Stratix III RAM WYSIWYG User Guide

Version 1.0

March 9, 2009

by

Altera Corporation

Altera Corporation Proprietary

Table of Contents:

1. Overview

This document describes the WYSIWYG primitive for Stratix III memory blocks. Stratix III memory blocks are very similar to Stratix II memory blocks in terms of features. Stratix III has two different kinds of RAM blocks – M9K and M144K. Stratix III RAM blocks support all the Stratix II memory features. In addition, there are a few new features in Stratix III memory block that are described in later sections.

The small RAM block in Stratix II has been removed in Stratix III and replaced by the LUTRAM feature. For details on the LUTRAM feature, please refer to the LUTRAM specific documentation.

M9K is now a 9K-bits RAM block instead of 4K-bits in Stratix II. And M144K contains 144K-bits instead of 576K-bits in Stratix II. More importantly, Stratix III M144K now supports all features of M9K, whereas the MRAM didn't support all features of the M4K in Stratix II.

1.1 Stratix III RAM block

The following picture shows the abstract functional I/O interface for Stratix III memory. The ports in blue color are new in Stratix III compared to Stratix II.

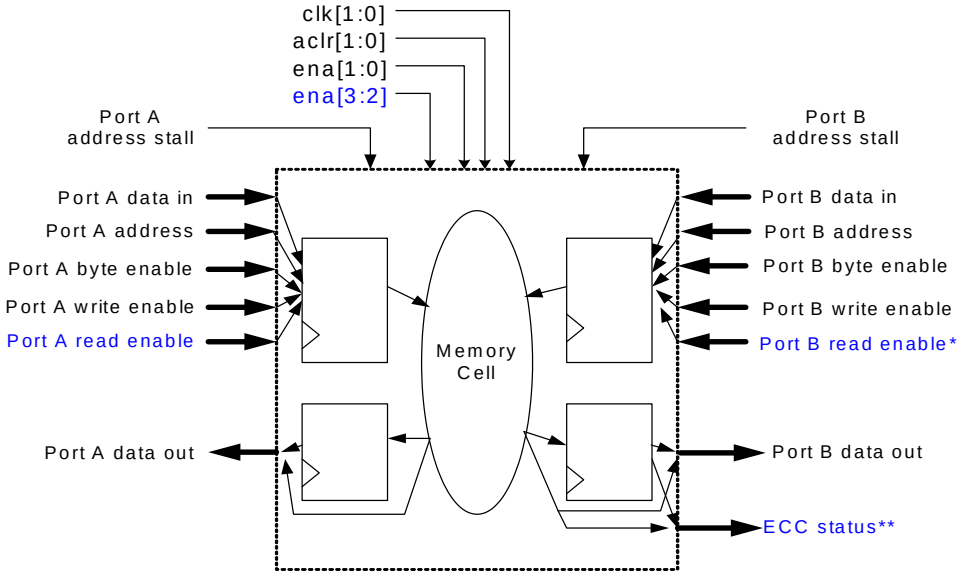


Figure 1: Stratix III RAM

* In Stratix II, port B read enable is shared with port B write-enable
** ECC feature is only available in M144K simple dual-port x64 mode

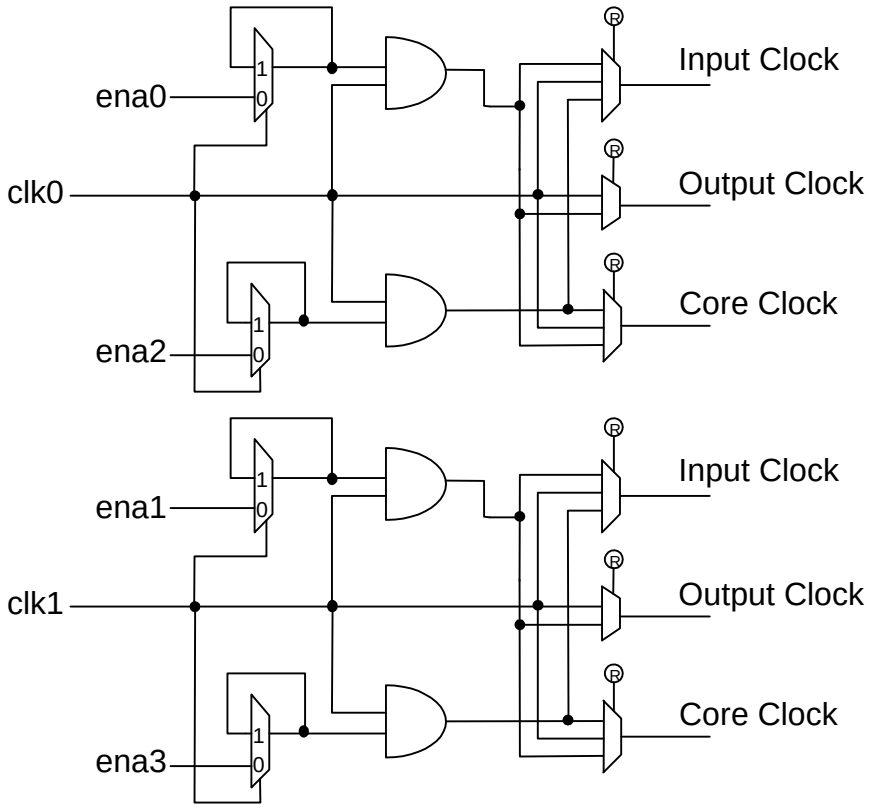
1.2 Changes from Stratix II

Other than the size changes mentioned above, another important change is that the M144K now supports all functional features of M9K plus the new ECC feature. This enables the shifting of resources from M144K blocks to M9K and vice versa.

In addition, the following feature enhancements were made to the Stratix III RAM block architecture.

1.2.1 Additional Clock Enables

An additional clock enable has been added to each clock to allow for a more flexible control of the RAM clocking. In Stratix III, each clock now has two clock-enable controls associated with it as opposed to one clock-enable in Stratix II. Therefore, the input registers, output registers, and core memory cells can have their own choice of using (i) one clock-enable, or (ii) the clock without a gating clock-enable. Note that the output registers only have the choice of using (i) clock-enable ena[1:0], or (ii) no clock-enable. They don't have access to the additional clock-enables ena[3:2]. This new feature helps to reduce the power consumption of the memory block and provides more opportunities to absorb the output registers into the RAM block.



1.2.2 Additional Read-During-Write Modes

Stratix II only has one same-port read-during-write mode which is the new data mode (a.k.a. flow-through mode). It means that the data output of the read port can show the new written data during the write operation at the same clock cycle. In Stratix III, this feature has been enhanced to provide the old-data mode as well. Therefore, the RAM outputs can show the old data at the address before the write proceeds. This feature provides a better match to the behavior of the non-blocking assignment in HDL.

Mode	Data	Address	Output	Mem Content
New-data(flow-through)	DA	R	DA	Mem[R] = DA
Old-data	DA	R	Previous Mem[R]	Mem[R] = DA

The mixed-port read-during-write mode remains the same as in Stratix II. The read-port can show the old-data when the write port is accessing the same location using the same clock.

1.2.3 Independent Write Enable (WE) and Read Enable (RE) Controls

Stratix II only has one write enable port to control the write/read mode which has the side effect that read-during-write is always enabled. When the WE port is high, the memory block is in write mode. When the WE port is low, it is in read mode. The memory block in Stratix III has separate read-enable and write-enable controls. This is useful to reduce power if the user does not care about the data output during a write operation.

1.2.4 Read During Byte-enable Mask Write

The status of data outputs that have been masked out by a byte enable during a write operation is unknown in Stratix II. Indeed, they remain at the same state as they were during the previous operation. Therefore, they are modeled as unknown (X) for simplicity.

Stratix III has the additional option of reading out the memory content at the current write address location if the read-during-write mode is set to “New Data”. In this case, the resulting output is equivalent to what is actually written to the memory cell.

Assume Mem[R] = YZ before the write, and same-port read-during-write is set to “new data”

Mode	Data	BE	Address	Output	Mem Content
Read masked byte	AB	10	R	AZ	Mem[R] = AZ
No read masked byte	AB	10	R	AX	Mem[R] = AZ

1.2.5 ROM Mode for the M144K Block

The MRAM in Stratix II does not support initialized content. The M144K block in Stratix III has been enhanced to allow user specified initial content.

1.2.6 Error-Correction Code(ECC) Feature for the M144K Block

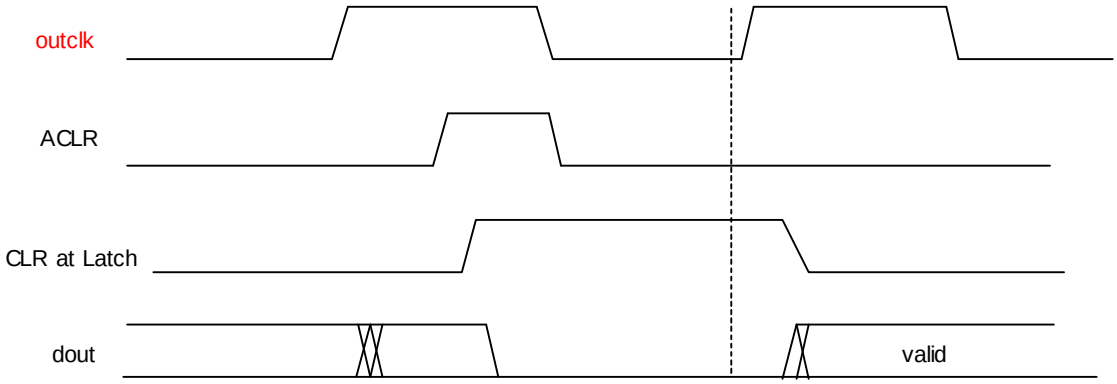
The M144K block in Stratix III supports the Error Correction Code (ECC) feature in simple dual-port mode with 64-bit data. It can detect and fix a single bit error event in the 64 bits data, or detect two errors in the 64 bits data without fixing the errors. It cannot detect three or more errors.

An important note is that the byte-enable feature cannot be used if ECC is engaged. In addition, the ECC status signal can be registered or not. Its operation will follow the port B output register mode. When port B data output is registered, the ECC output will be registered too with the same clock/clear control as the port B data output registers.

1.2.7 Asynchronous Clears on Address Registers and Output Latch

The Stratix II memory blocks provide asynchronous clear signals only on output registers. The Stratix III memory blocks provide asynchronous clear controls (ACLR) to address registers and to the output latch as well. This implies that the output latch can have an asynchronous reset even if the output registers are not used. Note however that the output latch is based on the core clock (a variation of input clock) and not the output clock. Therefore, it will resume to pass data through on the next input clock edge once ACLR is released.

Note that performing an asynchronous clear on the address registers can cause data corruption of the memory array. This feature has been mainly added to the memory block to improve the FIFO application since the FIFO application does not care about the data after asynchronous reset.



Output Latch Asynchronous Clear Waveform

RAM Register Group	Clear Source	Clear Disable
Port A Address	A_CLR	Yes
Port B Address	A_CLR/B_CLR	Yes
Output Latch	A_CLR/B_CLR	Yes
Output Registers	A_CLR/B_CLR	Yes

Asynchronous Clear Option for RAM Registers

2. RAM Operation Modes

Operation Mode	M9K (M9K)	M144K (M-RAM)
Single-port	8Kx1 1Kx9 4Kx2 512x18 2Kx4 256x36[1]	16Kx9 2Kx72[1] 8Kx18 4Kx36
Simple dual-port (dual_port)	WxM/RxN or WxY/RxZ M,N is (1,2,4,8,16,32) Y,Z is (9, 18, 36)	WxM / RxN M is (9,18,36, 72) N is (9,18,36, 72)
True dual-port (bidir_dual_port)	AxM/BxN or AxY/BxZ M,N is (1,2,4,8,16)[2] Y,Z is (9, 18)[2]	AxM / BxN M is (9,18,36) N is (9,18,36)

ROM [3]	Same as single-port[4]	Same as single-port[4]

Table - Stratix III RAM Configurations

1. The widest single-port mode is actually supported by emulation through true dual-port mode. Quartus II will pack two atoms(each with half data width) in true dual-port mode to support the widest mode.
2. The M9K memory block is now fully symmetric in terms of mixed-width configurations in true dual-port mode unlike the MEAB in Stratix II.
3. Inputs to Stratix III RAM are always registered. The ROM mode really means a pipelined ROM.
4. The widest ROM mode will be implemented through simple dual-port mode as opposed to a packed mode implementation for single-port mode. This is mainly for power savings considerations since ROM only needs one read-port and we can save power by disabling the activities in the write-port.

3. RAM Primitive

Just like Stratix II, we only have 1 WYSIWYG primitive Stratix III_ram_block to model the two RAM blocks and modes. New ports and parameters are highlighted in blue color. Obsolete ports and parameters from Stratix II are crossed out.

```
stratixiii_ram_block <block_name>
(
      // Port A inputs
      .portadatain[71..0](<port A write data source bus>),
      .portaaddr[13..0](<port A addresses bus>),
      .portawe(<port A write-enable source>),
      .portare(<port A read-enable source>),
      .portabyteenamasks[7..0](<port A byte-enable mask source
bus>),
      .portaaddrstall(<port A address stall source>),

      // Port B inputs
      .portbdatain[35..0](<port B write data source bus>),
      .portbaddr[13..0](<port B addresses bus>),
      .portbwe(<port B write-enable source>),
      .portbre(<port B read-enable source>),
      .portbbyteenamasks[3..0](<port B byte-enable mask source
bus>),
      .portbaddrstall(<port B address stall source>),

      // Control signals
      .clk0(<clock source 0>),
      .clk1(<clock source 1>),
      .ena0(<clock enable for clock 0>),
      .ena1(<clock enable for clock 1>),
      .ena2(<additional clock enable for clock 0>),
      .ena3(<additional clock enable for clock 1>),
      .clr0(<clear source 0>),
```

```
        .clr1(<clear source 1>),

        // Port A outputs
        .portadataout[71..0](<port A read data output bus>),

        // Port B outputs
        .portbdataout[71..0](<port B read data output bus>),

        .eccstatus[2..0](<Error correction/detection status>)
    );
    defparam <block_name>.operation_mode = <operation mode>;
    defparam <block_name>.enable_ecc = <enable the error
          correction/detection feature>;
    defparam <block_name>.mixed_port_feed_through_mode = <mixed port
          feed through mode>;
    defparam <block_name>.ram_block_type = <ram block type>;
    defparam <block_name>.logical_ram_name = <logical RAM's name>;
    defparam <block_name>.init_file = <name of the initialization
          file>;
    defparam <block_name>.init_file_layout = <layout of the
          initialization file>;
    defparam <block_name>.data_interleave_width_in_bits = <data
          interleave width in bits>;
    defparam <block_name>.data_interleave_offset_in_bits = <data
          interleave offset in bits>;

    defparam <block_name>.port_a_logical_ram_depth = <port A depth of
          the logical RAM >;
    defparam <block_name>.port_a_logical_ram_width = <port A width of
          the logical RAM >;
    defparam <block_name>.port_a_data_out_clock = <port A data out
          clock>;
    defparam <block_name>.port_a_data_out_clear = <port A data out
          clear>;
    defparam <block_name>.port_a_address_clear = <port A address
          clear>;
    defparam <block_name>.port_a_first_address = <port A starting
          address for this block>;
    defparam <block_name>.port_a_last_address = <port A ending
          address for this block>;
    defparam <block_name>.port_a_first_bit_number = <port A first
          logical bit position of this block>;
    defparam <block_name>.port_a_data_width = <width of the port A
          data bus of this block>;
    defparam <block_name>.port_a_address_width = <width of the port A
          address bus of this block>;
    defparam <block_name>.port_a_byte_enable_mask_width = <width of
          the port A byte-enable mask bus of this block>;
    defparam <block_name>.port_a_byte_size = <port A byte size>;
    defparam <block_name>.port_a_read_during_write_mode = <port A
          read-during-write mode>;

    defparam <block_name>.port_b_logical_ram_depth = <port B depth of
          the logical RAM >;
```

```
defparam <block_name>.port_b_logical_ram_width = <port B width of
      the logical RAM >;
defparam <block_name>.port_b_data_in_clock = <port B data in
      clock>;
defparam <block_name>.port_b_address_clock = <port B address
      clock>;
defparam <block_name>.port_b_address_clear = <port B address
      clear>;
defparam <block_name>.port_b_write_enable_clock = <port B write-
      enable clock>;
defparam <block_name>.port_b_read_enable_clock = <port B read-
      enable clock>;
defparam <block_name>.port_b_byte_enable_clock = <port B byte-
      enable clock>;
defparam <block_name>.port_b_data_out_clock = <port B data out
      clock>;
defparam <block_name>.port_b_data_out_clear = <port B data out
      clear>;
defparam <block_name>.port_b_first_address = <port B starting
      address for this block>;
defparam <block_name>.port_b_last_address = <port B ending
      address for this block>;
defparam <block_name>.port_b_first_bit_number = <port B first
      logical bit position of this block>;
defparam <block_name>.port_b_data_width = <width of the port B
      data bus of this block>;
defparam <block_name>.port_b_address_width = <width of the port B
      address bus of this block>;
defparam <block_name>.port_b_byte_enable_mask_width = <width of
      the port B byte-enable mask bus of this block>;
defparam <block_name>.port_b_byte_size = <port B byte size>;
defparam <block_name>.port_b_read_during_write_mode = <port B
      read-during-write mode>;

defparam <block_name>.clk0_input_clock_enable = <clock-enable
      source for clk0 when it feeds input registers>;
defparam <block_name>.clk0_core_clock_enable = <clock-enable
      source for clk0 when it feeds memory core>;
defparam <block_name>.clk0_output_clock_enable = <clock-enable
      source for clk0 when it feeds output registers>;
defparam <block_name>.clk1_input_clock_enable = <clock-enable
      source for clk1 when it feeds input registers>;
defparam <block_name>.clk1_core_clock_enable = <clock-enable
      source for clk1 when it feeds memory core>;
defparam <block_name>.clk1_output_clock_enable = <clock-enable
      source for clk1 when it feeds output registers>;
```

3.1 RAM Input Signals

<*block_name*> is the unique identifier for this particular block. *This field is required.*

In ‘single-port’ or ‘rom’ modes, the B port is not used, therefore all the port B inputs are optional.

The memory supports two clocks, four clock-enables and two clears signals. Each clock has two clock-enable controls. Registers using the clk0 can have the clock-enable choice of ena0 or ena2. Registers using the clk1 can have the choice of clock-enable ena1 or ena3.

Port A inputs (and core) will always use clk0. Port B inputs (and core) have the choice of using clk0 or clk1. But the inputs and core must use the same clock source. However, clock-enable can be selected differently since each clock has two choices of clock-enable source.

.portadatain(<*data sources*>**),** is the port A write data input bus to this RAM block. The port A data input bus is always registered by clk0 signal. The valid bus width will depend on the operation mode and the block type parameter. Please refer to table 1 for the valid option. If block type is auto, the valid bus width will be assumed to be the same as M144K. This port is optional. ROM mode does not use this port since it only does read operation. For all other modes, this port is required.

.portaaddr(<*addresses*>**),** are the address inputs for port A. Just like the port A data input bus, the port A address bus is always registered by clk0 signal. The valid address bus width will depend on the operation mode and the block type. Refer to table 1 for the valid address bus width. If block type is auto, the valid bus width will be assumed to be the same as M144K. This port is required.

.portawe(<*write-enable source*>**),** is the (active-high) signal that makes the port A active for writing. *It should be the same signal for all blocks of the same logical RAM.* This port is also always registered by clk0 signal and it is optional. ROM mode does not need this port since it does not allow writing. For all other operation modes, this port is required.

.portare(<*read-enable source*>**),** is the (active-high) signal that makes the port A active for reading. *It should be the same signal for all blocks of the same logical RAM.* This port is also always registered by clk0 signal and it is optional. Simple dual-port mode does not need this port since it only performs writing.

.portabyteenamasks(<*byte masks*>**),** are the byte enable masks for the port A write port. This byte-enable mask port is always registered by the clk0 signal. *This port is optional*. The valid bus width for the byte-enable masks depends on the data bus width and the block type. The byte-enable mask bus width should be equal to the data bus width divided by byte size. The total byte-enable masks (including port B) should not exceed 4 for M9K, and 8 for M144K. When the byte-mask input is 0, the corresponding data bytes are excluded from the write operation.

Note: the byte-enable port can't be used when ECC feature is engaged.

.**portaaddrstall**(<*address-stall source for port A address*>), is the (active-high) signal that is used to hold the port A address value for as long as it is enabled. *This port is optional*.

.portbdatain(<*data sources*>**),** is the port B write data input bus to this RAM block. The port B data input bus is always registered. You can specify the clock source through *port_b_data_in_clock* parameter to choose between clk0 and clk1. This port is optional. It will be used only in true dual-port mode. For valid bus width, please refer to table 1 for details.

.portbaddr(<*addresses*>**),** are the address inputs for port B. Similar to the port B data input bus, the port B address bus is always registered. You can specify the clock source through *port_b_address_clock* parameter to choose between clk0 and clk1. The valid address bus width will depend on the operation mode and the block type. Refer to table 1 for the valid address bus width. If block type is auto, the valid bus width will be assumed to be the same as M144K. This port is required. Please see table 1 for valid address bus width in each mode of different block type.

.portbwe(<*write-enable source*>**),** is the (active-high) signal that makes the port B active for writing. *It should be the same signal for all blocks of the same logical RAM.* This port is

also always registered. You can specify the clock source through *port_b_write_enable_clock* parameter to choose between clk0 and clk1. This port is optional.

.portbre(<*read-enable source*>**),** is the (active-high) signal that makes the port B active for reading. *It should be the same signal for all blocks of the same logical RAM.* This port is also always registered. You can specify the clock source through *port_b_read_enable_clock* parameter to choose between clk0 and clk1. This port is optional.

.portbbyteenamasks(<*byte masks*>**),** are the byte enable masks for the port B write port. This byte-enable mask port is always registered. And you can specify the clock source through *port_b_byte_enable_clock* parameter to choose between clk0 and clk1. *This port is optional*. The valid bus width for byte-enable masks depends on the corresponding data input bus and the block type. The byte-enable mask bus width should be equal to the data bus width divided by byte size. When the byte-mask input is 0, the corresponding data bytes are excluded from the write operation.

Note: the byte-enable port can't be used when ECC feature is engaged.

.**portbaddrstall**(<*address-stall source for port B address*>), is the (active-high) signal that is used to hold the port B address value for as long as it is enabled. *This port is optional*.

.clk0(<*clock source*>**),** designates one of the clocks for the address, data, output, and enable registers. *This signal should be the same signal for all blocks of the same logical RAM.* This port is required since port A inputs are always registered by clk0.

.clk1(<*clock source*>**),** designates one of the clocks for the address, data, output, and enable registers. *This signal should be the same signal for all blocks of the same logical RAM.* This port is optional.

.ena0(<*clock-enable 0*>**),** is the clock enable for .clk0. Only allowed when the .clk0 port on the RAM block is specified. This port is optional.

.ena1(<*clock-enable 1*>**),** is the clock enable for .clk1. Only allowed when the .clk1 port on the RAM block is specified. This port is optional.

.ena2(<*clock-enable 2*>**),** is the additional clock enable for .clk0. Only allowed when the .clk0 port on the RAM block is specified. This port is optional.

.ena3(<*clock-enable 3*>**),** is the additional clock enable for .clk1. Only allowed when the .clk1 port on the RAM block is specified. This port is optional.

.clr0(<*clear source*>**),** is one of the clear signals for the RAM. The clear signal can be used to reset the address registers, the output latches and the output registers. This port is optional.

.clr1(<*clear source*>**),** is one of the clear signals for the RAM. The clear signal can be used to reset the address registers, the output latches and the output registers. This port is optional.

3.2 RAM Output Signals

.portadataout(<*data outputs*>**),** is the A port read data output bus of this RAM block. The output can be registered or not registered by specifying the *port_a_data_out_clock parameter.* If the parameter is none, the output is not registered. Otherwise, the parameter will designate the clock source for the output registers. In addition, the outputs (registered or not) can be asynchronously cleared by clr0/clr1 signal through parameter *port_a_data_out_clear*.

.portbdataout(<*data outputs*>**),** is the B port read data output bus of this RAM block. The output can be registered or not registered by specifying the *port_b_data_out_clock* parameter. If the parameter is none, the output is not registered. Otherwise, the parameter will designate the clock source for the output registers. In addition, the outputs (registered or not) can be asynchronously cleared by clr0/clr1 signal through parameter *port_a_data_out_clear*.

.eccstatus(<*error correction/detection status*>**),** is the status of error correction/detection operation. The output can be registered or not registered by specifying the *eccstatus_clock* parameter. If the parameter is none, the output is not registered. Otherwise, the parameter will designate the clock source for the ECC status outputs. In addition, the outputs (registered or not) can be asynchronously cleared by clr0/clr1 signal through parameter *eccstatus_clear*.

3.3 RAM Modes

There are two types of information fields: fields with logical RAM-wide information and memory block-specific fields.

An important note here is that although the port B inputs can choose between clk0 and clk1, the inputs on the same port do not allow the use of different clocks. They must be synchronous to the same clock. So if port B data in uses clk0, port B addresses can't use clk1. Besides, the addition of extra clock-enable allows user to designate different clock-enable for inputs vs. the core.

Logical RAM-block wide information fields:

<*operation mode*> is one of *{single_port, dual_port, bidir_dual_port, rom}*. It specifies what functionality this memory block implements. *This field is required. It should be the same for all blocks of the same logical RAM.*

<*enable the error correction/detection feature*> is one of *{true or false}*. It specifies whether the ECC feature should be enabled. *This field is optional and defaults to false. It should be the same for all blocks of the same logical RAM.*

<*mixed port feed through mode*> is one of *{ dont_care, old }*. *This field is optional.* It only makes sense in dual_port or bidir_dual_port modes. The field is used to dictate the behavior of 'read-during-write on different ports' at the same location with the same clock. When it's 'dont_care', it means the read output is 'unknown'. When it's 'old', it means the read output is the old data in the address before the write occurs. The default value is 'dont_care'.

<*ram block type*>, is one of *{M9K, M144K, or auto}. This field is optional.* The default value is auto. This parameter will constrain the compiler where to place this RAM instance. When it's auto, the compiler will have the most freedom to move it between memory block types depending on logic memory functionality and configuration.

Note: It's recommended not to use this parameter if you intend to use auto since this is the default behavior. This has the advantage for the user to further constrain the block type for encrypted IP. If the parameter is specified, the user can't override it.

<*logical RAM's name*>, is the unique identifier for the corresponding logical RAM. *This field is required. It should be the same name for all blocks of the same logical RAM.*

<*name of the initialization file*>, is an identifier for the memory initialization file (.mif or .hex). *This field is optional (memory is not initialized). It should be the same file for all blocks of the same logical RAM.*

<*layout of the initialization file*>, is one of *{Port_A, Port_B}. This field is optional.* The default is Port_A. It indicates how the memory initialization file is organized(.mif or .hex). If it's

Port_A, the memory initialization file stores the memory as *port_a_logical_ram_depth* words with word size *port_a_logical_ram_width.*

<*data interleave width in bits*>, <*data interleave offset in bits*>, these two parameters, combined with the first bit parameter and the data width parameter, specifies what logical bits are contained in a RAM block. They are common for all possible views (portA read, portA write, portB read, portB write) and default to 1.

<*port A depth of the logical RAM* >, represents the logical depth of the A port of the corresponding logical RAM. *This field is required. It should be the same value for all blocks of the same logical RAM.*

<*port A width of the logical RAM*>, represents the logical width of the A port of the corresponding logical RAM. *This field is required. It should be the same value for all blocks of the same logical RAM.*

Note: The input registers (data-in, write-enable, address) are always clocked by clk0 for port A. So there is no need to specify the clock parameter here for port A. Also, since the inputs for port A are always registered, it's illegal to have clk0 unconnected.

<*port A data out clock*> is one of *{clock0, clock1, none}*. Designates the clock for the *port A* data output registers. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none*.

<*port A data out clear*> is one of *{clear0, clear1, none}*. Designates the asynchronous clear for the *port A* data outputs (registered or not). *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none.*

<*port A address clear*> is one of *{clear0, none}*. Designates the asynchronous clear for the *port A* address registers. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none.*

<*port A read-during-write mode*> is one of *{ new_data_with_nbe_read, new_data_no_nbe_read, old_data, dont_care }. This field is optional.* The field is used to dictate the port A behavior of 'read-during-write on the same port. When it's 'new_data_with_nbe_read', it means the read output will be the new data for data bytes that are not masked out. As for data bytes that are masked out, the output will be the content in the memory array at the current write location. When it's set to 'new_data_no_nbe_read', the non masked bytes will show new data. But the masked bytes will not change; they will maintain the status from previous operation. This setting is the Stratix II behavior. We model it as "X" (unknown) for masked bytes (see also section 1.2.4). When it's set to 'old_data', it means the read output is the old data in the write address before the write occurs. 'dont_care' setting is provided for possible LUTRAM conversion. Since the LUTRAM does not support any same-port read-during-write mode, the 'dont_care' setting is useful to hint the compiler that the RAM instance can be moved to LUTRAM if user does not care about the same-port read-during-write behavior. The default value is 'new_data_no_nbe_read' for backward compatibility with Stratix II.

<*port B depth of the logical RAM* >, represents the logical depth of the B port of the corresponding logical RAM. *This field is optional. It should be the same value for all blocks of the same logical RAM.*

<*port B width of the logical RAM*>, represents the logical width of the B port of the corresponding logical RAM. *This field is optional. It should be the same value for all blocks of the same logical RAM.*

Note: The physical memory used by a logical RAM must add up the same from all the ports of the RAM. So if the port A mode is 4Kx1, the port B mode can be 128x32, but cannot be 128x16 since this is not an exact match.

<*port B data in clock*> is one of *{clock0, clock1}*. Designates the clock for the port B data input registers. *This field is optional*, and should be the same value for all blocks of the same logical RAM. If port B is used, this field is required.

<*port B address clock*> is one of *{clock0, clock1}*. Designates the clock for the port B address registers. *This field is optional*, and should be the same value for all blocks of the same logical RAM. If port B is used, this field is required.

<*port B write-enable clock*> is one of *{clock0, clock1}*. Designates the clock for the *port B write-enable* register. *This field is optional*, and should be the same value for all blocks of the same logical RAM.

<*port B read-enable clock*> is one of *{clock0, clock1}*. Designates the clock for the *port B read-enable* register. *This field is optional*, and should be the same value for all blocks of the same logical RAM.

<*port B byte-enable clock*> is one of *{clock0, clock1}*. Designates the clock for the *port B byte-enable* register. *This field is optional*, and should be the same value for all blocks of the same logical RAM. If port B byte-enables are connected, this field is required. Otherwise, this field should not be used.

Note: The clock settings on port B inputs must be the same.

<*port B data out clock*> is one of *{clock0, clock1, none}*. Designates the clock for the *port B* data output registers. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none*.

<*port B data out clear*> is one of *{clear0, clear1, none}*. Designates the asynchronous clear for the *port B* data outputs (registered or not). *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none.*

<*port B address clear*> is one of *{clear0, clear1, none}*. Designates the asynchronous clear for the *port B* address registers. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none.*

<*port B read-during-write mode*> is one of *{ new_data_with_nbe_read, new_data_no_nbe_read, old_data }*. *This field is optional.* The field is used to dictate the port B behavior of 'read-during-write on the same port. When it's 'new_data_with_nbe_read', it means the read output will be the new data for data bytes that are not masked out. As for data bytes that are masked out, the output will be the content in the memory array at the current write location. When it's set to 'new_data_no_nbe_read', the non masked bytes will show new data. But the masked bytes will not change, they will maintain the status from previous operation. This setting is the Stratix II behavior. We model it as "X" (unknown) for masked bytes. When it's set to 'old_data', it means the read output is the old data in the write address before the write occurs. The default value is 'new_data_no_nbe_read' for backward compatibility with Stratix II.

Note: Unlike port A, dont_care setting is not provided in port B. This is because that LUTRAM can't support true dual-port mode and this behavior in port B is only meaningful if the RAM is in true dual-port mode. Since we can't move any true dual-port RAM to LUTRAM, there is no need to provide the 'dont_care' setting for resource balancing consideration.

<*clock-enable source for clk0 when it feeds input registers*> is one of *{ena0, ena2, none}*. It designates the clock-enable for the input registers that use clk0. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none*.

<*clock-enable source for clk0 when it feeds memory core*> is one of *{ena0, ena2, none}*. It designates the clock-enable for memory core access that uses clk0. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none*.

<*clock-enable source for clk0 when it feeds output registers*> is one of *{ena0, none}*. It designates the clock-enable for the output registers that use clk0. *This field is optional*, and should

be the same value for all blocks of the same logical RAM. Defaults to *none*. Note here, output registers has have no access to the additional clock-enable (ena2).

Note: Port A inputs will always use clk0, so Port A inputs only have access to ena0 and ena2. As for the memory core access, it must use the same clock as the input clock on the same port. So the port A memory core access only has the choice of ena0 and ena2. Port A data out, port B inputs, port B memory core, and port B outputs have the access to all 4 clock-enables, but it depends on which clock the port selects. Finally, the input clock and core clock share the same first stage clock mux that decides whether the clock should be gated by original clock-enables (ena0/ena1). So we can't have one of them set to the original enables (ena0/ena1) while the other one set to none. If one of them (input clock and core clock) is not gated by the clock-enable and the other is gated by clock-enable, the gated one must use the new additional clock-enables (ena2/ena3).

<*clock-enable source for clk1 when it feeds input registers*> is one of *{ena1, ena3, none}*. It designates the clock-enable for the input registers that use clk1. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none*.

<*clock-enable source for clk1 when it feeds memory core*> is one of *{ena1, ena3, none}*. It designates the clock-enable for memory core access that uses clk1. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none*.

<*clock-enable source for clk0 when it feeds output registers*> is one of *{ena1, none}*. It designates the clock-enable for the output registers that use clk1. *This field is optional*, and should be the same value for all blocks of the same logical RAM. Defaults to *none*. Note here, output registers has have no access to the additional clock-enable (ena3).

Memory block-specific information fields:

<*port A starting address for this block*>, represents the port A starting address of this particular block. *This field is required.*

<*port A ending address for this block*>, represents the port A ending address of this particular block. *This field is required.*

<*port A first logical bit position of this block*> gives the first writing bit position of the port A data in bus of this particular block within the corresponding logical RAM. *This field is required.*

<*width of the port A data bus of this block*> gives the width of the port A data in bus of this particular block within the corresponding logical RAM. *This field is required.*

<*width of the port A address bus of this block*> gives the width of the port A address bus of this particular block within the corresponding logical RAM. *This field is required.*

<*width of the port A byte-enable mask bus of this block*> gives the width of the port A byte-enable mask bus of this particular block within the corresponding logical RAM. *This field is required.*

<*port A byte size*> this describes the number of data bits each byte-enable mask controls. For example, if this field is 8, it means that each byte-enable mask signal will affect 8 bits data.

<*port B starting address for this block*>, represents the port B starting address of this particular block. *This field is required if port B is used.*

<*port B ending address for this block*>, represents the port B ending address of this particular block. *This field is required if port B is used.*

<*port B first logical bit position of this block*> gives the first writing bit position of the port B data in bus of this particular block within the corresponding logical RAM. *This field is required if port B is used.*

<*width of the port B data bus of this block*> gives the width of the port B data in bus of this particular block within the corresponding logical RAM. *This field is required if port B is used.*

<*width of the port B address bus of this block*> gives the width of the port B address bus of this particular block within the corresponding logical RAM. *This field is required.*

<*width of the port B byte-enable mask bus of this block*> gives the width of the port B byte-enable mask bus of this particular block within the corresponding logical RAM. *This field is required.*

<*port B byte size*> this describes the number of data bits each byte-enable mask controls. For example, if this field is 8, it means that each byte-enable mask signal will affect 8 bits data.

3.4 Registering Modes of I/O signals

The supported RAM I/O register modes are the same as for Stratix II. Stratix III RAM is always registered on the input side except for signals like clears, clock-enables, and address stalls. This is shown in Table 2 below:

Table 2 -- Stratix III RAM Registering Modes

Operation Mode	Write Logic*	Read Logic**	Data In	Data Out
Single Port	Reg.	Reg.	Reg.	Comb/Reg.
Simple Dual Port	Reg.	Reg.	Reg.	Comb/Reg.
True Dual Port	Reg.	Reg.	Reg.	Comb/Reg.
ROM	N/A	Reg.	N/A	Comb/Reg.

Write Logic* – address, write-enable, and byte-enable

Read Logic** - address, read-enable, and byte-enable

3.5 Polarities and Default Values

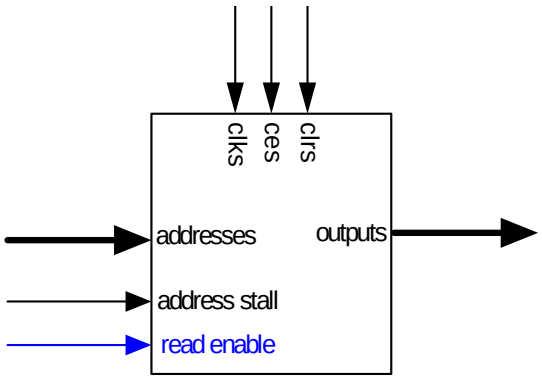
All signals are active high and all secondary inputs have programmable inversions. Upon power-up, all input registers will be 0 with the exception of read-enable and byte-enable registers that will be 1.

4. Operation Modes

This section describes the four operation modes supported by the Stratix III memory blocks. They are ROM mode, single-port mode, dual-port (simple dual-port) mode and bidir dual-port (true dual-port) mode.

4.1 ROM (Read-Only Memory) Mode

ROM is a pre-initialized memory block that can only perform read operations. The following picture illustrates its I/O interfaces. Addresses are always registered by clk0 input. The outputs can be unregistered or registered. When the outputs are registered, they can be registered by clk0 or clk1. However, all outputs must be registered by the same clock. Only output registers can be cleared. And the clear source can be clr0 or clr1. This mode is supported by the M9K and the M144K blocks.

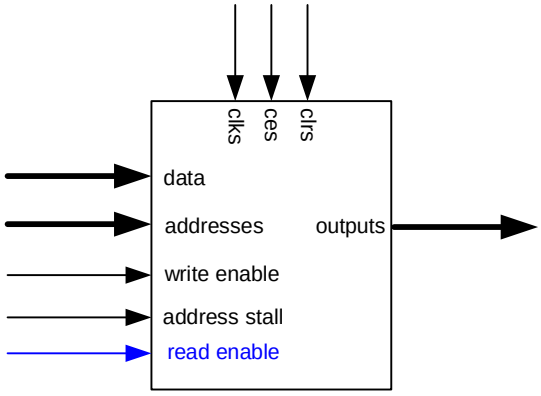


The followings show how the above functional ports map to the WYSIWYG.

addresses	-> portaaddr
address stall	-> portaaddrstall
outputs	-> portadataout
read enable	-> portare
clks	-> clk0, clk1
ces	-> ena0, ena1, ena2, ena3
clrs	-> clr0, clr1

4.2 Single-Port Mode

In a single-port RAM, only one location of the RAM can be accessed at a time. Read and write operations can be performed on the memory block. The following picture shows its I/O interfaces. Data, addresses and write-enable are always registered by clk0. The outputs can be unregistered or registered. Again, all outputs must be registered by the same clock (either clk0 or clk1). And only output registers can be cleared. The clear can come from clr0 or clr1.

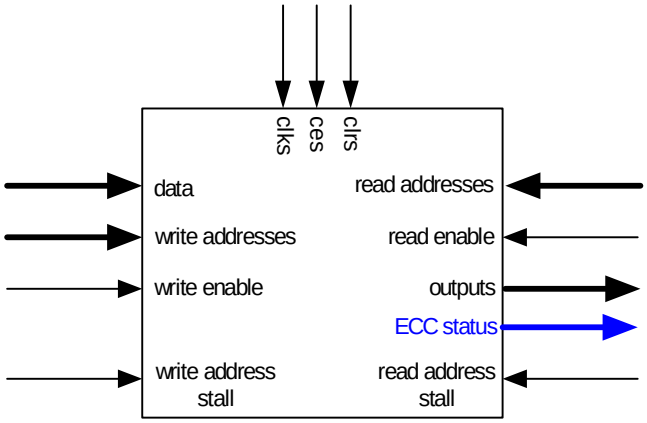


The followings show how the above functional ports map to the WYSIWYG.

data	-> portadatain
addresses	-> portaaddr
write enable	-> portawe
read enable	-> portare
address stall	-> portaaddrstall
outputs	-> portadataout
clks	-> clk0, clk1
ces	-> ena0, ena1, ena2, ena3
clrs	-> clr0, clr1

4.3 Simple Dual-Port Mode

This mode is also known as dual-port mode. In simple dual-port mode, 1 read and 1 write operation (1R1W) can be performed on different locations at the same time. However, the RAM can't perform 2 reads or 2 writes at the same time. The following picture shows the I/O interfaces. The read port and the write port can be separately registered. The write-port must be registered by clk0, and the read-port can be registered by either clk0 or clk1. Only output registers can be cleared. The clear signal can come from either clr0 or clr1. Also, the read data width and write data width can be different, but one must be a multiple of the other. The Stratix III M144K memory block also supports the ECC feature in this mode. When the ECC feature is enabled, the byte-enable feature cannot be used.

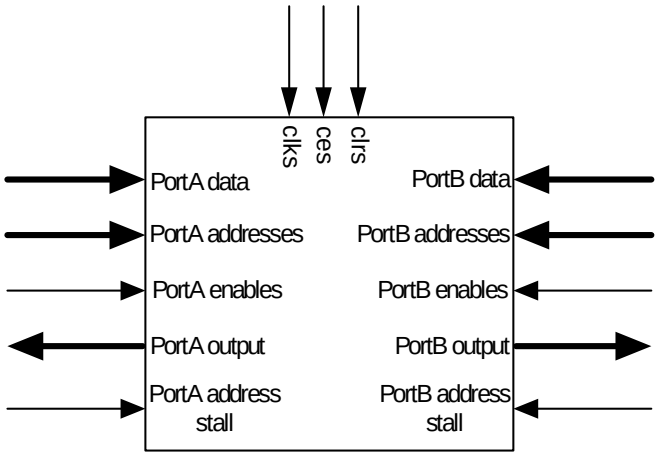


The followings show how the above functional ports map to the WYSIWYG.

data	-> portadatain
write addresses	-> portaaddr
write enable	-> portawe
write address stall	-> portaaddrstall
read addresses	-> portbaddr
read enable	-> portbrewe
outputs	-> portbdataout
ECC status	-> eccstatus
read address stall	-> portbaddrstall
clks	-> clk0, clk1
ces	-> ena0, ena1, ena2, ena3
clrs	-> clr0, clr1

4.4 True Dual-Port Mode

This mode is also known as bidir dual-port. In true dual-port mode, 2W, 1R1W, or 2R operations can be performed at different locations at the same time. The following picture shows the I/O interfaces. Port A and Port B can be separately registered. Port A inputs (data, address and enable) must be registered by clk0. Port B inputs can be registered by either clk0 or clk1, but all port B inputs must be registered by the same clock. Both Port A and Port B outputs can be registered separately by either clk0 or clk1. They can be cleared by clr0 or clr1. Only output registers can be cleared. Also, the port A data width and port B data width can be different, but one must be a multiple of the other.



The followings show how the above functional ports map to the WYSIWYG.

PortA data	-> portadatain
PortA addresses	-> portaaddr
PortA enables	-> portawe, portare
PortA output	-> portadataout
PortA address stall	-> portaaddrstall
PortB data	-> portbdatain
PortB addresses	-> portbaddr
PortB enables	-> portbwe, portbre
PortB output	-> portbdataout
PortB address stall	-> portbaddrstall
clks	-> clk0, clk1
ces	-> ena0, ena1, ena2, ena3
clrs	-> clr0, clr1