Cyclone II RAM WYSIWYG User Guide

Version 1.2

January 21, 2004

Table of Contents:

1. Overview

This document describes the WYSIWYG primitive for Cyclone II memory block. Cyclone II memory block is functionally equivalent to the Stratix II M4K block. Unlike Stratix/Stratix II, it only contains 1 type of memory block – M4K. For details about Stratix II RAM primitive, please refer to Stratix II RAM Wysiwyg User Guide stratixii_ram_wys_user.doc.

Feature-wise, Cyclone II memory is almost a superset of Cyclone memory except that it does not allow asynchronous clear on the input registers as Cyclone. Asynchronous clear is only available on the output registers in Cyclone II.

1.1 Cyclone II RAM block

The following picture shows the abstract functional I/O interface for Cyclone II memory block.

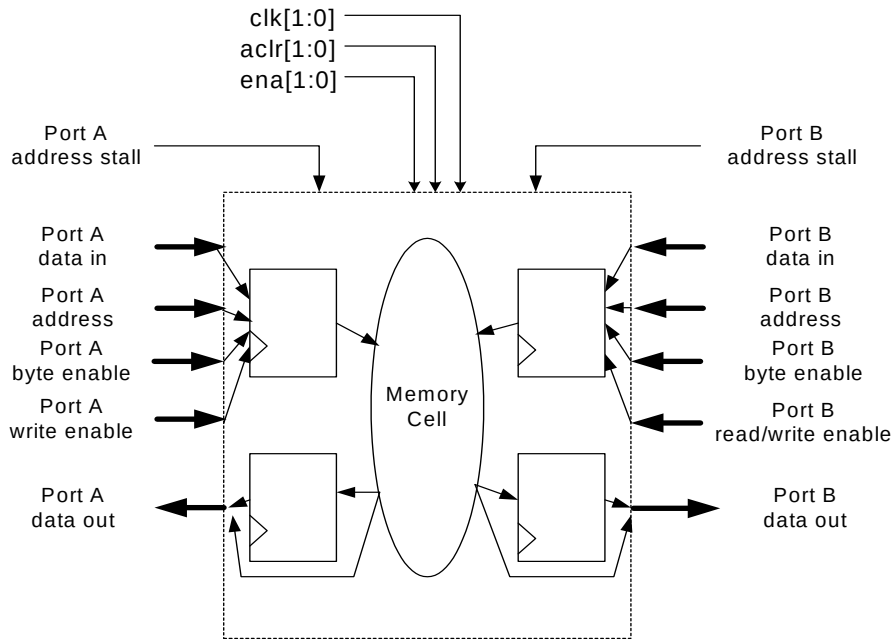


Figure 1: CycloneII RAM

1.2 Improvements from Cyclone

Since Cyclone II memory block is functionally identical to Stratix II M4K block, it inherits all improvements of the Stratix II M4K block.

1.2.1 Byte-Enable

Byte-enable control has been made more flexible in general. It will be available in x1, x2, x4, and x9 modes as well. In x1, x2, x4 and x9 modes, it will act similar to a write-enable. Either all bits are enabled or disabled by the single-bit byte-enable control signal. This feature is mainly added to help fitting and performance. Currently, in Cyclone, given a 512x18 RAM with two byte-enable controls, the RAM has to be re-partitioned into multiple shallower pieces with external stitched logic since M4K does not support byte-enable in 512x9 mode. With this new feature, we can then partition the 512x18 RAM into 2 M4Ks in 512x9 mode without the external stitching logic. Each M4K will get 1 byte-enable control.

1.2.2 Address-Stall

Address-Stall feature has been added to M4K in Cyclone II as compared to Cyclone. Address stall is used to hold the previous address value for as long as the stall signal is enabled. This is to improve the efficiency in cache-miss applications.

1.2.3 Improved clock-enable control

Cyclone II memory blocks have more flexible clock-enable control. In Cyclone, once the clock is paired with clock-enable, all registers fed by the clock will get the clock-enable. This has been causing problems in our DCFIFO (dual-clock FIFO) megafunction implementation. The DCFIFO megafunction requires two clocks, one for read and one for write. However, it also requires gating the read clock on the output registers with clock-enable while leaves the read clock not-gated on read input registers. This is something Cyclone hardware can't support. And this incurs some area overhead on the DCFIFO implementation in Cyclone.

In Cyclone II, with the more flexible clock-enable control, the DCFIFO megafunction can use the clock-enable independently on input registers vs. output registers. It no longer needs the extra overhead logic.

The following table dictates all different modes supported by Cyclone II memory block:

Operation Mode	Single-port / ROM[1]	Simple dual-port (dual_port)	True dual-port (bidir_dual_port)
Configuration	4Kx1 2Kx2 1Kx4 512x9 256x18 128x36	WxM / RxN or WxY / RxZ M,N is (1,2,4,8,16,32) Y,Z is (9,18,36)	AxM/BxN [2] or AxY/BxZ [2] M,N is (1,2,4,8,16), M ≥ N Y,Z is (9, 18), Y ≥ Z

Table 1 Cyclone II RAM Configurations

Note:

1. Since all inputs to Cyclone II RAM need to be registered, the ROM mode really means a pipelined ROM.
2. For users of mixed-width Bidir dual-port RAM, the RAM is still symmetric because the megafunction will swap the ports when the given B port is wider than A port. However, when you write RAM WYSIWYG primitive, you have to make sure Port A is wider than Port B.

2. RAM Primitive

The following is the WYSIWYG primitive that defines the Cyclone II memory block.

```
cycloneii_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>),
      .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>),
      .portbrewe(<port B read-enable/write-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>),
      .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_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_disable_ce_on_input_registers =
      <port A disable clock-enable on input registers>;
defparam <block_name>.port_a_disable_ce_on_output_registers =
      < port A disable clock-enable on output registers>;
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_read_enable_write_enable_clock =
      <port B read-enable/write-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_disable_ce_on_input_registers =
      <port B disable clock-enable on input registers>;
defparam <block_name>.port_b_disable_ce_on_output_registers =
      < port B disable clock-enable on output registers>;
```

2.1 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 block only supports two clocks, clock-enables and clears signals.

.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 clock0 signal. The valid bus width will depend on the operation mode. Please refer to table 1 for the valid option. 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 clock0 signal. The valid address bus width will depend on the operation mode. Refer to table 1 for the valid address bus width. 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 clock0 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.

.portabyteenamasks(<*byte masks*>**),** are the byte enable masks for the port A write port. This byte-enable mask port is always registered by clock0 signal. *This port is optional*. The valid bus width for the byte-enable masks depends on the data bus width. 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.

.**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 clock0 and clock1. 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 clock0 and clock1. The valid

address bus width will depend on the operation mode. Refer to table 1 for the valid address bus width. This port is required. Please see table 1 for valid address bus width in each mode.

.portbrewe(<*read-enable/write-enable source*>**),** is the (active-high) signal that makes the port B active for reading(or 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_read_enable_write_enable_clock* parameter to choose between clock0 and clock1. This port is optional.

Note: In bidir_dual_port mode, this port will be used as write-enable and it's also 'active-high'.

.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 clock0 and clock1. *This port is optional*. The valid bus width for byte-enable masks depends on the corresponding data input bus. The byte-enable mask bus width should be equal to the data bus width divided by byte size.

.**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.

.clr0(<*clear source*>**),** is one of the clear signals for the RAM. The clear signal can only clear the output registers. This port is optional.

.clr1(<*clear source*>**),** is one of the clear signals for the RAM. The clear signal can only clear the output registers. This port is optional.

2.2 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. Also, the output register can be asynchronously cleared by clear0/clear1 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. Also, the output register can be asynchronously cleared by clear0/clear1 signal through parameter *port_a_data_out_clear*.

2.3 RAM Modes

There are two types of information fields: fields which give Logical RAM-wide information,and fields that are block-specific. An important note here is although the B port inputs can choose between clock0 and clock1, the inputs on the same port do not allow to use different clocks. They must be synchronous to the same clock. So if port B data in uses clock0, port B addresses can't use clock1.

RAM-block wide information fields are as follows:

<*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.*

<*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 always *{ M4K}* since Cyclone II only contains the M4K memory block*. This field is optional.* It will be default to M4K.

<*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.

RAM-block wide port information fields are as follows:

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

<*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.

<*port B 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 B data out clear*> is one of *{clear0, clear1, none}*. Designates the asynchronous clear 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.*

The following information fields are block-specific:

<*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. The byte size should not exceed 9.

<*port A disable clock-enable on input registers*> is one of { on, off }. When it is on, the clock to the port A input registers will not be gated by the clock-enable. When it is off, the clock to the port A input registers will be gated by its corresponding clock-enable signal.

<*port A disable clock-enable on output registers*> is one of { on, off }. When it is on, the clock to the port A output registers will not be gated by the clock-enable. When it is off, the clock to the port A output registers will be gated by its corresponding clock-enable signal.

<*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. The byte size should not exceed 9.

<*port B disable clock-enable on input registers*> is one of { on, off }. When it is on, the clock to the port B input registers will not be gated by the clock-enable. When it is off, the clock to the port B input registers will be gated by its corresponding clock-enable signal.

<*port B disable clock-enable on output registers*> is one of { on, off }. When it is on, the clock to the port B output registers will not be gated by the clock-enable. When it is off, the clock to the port B output registers will be gated by its corresponding clock-enable signal.

2.4 Registering Modes of I/O signals

Supported ram registering modes are the same as Cyclone/Stratix. Cyclone II RAM is always registered on input sides. This is shown in Table 2:

Table 2 -- Cyclone II 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.

2.5 Polarities and Default Values

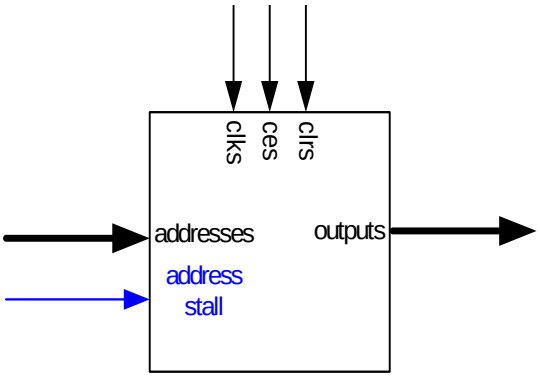
All signals are active high and all secondary inputs have programmable inversions. Upon power-up, all input registers will be 0 except that read-enable and byte-enable registers will be 1.

3. Operation Modes

This section describes the various operation modes supported in Cyclone II memory block. Basically, there are 4 modes supported just like Cyclone. They are ROM, single-port, dual-port and bidir dual-port.

3.1 ROM (Read-Only Memory)

ROM is a pre-initialized memory block that can only perform the read operation. The following picture illustrates its I/O interfaces. The addresses and the outputs can be separately registered.



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

addresses -> portaaddr

address stall -> portaaddrstall

outputs -> portadataout

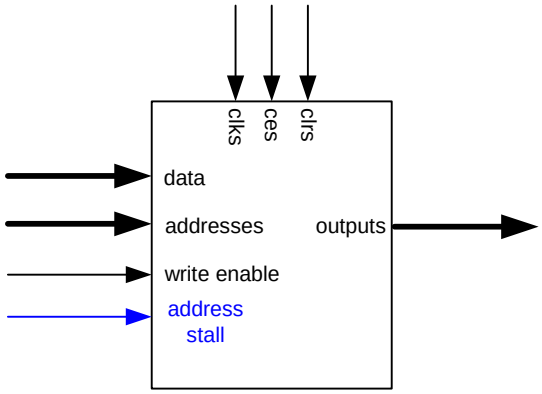
clks -> clk0, clk1

ces -> ena0, ena1

clrs -> clr0, clr1

3.2 Single-Port

In a single-port RAM, you can only access one location of the RAM at one time. You can perform read/write operations on the memory block. The following picture shows its I/O interfaces. The inputs and outputs can be separately registered.

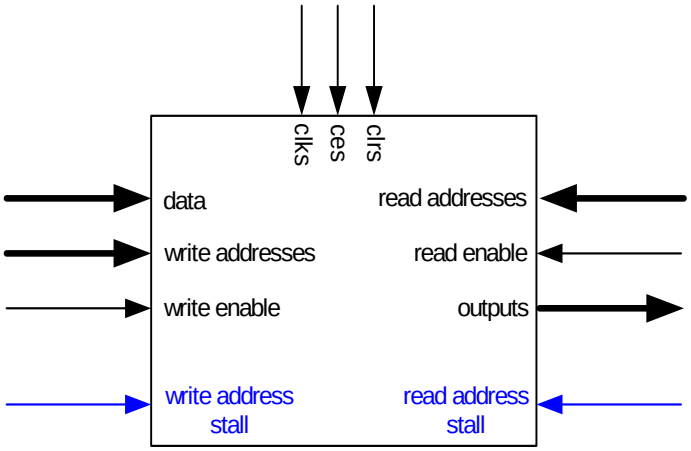


The followings show how the above functional ports map to the WYSIWYG.

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

3.3 Simple Dual-Port (a.k.a. dual-port)

In simple dual-port mode, you can do 1 read and 1 write operation (1R1W) on different locations at the same time. The following picture shows the I/O interfaces. The read port and the write port can be separately registered. Also, the read data width and write data width can be different, but one must be a multiple of the other.

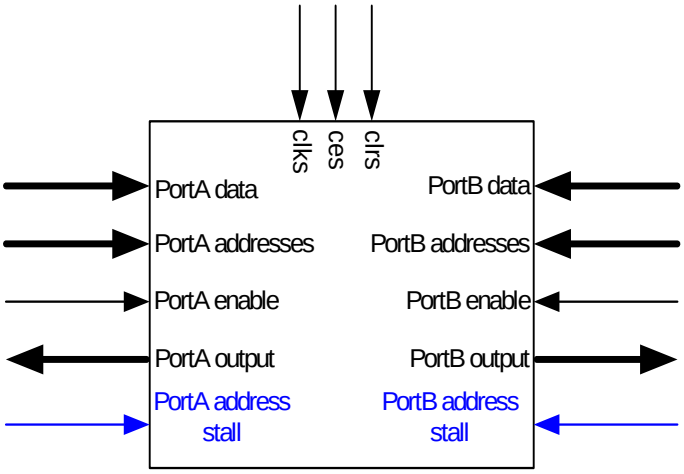


The followings show how the above functional ports map to the WYSIWYG.

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

3.4 True Dual-Port (a.k.a. bidir_dual_port)

In true dual-port mode, you can do 2W/1R1W/2R operations in different locations at the same time. The following picture shows the I/O interfaces. Port A and Port B can be separately registered. 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 enable	-> portawe
PortA output	-> portadataout
PortA address stall	-> portaaddrstall
PortB data	-> portbdatain
PortB addresses	-> portbaddr
PortB enable	-> portbrewe
PortB output	-> portbdataout
PortB address stall	-> portbaddrstall
clks	-> clk0, clk1
ces	-> ena0, ena1
clrs	-> clr0, clr1

4. Internal Input registers

Input registers of a RAM can be referred to using the input register name extensions. Users can make assignments to the input registers by padding the name extensions to the RAM block instance name. The following is a list of RAM internal register name extensions.

Input Register	Name extension	Requires index?
Port A Write Enable	porta_we_reg	N
Port B Read Enable	portb_we_reg	N
Port B Write Enable	portb_re_reg	N
Port A Data In	porta_datain_reg	Y

Port B Data In	portb_datain_reg	Y
Port A Address Register	porta_address_reg	Y
Port B Address Register	portb_address_reg	Y
Port A Byte Enable Register	porta_bytena_reg	Y
Port B Byte Enable Register	portb_bytena_reg	Y

The format of the register name is *<ramblock_instance_ name>~<register_name>[<index>]*.

The “index” is not required for read enable registers and write enable registers.
E.g., *ram~porta_we_reg*
The “index” is required for data-in registers, address registers, and byte enable registers.
E.g., *ram~porta_address_reg9*

5. Example

The following example instantiates a true dual-port (bidir dual-port) RAM that is 2 bits wide on Port A and 1 bit wide on Port B. It is 500 words deep on Port A, and 1000 words deep on Port B. The block type is set to "M4K”.

```
cycloneii_ram_block bdp_a500x2_b1000x1 (
       .portawe( a_we ),
       .portbrewe( b_we ),
       .clk0( clk[0] ),
       .clk1( clk[1] ),
       .ena0( clk_en[0] ),
       .ena1( clk_en[1] ),
       .clr0( clr[0] ),
       .clr1( clr[1] ),
       .portadatain( a_data_in[1:0] ),
       .portaaddr( a_address[8:0] ),
       .portbdatain( b_data_in[0] ),
       .portbaddr( b_address[9:0] ),
       .portadataout( a_data_out[1:0] ),
       .portbdataout( b_data_out[0]));
defparam bdp_a500x2_b1000x1 .operation_mode = "bidir_dual_port";
defparam bdp_a500x2_b1000x1 .ram_block_type = "M4K";
defparam bdp_a500x2_b1000x1 .mixed_port_feed_through_mode = "dont_care";
defparam bdp_a500x2_b1000x1 .logical_ram_name = "RAM_mixed_width_500x2_1000x1";
defparam bdp_a500x2_b1000x1 .data_interleave_width_in_bits = 1;
defparam bdp_a500x2_b1000x1 .data_interleave_offset_in_bits = 1;
defparam bdp_a500x2_b1000x1 .port_a_logical_ram_depth = 500;
defparam bdp_a500x2_b1000x1 .port_a_logical_ram_width = 2;
defparam bdp_a500x2_b1000x1 .port_a_data_in_clear = "none";
defparam bdp_a500x2_b1000x1 .port_a_address_clear = "none";
defparam bdp_a500x2_b1000x1 .port_a_write_enable_clear = "none";
defparam bdp_a500x2_b1000x1 .port_a_byte_enable_clear = "none";
defparam bdp_a500x2_b1000x1 .port_a_data_out_clock = "clock0";
defparam bdp_a500x2_b1000x1 .port_a_data_out_clear = "clear1";
defparam bdp_a500x2_b1000x1 .port_a_first_address = 0;
defparam bdp_a500x2_b1000x1 .port_a_last_address = 499;
defparam bdp_a500x2_b1000x1 .port_a_first_bit_number = 0;
defparam bdp_a500x2_b1000x1 .port_a_data_width = 2;
defparam bdp_a500x2_b1000x1 .port_a_address_width = 9;
defparam bdp_a500x2_b1000x1 .port_b_logical_ram_depth = 1000;
defparam bdp_a500x2_b1000x1 .port_b_logical_ram_width = 1;
defparam bdp_a500x2_b1000x1 .port_b_data_in_clock = "clock1";
defparam bdp_a500x2_b1000x1 .port_b_data_in_clear = "none";
defparam bdp_a500x2_b1000x1 .port_b_address_clock = "clock1";
```

```
defparam bdp_a500x2_b1000x1 .port_b_address_clear = "none";
defparam bdp_a500x2_b1000x1 .port_b_read_enable_write_enable_clock = "clock1";
defparam bdp_a500x2_b1000x1 .port_b_read_enable_write_enable_clear = "none";
defparam bdp_a500x2_b1000x1 .port_b_byte_enable_clear = "none";
defparam bdp_a500x2_b1000x1 .port_b_data_out_clock = "clock1";
defparam bdp_a500x2_b1000x1 .port_b_data_out_clear = "clear0";
defparam bdp_a500x2_b1000x1 .port_b_first_address = 0;
defparam bdp_a500x2_b1000x1 .port_b_last_address = 999;
defparam bdp_a500x2_b1000x1 .port_b_first_bit_number = 0;
defparam bdp_a500x2_b1000x1 .port_b_data_width = 1;
defparam bdp_a500x2_b1000x1 .port_b_address_width = 10;
```