses are latched at the rising edge of the write clock.

Port Width Configuration

The port width configuration for memory is defined as the memory depth (number of words) × the width of the data input bus.

Table 4 shows the supported port width configuration (per memory block) for TriMatrix memory blocks in Stratix III devices.

Table 4. Port Width Configuration for TriMatrix Memory Blocks in Stratix III Devices

MLABs	M9K	M144K
16 × 8	8 K × 1	16 K × 8
16 × 9	4 K × 2	16 K × 9
16 × 10	2 K × 4	8 K × 16
16 × 16	1 K × 8	8 K × 18
16 × 18	1 K × 9	4 K × 32
16 × 20	512 × 16	4 K × 36
—	512 × 18	2 K × 64
—	256 × 32	2 K × 72
—	256 × 36	—

For more information about the supported port width configuration for different TriMatrix memory blocks, refer to the *TriMatrix Embedded Memory Blocks* chapter in your target device handbook.

If your port width configuration (either the depth or the width) is more than the amount a TriMatrix memory block can support, additional memory blocks (of the same type) are used. For example, if you configure your M9K as 512 × 36 (exceeds the supported port width of 512 × 18), two M9Ks are used to implement your RAM.

In addition to the supported configuration provided, you can set the memory depth to a non-power of two, but the actual memory depth allocated can be varied. The variation depends on the type of resource implemented.

If the RAM is implemented in TriMatrix memory blocks, setting a non-power of two for the memory depth reflects the actual memory depth. If the RAM is implemented in logic cells (and not using Stratix M512 emulation logic cell style that can be set through the RAM MegaWizard plug-ins), setting a non-power of two for the memory depth does not reflect the actual memory depth. In this case, you write to or read from up to $2^{\texttt{address\_width}}$ memory locations even though the memory depth you set is less than $2^{\texttt{address\_ width}}$. For example, if you set the memory depth to 3, and the RAM is implemented using logic cells, your actual memory depth is 4.

You should confirm the actual memory depth implemented by referring to the fitter report or the design variation file generated when you implement the depth of your RAM in a non-power of two.

Mixed Width Port

The simple dual-port and true dual-port RAM support mixed width port configuration and the valid ratio between port A and port B width are 1, 2, 4, 8, 16, and 32.

Memory depth of 1 word is not supported by the ALTSYNCRAM megafunction in simple dual-port and true dual-port RAM with mixed-width port. The RAM MegaWizard plug-in prompts you an error message when the memory depth is less than 2.

If you manually change the memory depth to 1 word from the generated wrapper file, the Quartus II's Assembler displays an internal error message stating **rams with 1 word are not allowed in Quartus 8.1**. Note that for the Quartus II version before 8.1, you do not get an internal error message but the generated **.pof** file is improper.

For the Quartus II software version 9.0, memory depth of 1 word is not natively supported by the megafunction. But if you manually change the memory depth to less than 2, the Assembler is able to generate the correct **.pof** file.

Maximum Block Depth

You can limit the maximum block depth of the TriMatrix memory block you use. The memory block can be sliced to have the maximum block depth you want. For example, the capacity of a M9K block is 9,216 bits, and by default, has the memory depth of 8K, in which each address is capable of storing 1 bit (8K × 1). If you set the maximum block depth to 512, it slices the M9K block to have a depth of 512 and each address is capable of storing up to 18 bits (512 × 18).

You can use this option to save power in your devices. However, this parameter may increase the number of LEs and affect design performance.

Table 5 shows the estimated power usage settings for an 8K × 36 (M9K RAM block) design of a Stratix III EP3SE50 device.

Table 5. Power Usage Setting for 8K × 36 (M9K) Design of a Stratix III Device

M9K Slice Type	Dynamic Power (mW)	ALUT Usage	M9Ks
8K × 1 (default setting)	51.49	0	36
4K × 2	20.28 (39%)	38	36
2K × 4	10.80 (21%)	44	36
1K × 9	6.08 (12%)	125	32
512 × 18	4.51 (9%)	212	32
256 × 36	6.36 (12%)	467	32

As the RAM is sliced shallower, the dynamic power usage decreases. For a 256 deep RAM block, the power used by the extra LEs starts to outweigh the power gain achieved by shallower slices.

You can also use this option to reduce the total number of memory blocks used (but at the expense of LEs). From Table 5, the 8K × 36 RAM uses 36 M9K RAM blocks with a default slicing of 8K × 1 slices. By setting the maximum block depth to 1K, the 8K × 36 RAM can fit into 32 M9K blocks.

The maximum block depth must be in a power of two, and the valid values vary between different TriMatrix memory blocks.

Table 6 shows the valid range for the maximum block depth for different TriMatrix memory blocks.

Table 6. Valid Range for Maximum Block Depth for Different TriMatrix Memory Blocks

TriMatrix Memory Blocks	Valid Range (1)
M144K	4,096 – 13,1072
M9K	128 – 8,192
MLAB	32 – 64 (2)
M512	32 – 64
M4K	128 – 4,096
M-RAM	4,096 – 65,536

Notes to Table 6:

(1) The maximum block depth must be in a power of two.
(2) The maximum block depth setting for MLAB is not available for Stratix III devices.

The MegaWizard Plug-In Manager displays an error if you enter an invalid value for the maximum block depth. If you are not sure of the proper maximum block depth to set or the setting is not important for your design, you can let the compiler select the maximum block depth that has the proper port width configuration and type of TriMatrix memory block of your RAM.

Clocking Modes and Clock Enable

Altera RAM supports various types of clocking modes depending on the RAM mode you select.

Table 7 shows the supported types of clocking modes in most Altera devices.

Table 7. Supported Types of Clocking Modes in Most Altera Devices

Clocking Modes	Single-port RAM	Simple Dual-port RAM	True Dual-port RAM	Tri-port RAM
Single clock	✓	✓	✓	✓
Read/Write	—	✓	—	✓
Input/Output	✓	✓	✓	✓
Independent	—	—	✓	—

Asynchronous clock mode is only supported by legacy devices.

In single clock mode, a single clock can be used together with a clock enable to control all registers or selected registers of the memory blocks.

In read/write clock mode, a separate clock is available for each read port and write port. The read clock controls all the read port registers (data output, read address, and read-enable registers) and the write clock controls all the write port registers (data input, write address, write enable, and byte enable registers).

In input/output clock mode, a separate clock is available for each input port and output port. The input clock controls all input registers (data input, addresses, byte enables, read enables, and write-enables registers) to the memory and the output clock controls the output registers (data output).

In independent clock mode, a separate clock is available for each port (A and B). Clock A controls all registers on the port A side, while clock B controls all registers on the port B side.

☞ Independent clock enables are supported for both read/write clocks, both input/output clocks, or both port A/port B registers, depending on the clocking modes selected.

Address Clock Enable

The address clock enable is an active high asynchronous control signal used to hold the previous address value for as long as the signal is enabled. When the RAM function is configured in dual-port mode, each port has its own independent address clock enable.

Figure 5 and Figure 6 show the effect of address clock-enable signal during the read operation and the write operation, respectively.

Figure 5. Address Clock Enable During Read Operation

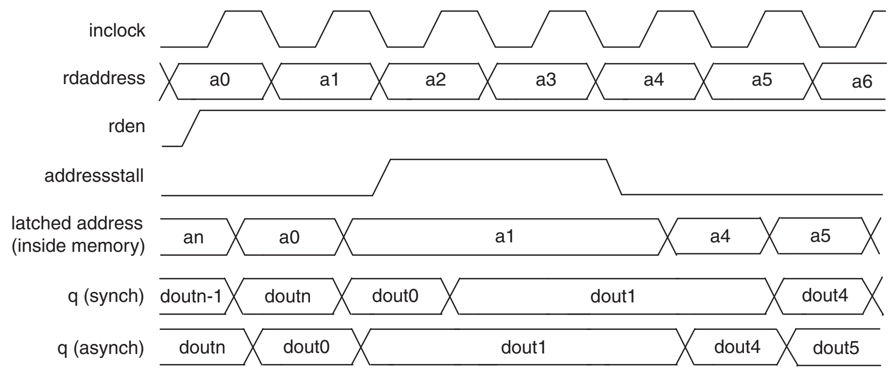
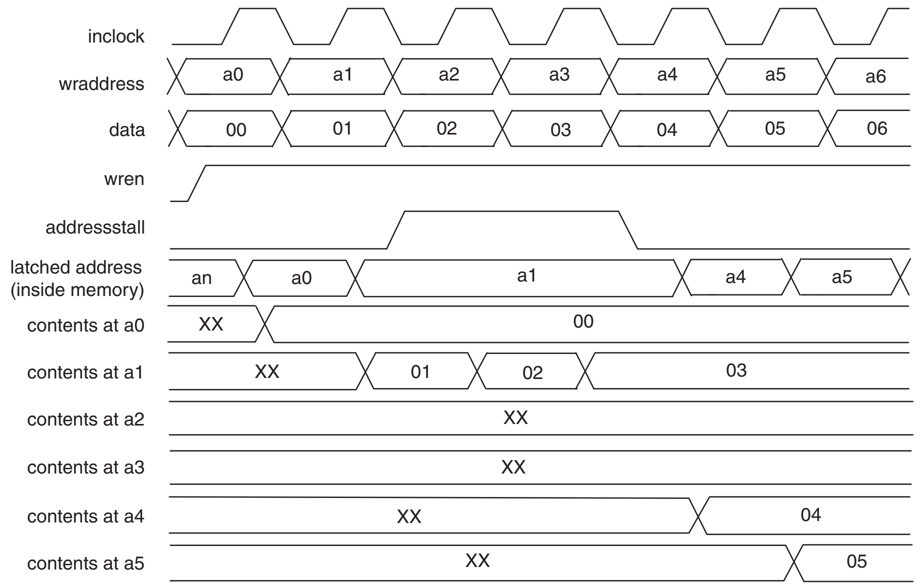


Figure 6. Address Clock Enable During Write Operation



☞ To configure the address clock enable feature, click **More Options** besides the **clock enable** options from the RAM MegaWizard plug-in. Turn on the option to create the `addressstall` port to enable the feature.

☞ The tri-port RAM does not support the address clock-enable feature.

Byte Enable

All TriMatrix memory blocks (except M512) support byte enables that mask the input data so that only specific bytes, nibbles, or bits of data are written. The unwritten bytes or bits retain the previous written value.

The write-enable signal and the byte-enable signal control the write operation of a RAM. The default value for byte-enable signals is high (enabled), in which case the write operation is controlled only by the write-enable signal.

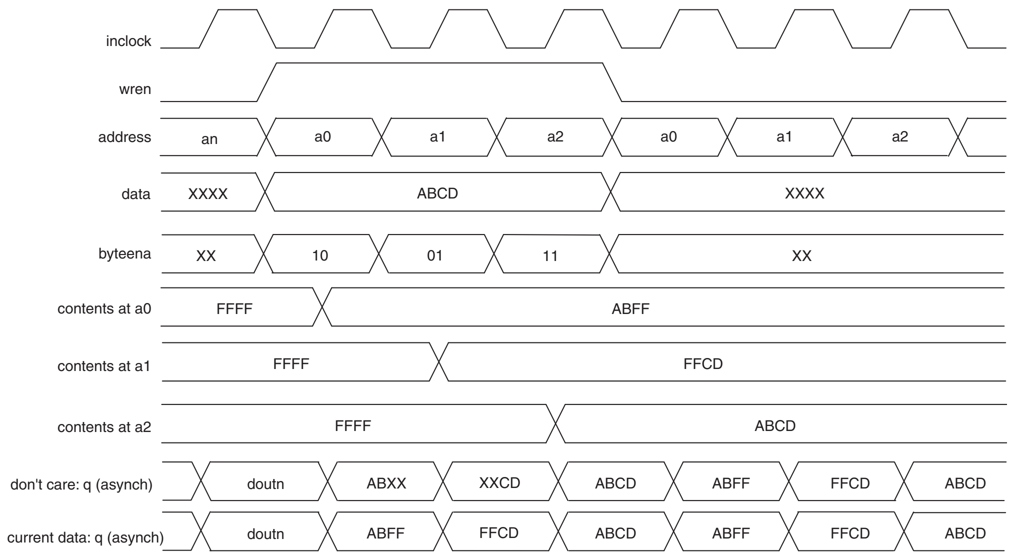
Byte enable operates in a one-hot fashion, with the least significant bit (LSB) of the byte-enable signal corresponds to the least significant byte of the data bus. For example, if you use a RAM block in x18 mode, with the byte-enable signal equals 01, `data [8..0]` is enabled and `data [17..9]` is disabled. Similarly, if the byte-enable signal equals 11, both data bytes are enabled.

You can specifically define and set the size of a byte for byte-enable signal. The valid values are 5, 8, 9, and 10, depending on the types of TriMatrix memory blocks. The values of 5 and 10 are only supported by MLAB.

To create a byte-enable port, the width of the `data` input port must be a multiple of the size of a byte for the byte-enable signal. For example, if you use an MLAB, the byte enable is only supported if your data bits are multiples of 5, 8, 9 or 10, that is 10, 15, 16, 18, 20, 24, 25, 27, 30, and so on. If the width of the `data` input port is 10, then you can only define the size of a byte as 5. In this case, you get a 2-bit byte-enable signal, each bit controlling 5 bits of data input written. If the width of the `data` input port is 20, then you can define the size of a byte as either 5 or 10. If you define 5 bits of input data as a byte, you get a 4-bit byte-enable signal, each bit controlling 5 bits of data input written. If you define 10 bits of input data as a byte, you get a 2-bit byte-enable signal, each bit controlling 10 bits of data input written.

Figure 7 shows the effect of the byte enable on the data written into the memory, and the data reading from the memory.

Figure 7. Byte Enable Functional Waveform



When a byte-enable bit is de-asserted (LOW) during a write cycle, the corresponding byte (masked byte) of the `q` output can appear as either a "Don't Care" value or the current data at that location. This selection can be controlled via the RAM MegaWizard plug-ins when the read-during-write for output behavior is set to "New Data".

The tri-port RAM does not support the byte-enable feature.

Asynchronous Clear

The asynchronous-clear signal does not affect the input registered ports for some devices. Table 8 shows the input and output registers that are affected by the asynchronous clear signal for different TriMatrix memory blocks in different Cyclone and Stratix series of devices.

Table 8. Register Clears for Different TriMatrix Memory Blocks in Different Devices

Devices	TriMatrix Memory Blocks	Register Clears
Stratix, Stratix GX, Cyclone *(1)*	M512	Input and output registers *(3)*
	M4K	
	M-RAM	Output registers only *(4)*
Stratix II, Stratix II GX, Cyclone II *(1)*	M512	
	M4K	
	M-RAM	
Stratix IV, Stratix III, Cyclone III *(2)*	MLAB	
	M9K	
	M144K	

Notes to Table 8:

(1) Cyclone and Cyclone II devices have M4K blocks only.

(2) Cyclone III devices have M9K blocks only.

(3) You can specifically select which input or output registers are affected by the asynchronous-clear signal from the RAM MegaWizard plug-ins.

(4) You can specifically select which output registers are affected by the asynchronous clear signal from the RAM MegaWizard plug-ins. You also can asynchronously clear the read address input for port B in simple dual-port mode, when your target is a Stratix IV, Stratix III, or Cyclone III device.

You can ensure which register ports take effect on the asynchronous-clear signal from the RAM MegaWizard plug-ins. Click on the **More Options** button next to the **asynchronous clear** option, and the asynchronous clear page appears. The page shows the ports that are disabled. These disabled ports are not affected by the asynchronous clear signal. In the same page, you can turn on the available ports that can be affected by the asynchronous clear signal.

Stratix IV, Stratix III, and Cyclone III TriMatrix memory blocks support asynchronous clears on the output latches (except MLAB block) and output registers. This allows you to clear the RAM outputs via the output latch asynchronous clear even if the `q` output port is not registered.

Read Enable

Support for the read enable depends on the target device, memory block type, and RAM mode you select. Table 9 shows the configurations of RAM for the Stratix and Cyclone series of devices that support the read enable feature.

Table 9. Read-Enable Supports for the Stratix and Cyclone Series of Devices

RAM Modes	Stratix IV, Stratix III, Cyclone III Devices		Stratix II, Stratix II GX, Stratix, Stratix GX, Cyclone II, Cyclone Devices	
	M9K, M144K	**MLAB**	**M512, M4K**	**M-RAM**
Single-port	✓	—	—	—
Simple dual-port	✓	—	✓	—
True dual-port	✓	—	—	—
Tri-port	✓	✓	✓	—

If you use the read-enable signal and perform a write operation (with the read enable deactivated), the `data` output port retains the previous values held during the most recent active read enable. If you activate the read enable during a write operation, or if you do not create a read-enable signal, the output port shows the new data being written, the old data at that address, or a "Don't Care" value when read-during-write occurs at the same address location. You can set the output behavior from the RAM MegaWizard Plug-In.

For more information about the read-during-write output behavior, refer to the "Read-During-Write Output Behavior" on page 1–16.

Read-During-Write Output Behavior

The read-during-write output behavior occurs when a read operation and a write operation target the same memory location at the same time. You can use the RAM MegaWizard plug-ins to configure the read-during-write behavior of your RAM. There are two types of read-during-write operations available: same-port and mixed-port.

The same-port read-during-write occurs when the input and output of the same port access the same address location with the same clock. The mixed-port read-during-write occurs when one port reads and another port writes to the same address location with the same clock. The output choices for the read-during-write behavior vary, depending on the types of read-during-write and types of TriMatrix memory block implemented.

Table 10 shows the available output choices for the same-port and mixed-port read-during-write for different TriMatrix memory blocks.

Table 10. Output Choices for the Same-Port and Mixed-Port Read-During-Write for Different TriMatrix Memory Blocks

TriMatrix Memory Blocks	Same-Port *(1)*	Mixed-Port *(2)*
M144K and M9K	New Data, Old Data, or Don't Care *(3)*	Old Data or Don't Care *(5)*
MLAB *(6)*	Don't Care	
M-RAM, M4K, and M512	New Data *(4)*	

Notes to Table 10:

(1) The same-port read-during-write only occurs in the single-port RAM, or the same port of a true dual-port RAM (two read and write ports) with the read and write operations synchronous under the same clock.

(2) The mixed-port read-during-write only occurs in the simple dual-port (one read port and one write port) or the true dual-port RAM with the read and write operations synchronous under the same clock.

(3) "Don't Care" is not available for selection in true dual-port mode.

(4) There is no option page available from the RAM MegaWizard plug-ins in this mode. By default, the new data flows through to the output.

(5) The M-RAM does not support "Old Data" behavior.

(6) For MLAB, New Data flow through to the output port by default if the read address and output data are controlled by the write clock.

The read-during-write old data mode is not supported when the Error Correction Code (ECC) is engaged.

A read-during-write output choice is not configurable for the tri-port RAM.

In new data mode (or flow-through), the new data is available on the rising edge of the same clock cycle in which it was written. In old data mode, the RAM outputs reflect the old data at the address before the write operation proceeds. In don't care mode, the RAM outputs reflect "Don't Care".

If you are not concerned about the output when read-during-write occurs, you should select the don't care mode. Using the don't care mode increases the flexibility in the type of memory block used, provided you do not assign block type when instantiating the memory block. You also get a potential performance gain by selecting the don't care mode.

Power-Up Conditions and Memory Initialization

Different TriMatrix memory blocks have different power-up output behavior and also depend on whether the output port is registered.

Table 11 shows the power-up conditions in different TriMatrix memory blocks.

Table 11. Power-Up Conditions for Different TriMatrix Memory Blocks

TriMatrix Memory Blocks	Power-Up Conditions
M512	Outputs cleared
M4K	Outputs cleared
M-RAM	Outputs cleared if registered, otherwise unknown
MLAB	Outputs cleared if registered, otherwise reads memory contents
M9K	Outputs cleared
M144K	Outputs cleared

The outputs of M512, M4K, M9K, and M144K blocks always power-up to zero, whether the output registers are used or bypassed. Even if a memory initialization file is used to pre-load the contents of the RAM blocks, the outputs still power-up to cleared. If the contents are not initialized and after power-up an address is read before being written, the output from the read operation is zero.

MLAB and M-RAM blocks power-up to zero only if output registers are used. If output registers are not used, MLAB blocks power-up to reading the memory contents while M-RAM blocks power-up to an unknown state.

When the memory block type is set to **Auto** in the RAM MegaWizard plug-ins, the compiler is free to choose any RAM block type, where the power-up value depends on the chosen memory block type.

To identify the type of memory block the software selected to implement your RAM, refer to the fitter report after compilation. For the chosen memory block type, refer to Table 11 for the power-up conditions.

All memory blocks (except M-RAM) support memory initialization via the Memory Initialization File (**.mif**) or Hexadecimal (Intel-format) file (**.hex**). You can include the files from the RAM MegaWizard plug-ins when configuring and building your RAM.

Error Correction Code (ECC)

Stratix III and Stratix IV M144K blocks have built-in support for the ECC up to a data width of x64 for the simple dual-port mode. The ECC allows you to detect and correct data errors in the memory array. The ECC features single-error-correction double-error detection (SECDED) implementation where it can detect and fix a single-bit error or detect two-bit errors (without fixing).

The M144K ECC status is communicated via a three-bit status flag `eccstatus[2..0]`. The status flag can be either registered or unregistered. When registered, it uses the same clock and asynchronous clear signals as the output registers. When not registered, it cannot be asynchronously cleared.

Table 12 shows the truth table for the ECC status flags.

Table 12. Truth Table for ECC Status Flags

Status	Eccstatus[2..0]
No error	000
Single error and fixed	011
Double error and no fix	101
Illegal	001
	010
	100
	11X

☞ The ECC features cannot be used when the byte-enable feature is engaged. Also, read-during-write old data mode is not supported when the ECC is engaged.

You can also use Altera's soft IP megafunctions, ALTECC_ENCODER and ALTECC_DECODER, to implement the ECC external to your memory blocks. For more information, refer to the *Error Correction Code Megafunction User Guide*.

Design Example with External ECC Implementation

The ECC features are only supported internally in simple dual-port mode by Stratix III and Stratix IV devices when the M144K is implemented. Therefore, this design example provides you with an idea of how ECC features can be implemented in other RAM modes, regardless of the type of device memory block used. It also demonstrates features of the same-port and the mixed-port read-during-write behaviors.

This design example takes a true dual-port RAM as an example and illustrates how the ECC feature can be implemented external to the RAM. The ALTECC_ENCODER and ALTECC_DECODER megafunctions are required. The ALTECC_ENCODER megafunction encodes the data input before writing the data into the RAM, while the ALTECC_DECODER decodes the data output before transfering the data out to other parts of the logic.

In this design example, the raw data width is 8 bits and is encoded by the ALTECC_ENCODER megafunction block to produce a 13-bit width data written into the true dual-port RAM when write-enable signal is asserted. Since the RAM mode has two dedicated write ports, another encoder is implemented for the other input port of the RAM.

Two ALTECC_DECODER megafunction blocks are also implemented at each of the `data` output ports of the RAM. When the read-enable signal is asserted, the encoded data is read out from the respective RAM address and decoded by the ALTECC_DECODER megafunction blocks, respectively. The decoder shows the status of the data as no error detected, single-bit error detected and corrected, or fatal error (more than 1-bit error).

For more information about ECC features and the megafunctions settings, refer to the *Error Correction Code (ALTECC_ENCODER and ALTECC_DECODER) Megafunction User Guide*.

This example also includes a "corrupt zero bit" control signal at port A of the RAM. When the signal is asserted, it changes the state of the zero-bit (LSB) encoded data before it is written into the RAM. This signal is used to corrupt the zero-bit data storing through port A, and examines the effect of the ECC features.

This design example is intended to illustrate the concept of how ECC features can be implemented with the RAM for the case where the ECC is not supported internally by the RAM. It may not represent the optimized design or implementation.

Configuration Settings

These design examples contain the following files:

- **ecc_encoder.v**
- **ecc_decoder.v**
- **true_dp_ram.v**
- **top_dpram.v**
- **true_dp_ram.vt**
- **true_dp.do**

The **ecc_encoder.v** is a design variation file for the ALTECC_ENCODER megafunction that is pre-configured with the settings shown in Table 13.

Table 13. ALTECC_ENCODER Megafunction Settings

MegaWizard Page	Available Options	Configured Settings
3	Currently selected device family:	Stratix III
	How do you want to configure this module?	Configure this module as an ECC encoder
	How wide should the data be?	8 bits
	Do you want to pipeline the functions?	Yes, I want an output latency of 1 clock cycle
	Create an 'aclr' asynchronous clear port	Not selected
	Create a 'clocken' clock enable clock	Not selected

The **ecc_decoder.v** is a design variation file for the ALTECC_DECODER megafunction that is pre-configured with the settings shown in Table 14.

Table 14. ALTECC_DECODER Megafunction Settings

MegaWizard Page	Available Options	Configured Settings
3	Currently selected device family:	Stratix III
	How do you want to configure this module?	Configure this module as an ECC decoder
	How wide should the data be?	13 bits
	Do you want to pipeline the functions?	Yes, I want an output latency of 1 clock cycle
	Create an 'aclr' asynchronous clear port	Not selected
	Create a 'clocken' clock enable clock	Not selected

For more information about the options available from the ALTECC MegaWizard Plug-In Manager, refer to *Error Correction Code (ALTECC_ENCODER and ALTECC_DECODER) Megafunction User Guide*.

The **true_dp_ram.v** is a design variation file for the true dual-port RAM (instantiated through the ALTSYNCRAM megafunction) that is pre-configured with the settings shown in Table 15.

Table 15. RAM:2-PORT MegaWizard Plug-In Settings

MegaWizard Page	Available Options	Configured Settings
3	Currently selected device family:	Stratix III
	How will you be using the dual port ram?	With two read/write ports
	How do you want to specify the memory size?	As a number of words
4	How many 8-bit words of memory?	16
	Use different data widths on different ports	Not selected
	How wide should the 'q_a' output bus be?	13
	What should the memory block type be?	M9K
	Set the maximum block depth to	Auto
5	Which clocking method do you want to use?	Single clock
	Create 'rden_a' and 'rden_b' read enable signals	Not selected
	Byte Enable Ports	Not selected
6	Which ports should be registered?	All `write` input ports and `read` output ports
	Create one clock enable signal for each signal	Not selected
	Create an 'aclr' asynchronous clear for the registered ports	Not selected
7	Mixed Port Read-During_Write for Single Input Clock RAM	Old memory contents appear
8	Port A Read-During-Write Option	New Data
	Port B Read-During-Write Option	Old Data

The **top_dpram.v** is a design variation file that contains the top level that instantiates two encoders, a true dual-port RAM, and two decoders. To simulate the design, a testbench, **true_dp_ram.vt,** is created for you to run in the ModelSim®-Altera software.

Functional Simulation in the ModelSim-Altera Software

Simulate the design in the ModelSim-Altera software to generate a waveform that displays the functional behavior of the design example. You should be familiar with the ModelSim-Altera software before trying the design example. If you are unfamiliar with the ModelSim-Altera software, refer to the **Support** page for software products on the Altera website. On the **Support** page, there are links to such topics as installation, usage, and troubleshooting.

Set up and simulate the design in the ModelSim-Altera software by performing the following steps:

1. Unzip the **DesignExample_ram.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files.
5. Click **OK**
6. On the Tools menu, point to **TCL** and click **Execute Macro**. The **Execute Do File** dialog box appears.
7. Select the **true_dp.do** file and click **Open**. The **true_dp.do** file is a script file that automates all the necessary settings, compiles and simulates the design files, and displays the simulation waveform.
8. Verify the result shown in the **Waveform Viewer** window.

You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in **true_dp.do** accordingly.

Understanding the Simulation Results

The top level contains the input and output ports shown in Table 16.

Table 16. Top Level Input and Output Ports Representations

Ports Name	Ports Type	Descriptions
`clock`	Input	System Clock for the encoders, RAM, and decoders.
`corrupt_dataa_bit0`	Input	Registered active high control signal that 'twist' the zero bit (LSB) of input encoded data at port A before writing into the RAM *(1)*

`address_a`
`data_a`
`wren_a`
`rden_a`
	Input	Address input, data input, write enable, and read enable to port A of the RAM. *(1)*

`address_b`
`data_b`
`wren_b`
`rden_b`
	Input	Address input, data input, write enable, and read enable to port B of the RAM. *(1)*

`rdata1`
`err_corrected1`
`err_detected1`
`err_fatal1`
	Output	Output data read from port A of the RAM, and the ECC-status signals reflecting the data read. *(2)*

`rdata2`
`err_corrected2`
`err_detected2`
`err_fatal2`
	Output	Output data read from port B of the RAM, and the ECC-status signals reflecting the data read. *(2)*

Notes to Table 16:

(1) For input ports, only data signal goes through the encoder; others bypass the encoder and directly go to the RAM block. Since the encoder uses one pipeline, those signal that bypass the encoder require additional pipeline before going to the RAM. This has been implemented in the top level.

(2) The encoder and decoder each use one pipeline while the RAM uses two pipelines, making the total pipeline equal to four. Therefore, read data is only shown at output ports four clock cycles after the read enable is initiated.

Figure 8 shows the expected simulation waveform results in the ModelSim-Altera software.

Figure 8. Simulation Results

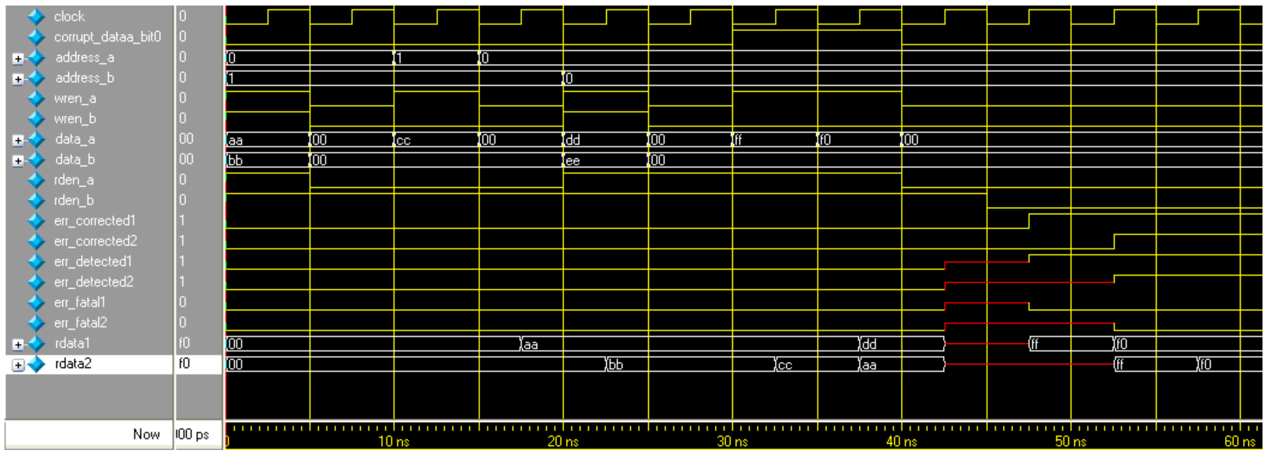
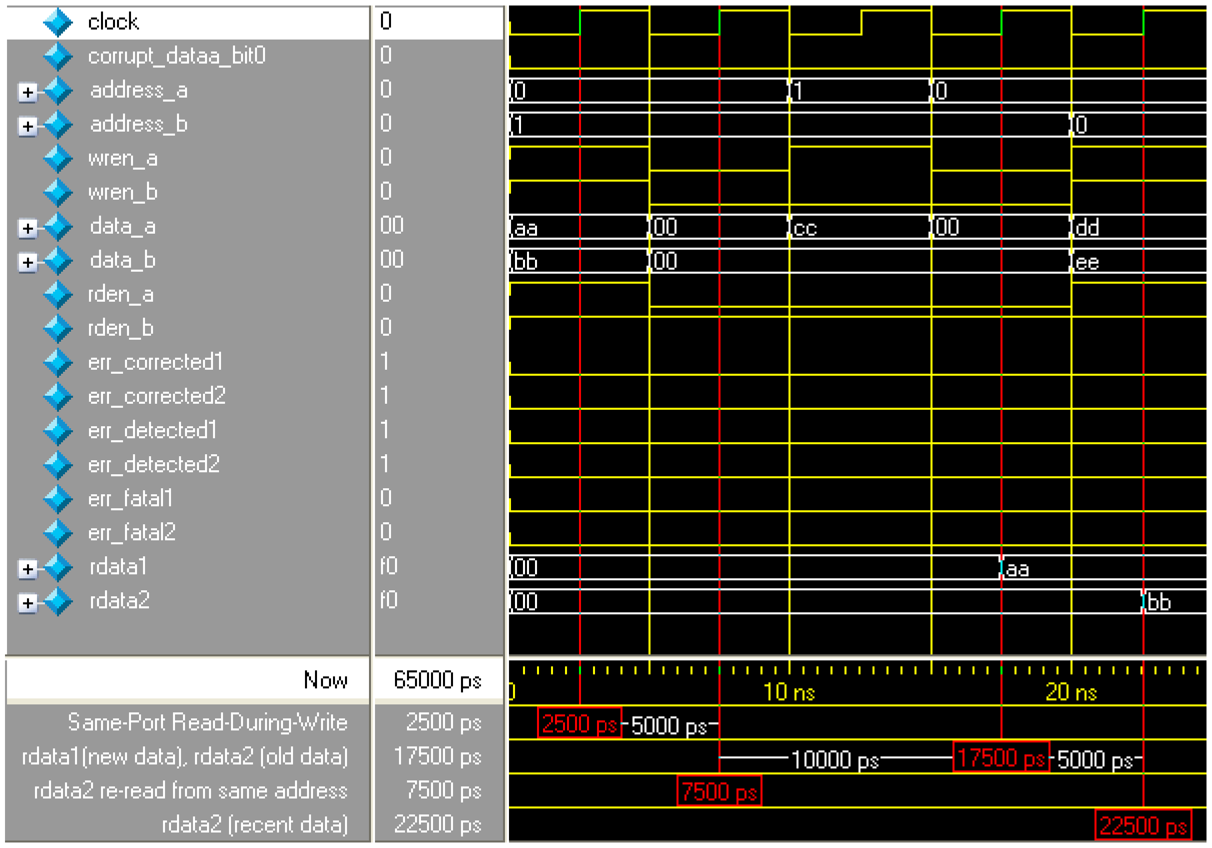


Figure 9 shows the magnified portion when same-port read-during-write occurs for each port A and port B of the RAM.

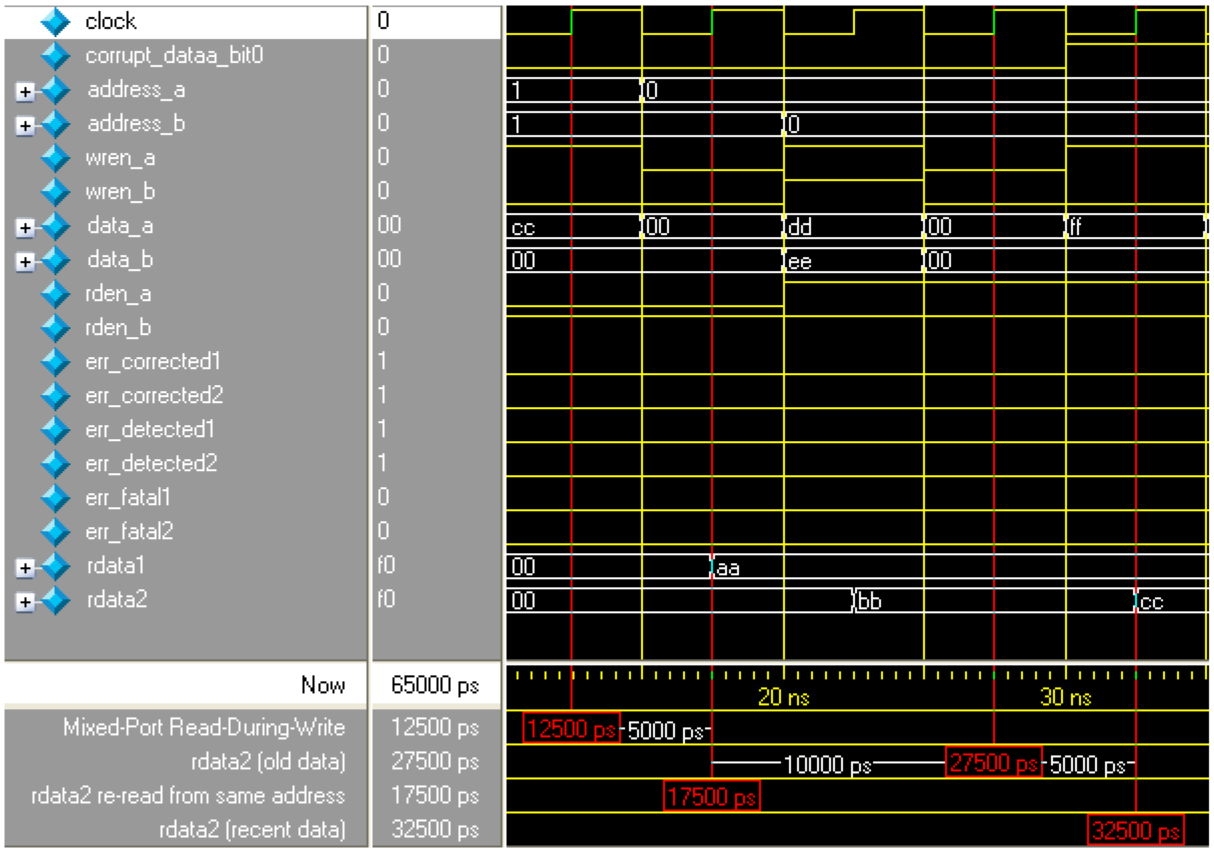
Figure 9. Same-Port Read-During-Write



At 2,500 ps, same-port read-during-write occurs for each port A and port B. Since the true dual-port RAM configured to port A is reading the new data and port B is reading the old data when the same-port read-during-write occurs, the `rdata1` port shows the new data 'aa' and the `rdata2` port shows the old data '00' after four clock cycles at 17,500 ps. When re-read from the same address at the next rising clock edge at 7,500 ps, the `rdata2` port shows the recent data 'bb' at 22,500 ps.

Figure 10 shows the magnified portion when mixed-port read-during-write occurs.

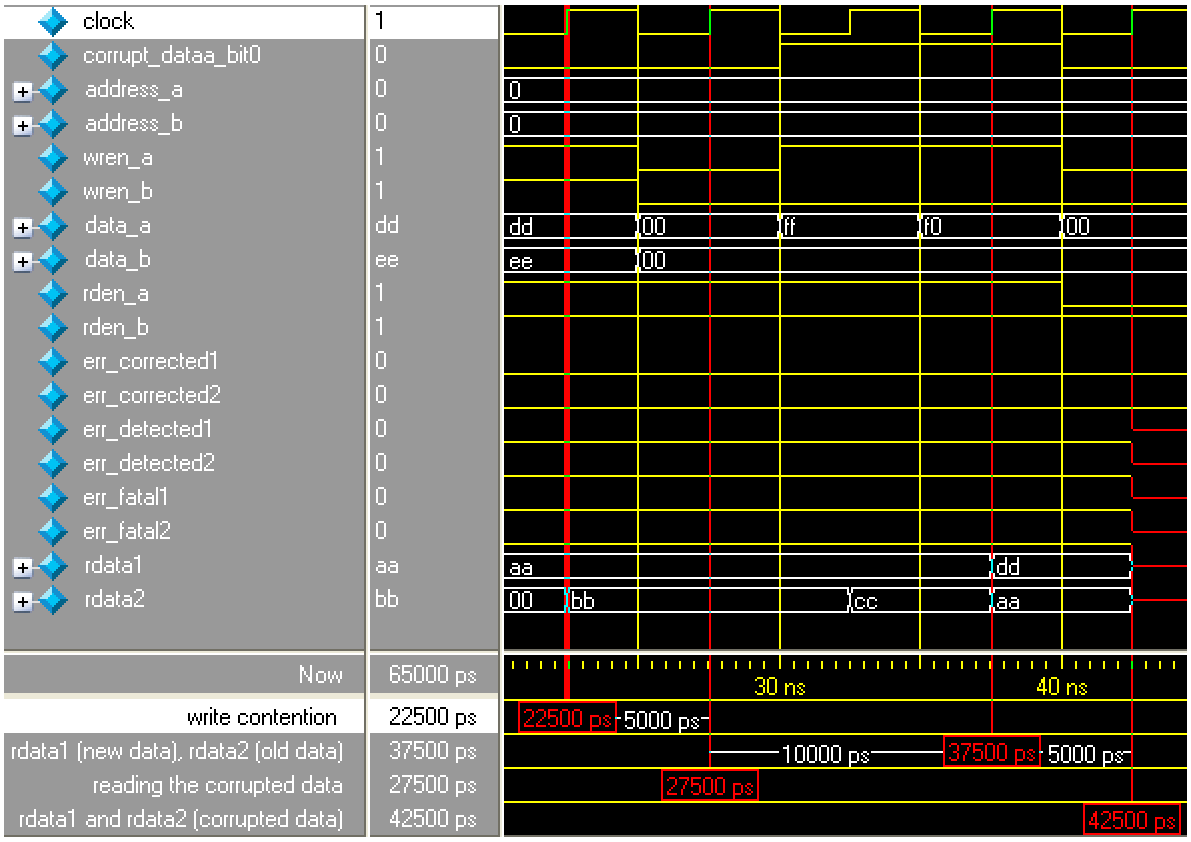
Figure 10. Mixed-Port Read-During-Write



At 12,500 ps, mixed-port read-during-write occurs when data 'cc' is both written to port A, and is reading from port B, simultaneously targeting the same address '1'. Since the true dual-port RAM is configured to mixed-port read-during-write is showing the old data, the `rdata2` port shows the old data 'bb' after four clock cycles at 27,500 ps. When re-read from the same address at the next rising clock edge at 17,500ps, the `rdata2` port shows the recent data 'cc' at 32,500 ps.

Figure 11 shows the magnified portion write contention occurs.

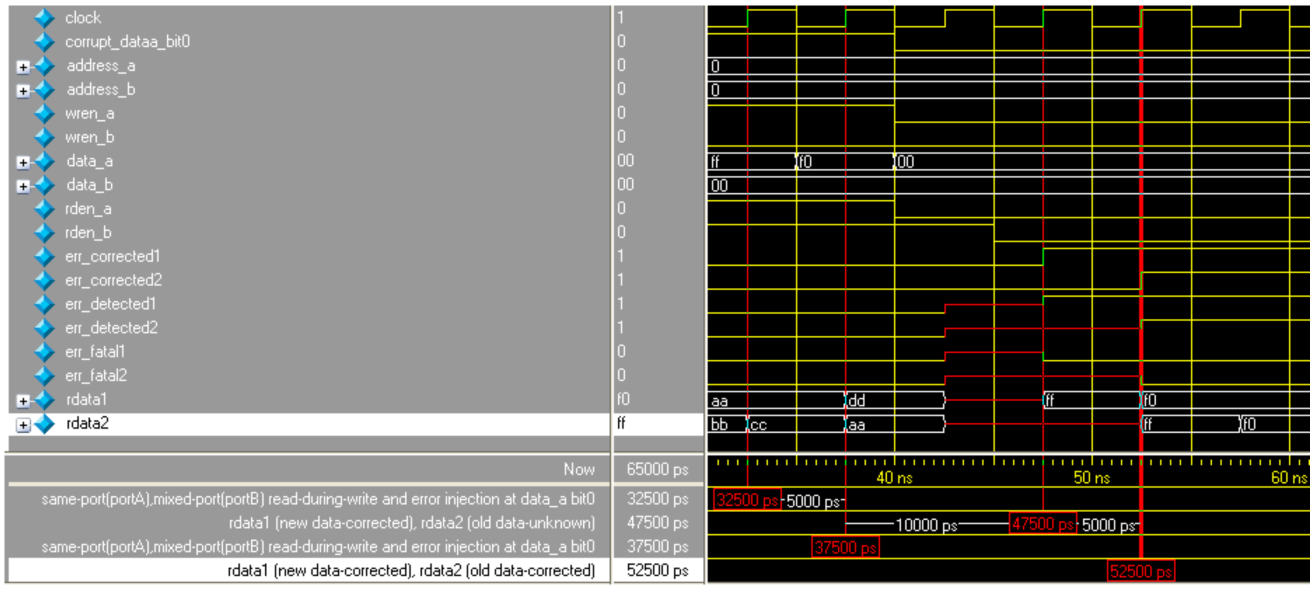
Figure 11. Write Contention



At 22,500 ps, write contention occurs when data 'dd' and 'ee' are written to address '0' simultaneously. Also, same-port read-during-write occurs for port A and port B. The setting for port A and port B for same-port read-during-write takes effect when the `rdata1` port shows the new data 'dd' and the `rdata2` port shows the old data 'aa' after four clock cycles at 37,500 ps. When re-read from the same address at the next rising clock edge at 27,500 ps, `rdata1` and `rdata2` ports show unknown values at 42,500 ps. Also, the unknown data input to the decoder results in an unknown ECC status.

Figure 12 shows the magnified portion of the effect when an error is injected to twist the LSB of the encoded data at port A by asserting `corrupt_dataa_bit0`.

Figure 12. Error Injection– Asserting `corrupt_dataa_bit0`


clock 1
corrupt_dataa_bit0 0
address_a 0
address_b 0
wren_a 0
wren_b 0
data_a 00
data_b 00
rden_a 0
rden_b 0
err_corrected1 1
err_corrected2 1
err_detected1 1
err_detected2 1
err_fatal1 0
err_fatal2 0
rdata1 f0
rdata2 ff
Now 65000 ps
same-port(portA),mixed-port(portB) read-during-write and error injection at data_a bit0 32500 ps
rdata1 (new data-corrected), rdata2 (old data-unknown) 47500 ps
same-port(portA),mixed-port(portB) read-during-write and error injection at data_a bit0 37500 ps
rdata1 (new data-corrected), rdata2 (old data-corrected) 52500 ps
0
0
ff
f0
00
00
aa
dd
ff
f0
bb
cc
aa
ff
f0
40 ns
50 ns
60 ns
32500 ps
5000 ps
10000 ps
47500 ps
5000 ps
37500 ps
52500 ps

At 32,500 ps, same-port read-during-write occurs at port A while mixed-port read-during-write occurs at port B. Also, `corrupt_dataa_bit0` is asserted to corrupt the LSB of encoded data at port A; therefore, the storing data has the LSB corrupted. That is, the intended data ‘ff’ is corrupted and become ‘fe’ and stored at address ‘0’. After four clock cycles at 47,500ps, the `rdata1` port shows the new data ‘ff’ that has been corrected by the decoder, and the ECC status signals, `err_corrected1` and `err_detected1`, are asserted. For `rdata2` port, old data (which is unknown) is shown and the ECC-status signal remains unknown.

☞ The decoders only correct the single-bit error of the data shown at `rdata1` and `rdata2` ports. The actual data stored at the address ‘0’ in the RAM remains corrupted, until new data is written.

At 37,500 ps, the same condition happens to port A and port B. The difference is port B reads the corrupted old data (‘fe’) from address ‘0’. After four clock cycles at 52,500 ps, the `rdata2` port shows the old data 'ff' that has been corrected by the decoder and the ECC status signals, `err_corrected2` and `err_detected2`, are asserted to show the data has been corrected.

Ports and Parameters

The ports and parameters details are only relevant if you bypass the MegaWizard® Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in your design. The details of the parameters are hidden from MegaWizard Plug-In Manager interface. Altera recommends using the MegaWizard plug-ins to configure and build your RAM functions. This ensures the combination options you set for the RAM are valid.

For most current information on the ports and parameters for the memory megafunctions (ALTSYNCRAM, ALTDPRAM, ALT3PRAM, and LPM_RAM_DQ), refer to the latest version of the Quartus® II Help .

Revision History

The following table shows the revision history for this user guide.

Date	Version	Changes Made
December 2008	3.0	Complete re-write of the Guide
March 2007	2.0	Complete re-write of the Guide
September 2004	1.0	Initial Release

Referenced Documents

This user guide references the following documents:

- *TriMatrix Embedded Memory Blocks* chapter in your target device handbook.
- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook.*
- *Error Correction Code (ALTECC_ENCODER and ALTECC_DECODER) Megafunction User Guide*.

How to Contact Altera

For the most up-to-date information about Altera® products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, file names, file name extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <*file name*>, <*project name*>**.pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input`. Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
⚠ CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
⚠ WARNING	A warning calls attention to a condition or possible situation that can cause injury to the user.
↵	The angled arrow indicates you should press the Enter key.
👣	The feet direct you to more information on a particular topic.