Cyclone III RAM WYSIWYG User Guide

Version 1.0

March 9, 2009

by

Altera Corporation

Table of Contents:

1. Overview

This document describes the WYSIWYG primitive for the Cyclone III memory block. The Cyclone III memory block is identical to the Stratix III M9K memory block in terms of feature sets.

Please see the document stratixiii_ram_wys_eda.pdf for the WYSIWYG description details.

1.1 Cyclone III RAM block

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

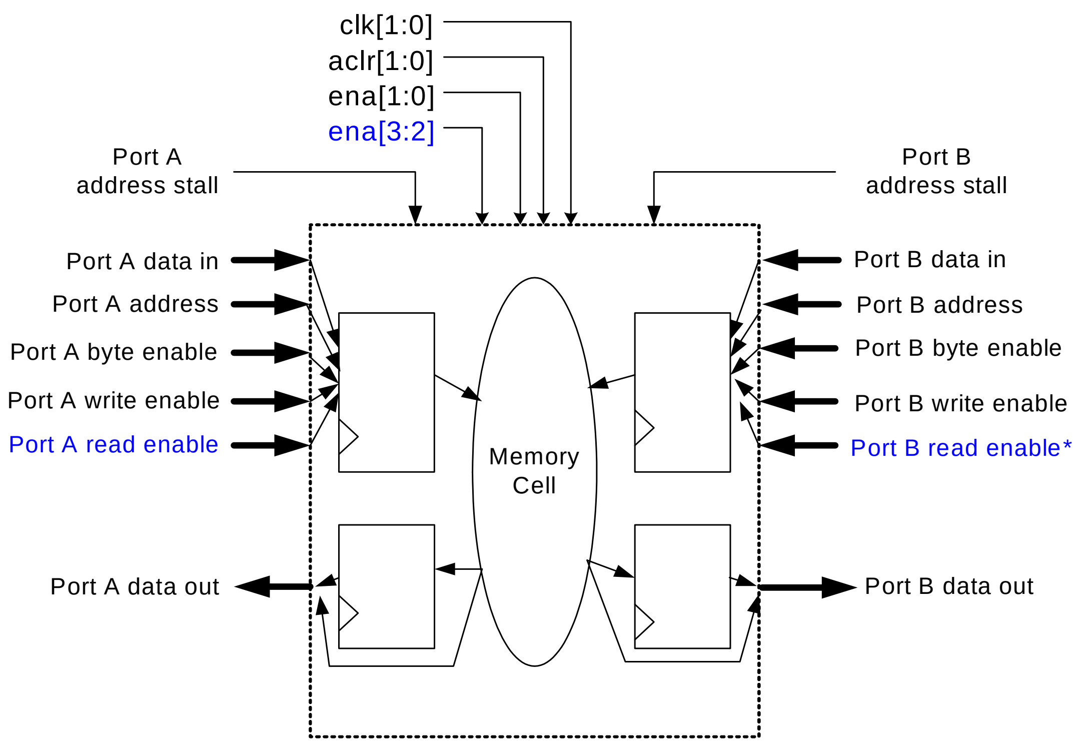

Figure 1: Cyclone III RAM

* In Cyclone II, port B read enable is shared with port B write-enable

2. RAM Operation Modes

The Cyclone III memory block is identical to the Stratix III M9K memory block. Please see the document stratixiii_ram_wys.doc for supported operation modes.

3. RAM Primitive

The Cyclone III RAM WYSIWYG primitive is mostly identical to the Stratix III RAM WYSIWYG primitive with the exception of the features that are only supported by the M144K memory block. Therefore, the ram_block_type parameter can only be set to { M9K, M8K, AUTO }. Please see stratixiii_ram_wys.doc for a detailed description. The following shows the WYSIWYG format.

```
cycloneiii_ram_block <block_name>
(
      // Port A inputs
      .portadatain(<port A write data source bus>),
      .portaaddr(<port A addresses bus>),
      .portawe(<port A write-enable source>),
      .portare(<port A read-enable source>),
      .portabyteenamasks(<port A byte-enable mask source bus>),
      .portaaddrstall(<port A address stall source>),

      // Port B inputs
      .portbdatain(<port B write data source bus>),
      .portbaddr(<port B addresses bus>),
      .portbwe(<port B write-enable source>),
      .portbre(<port B read-enable source>),
      .portbbyteenamasks(<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(<port A read data output bus>),

      // Port B outputs
      .portbdataout(<port B read data output bus>),
);
defparam <block_name>.operation_mode = <operation mode>;
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>;
```