RTG4
User Guide
Dual-Port Large SRAM Configuration

Microsemi
a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

5-02-00604-6.0/08.19

Table of Contents

Introduction

A Dual-Port Large SRAM enables read and write access on both ports, Port A and Port B (Figure 1).

The core configurator automatically cascades Large SRAM blocks to create wider and deeper memories by choosing the most efficient aspect ratio. It also handles the grounding of unused bits. The core configurator supports the generation of memories that have different aspect ratios on each port.

Dual-Port Large SRAM is synchronous for memory write and read operations, setting up the addresses as well as writing and reading the data. The memory write and read operations will be triggered at the rising edge of the clock. The address, data, block-port select, write-enable, and read-enable inputs are registered.

Optional pipeline registers are available at both the read data ports to improve the clock-to-out delay.

When ECC is enabled, output flags are generated to indicate single-bit-correct and double-bit-detect for each port.

In this document, we describe how you can configure a Dual-Port Large SRAM instance and define how the signals are connected. For more details about the Dual-Port Large SRAM, refer to the RTG4 User's Guide.

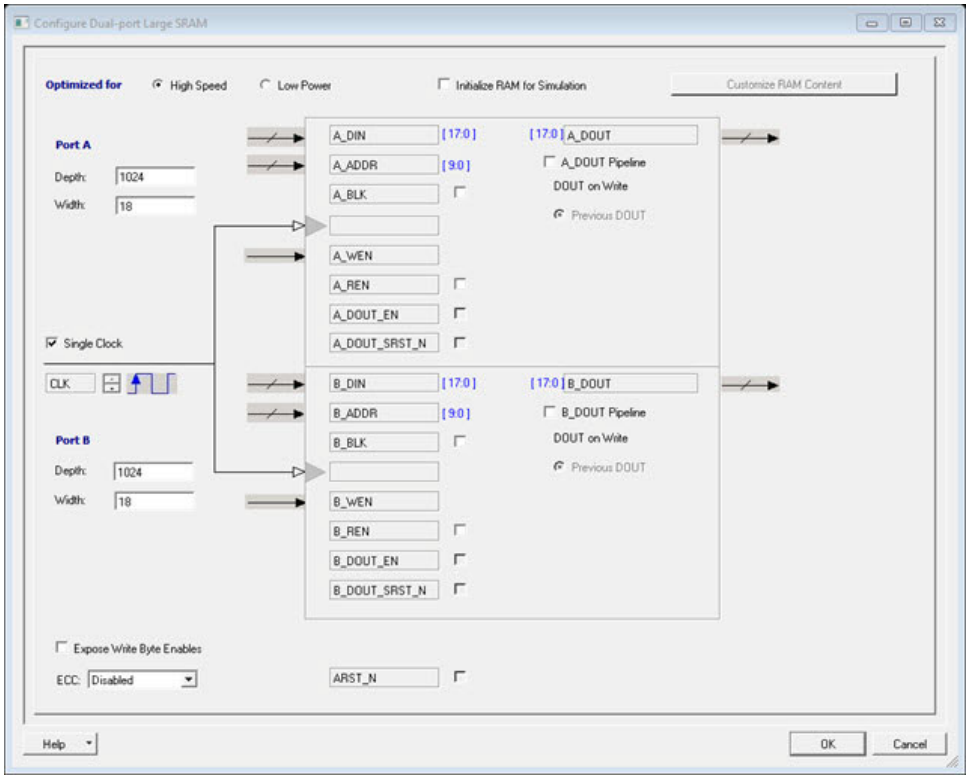

***Figure 1* • Dual-Port Large SRAM Configurator**

1 – Functionality

Optimization for High Speed or Low Power

Selecting High Speed results in a macro optimized for speed and area (width cascading).

Selecting Low Power results in a macro optimized for low power, but uses additional logic at the input and output (depth cascading). Performance for a low power optimized macro may be inferior to that of a macro optimized for speed.

Port A Depth/Width and Port B Depth/Width

The depth range for each port is 1-32768. The width range for each port is 1-3762.

The two ports can be independently configured for any depth and width. (Port A Depth * Port A Width) must equal (Port B Depth * Port B Width).

Single Clock (CLK) or Independent Port A and B Clocks (A_CLK, B_CLK)

The default configuration for Dual-Port Large SRAM is a Single clock (CLK) to drive both A and B ports with the same clock. Uncheck the Single clock checkbox to drive independent clocks - one for each port (A_CLK and B_CLK).

Click the waveform next to any of the clock signals to toggle its active edge.

Block Select (A_BLK, B_BLK) and Read/Write Control (A_WEN, B_WEN)

De-asserting A_BLK forces A_DOUT to zero. De-asserting B_BLK forces B_DOUT to zero.

Asserting A_BLK when A_WEN is low reads the RAM at the address A_ADDR onto the input of the A_DOUT register, on the next rising edge of A_CLK.

Asserting A_BLK when A_WEN is high writes the data A_DIN into the RAM at the address A_ADDR, on the next rising edge of A_CLK.

Asserting B_BLK when B_WEN is low, reads the RAM at the address B_ADDR onto the input of the B_DOUT register, on the next rising edge of B_CLK.

Asserting B_BLK when B_WEN is high, writes the data B_DIN into the RAM at the address B_ADDR, on the next edge of B_CLK.

The default configuration for A_BLK and B_BLK is unchecked, which ties the signal to the active state and removes it from the generated macro. Click the respective checkbox to insert that signal on the generated macro. Click the signal arrow (when available) to toggle its polarity.

Pipeline for Read Data Output of Port A and B

Click the Pipeline checkbox to enable pipelining of Read data (A_DOUT or B_DOUT).

Turning off pipelining of Read data of a port also disables the configuration options of the respective DOUT_EN and DOUT_SRST_N signals.

Register Enable (A_DOUT_EN and B_DOUT_EN)

The pipeline registers for ports A and B have active high, enable inputs. The default configuration is to tie these signals to the active state and remove them from the generated macro. Click each signal's checkbox to insert that signal on the generated macro.

Click the signal arrow (when available) to toggle its polarity.

Synchronous Reset (A_DOUT_SRST_N and B_DOUT_SRST_N)

The pipeline registers for ports A and B have active low, synchronous reset inputs. The default configuration is to tie these signals to the inactive state and remove them from the generated macro. Click each signal's checkbox to insert that signal on the generated macro.

Click the signal arrow (when available) to toggle its polarity.

Read Enable (A/B_REN)

De-asserting A_REN holds the previous Read data on port A and de-asserting B_REN holds the previous Read data on port B.

Asserting the A_REN reads the RAM at the A_ADDR onto port A's Read Data register on the next rising edge of the clock. Similarly, asserting the B_REN reads the RAM at the B_ADDR onto port B's Read Data register on the next rising edge of the clock.

The default configuration for the A_REN or B_REN option is unchecked, which ties the signal to the active state and removes it from the generated macro. Click the checkbox (when available) to insert that signal on the generated macro. Click the signal arrow (when available) to toggle its polarity.

The A_REN option for Port A and the B_REN option for Port B are disabled (greyed-out) in the Configurator if the Depth x Width of the port and the Optimization mode of the macro requires depth cascading and the address space is fractured .

Consider, for example, a 2048x18 Dual-Port Large SRAM macro without ECC. If it is optimized for High Speed, the Configurator generates two RAM1K18_RT blocks, each configured for 2048x9 (width-wise cascading). Because there is no depth-wise cascading, the Configurator enables the A_REN and B_REN options. When each of the checkboxes is selected, the respective signal is exposed as an input port.

When the same 2048x18 Dual-Port Large SRAM macro is optimized for Low Power, the Configurator generates two RAM1K18_RT blocks, each configured for 1024x18 (depth-wise cascading). The Configurator disables the A_REN and B_REN options and these signals cannot be generated.

For a 4096xn Dual-Port SRAM macro, regardless of Low Power or High Speed optimization, depth-wise cascading is necessary, and it causes the address space to be fractured. The Configurator disables the A_REN and B_REN options, and these signals cannot be generated.

Expose Write Byte Enables (A/B_WBYTE_EN)

When enabled, write byte enables (A_WBYTE_EN and B_WBYTE_EN) are available as top-level buses. Each bit of A_WBYTE_EN gated by the A_WEN signal, enables writing to an individual byte of A_DIN. Each bit of B_WBYTE_EN gated by the B_WEN signal, enables writing to an individual byte of B_DIN.

Asynchronous Reset (ARST_N)

The pipeline registers for ports A and B have an active low, asynchronous reset input. The default configuration is to tie this signal to the inactive state and remove it from the generated macro. Click the checkbox to insert the asynchronous active low ARST_N signal on the generated macro.

Click the signal arrow (when available) to toggle its polarity.

Error Correction Code (ECC)

Three options are available for ECC:

- Disabled
- Pipelined
- Non-Pipelined

When ECC is disabled, each port could be configured to either 18 bits or 9 bits width.

When ECC is enabled (Pipelined or Non-Pipelined), both ports have word widths equal to 18 bits.

DOUT Registers Truth Table

Table 1-1 describes the functionality of the control signals on the A_DOUT and B_DOUT registers.

***Table 1-1* • A_DOUT and B_DOUT Registers Truth Table**

ARST_N	_CLK	DOUT_EN	DOUT_SRST_N	D	Q_{n+1}
0	X	X	X	X	0
1	Not rising	X	X	X	Q_n
1	↑	0	X	X	Q_n
1	↑	1	0	X	0
1	↑	1	1	D	D

2 – Internal Connections of the Configurator

This chapter describes an example where Dual-Port LSRAM configuration generates a component with the following address widths.

- A_ADDR[*A_MSB:0*]
- B_ADDR[*B_MSB:0*]

Let *M* be the width of the address on the A-port of each LSRAM block and *N* be the width of the address on the B-port of each LSRAM block.

Let the decoder logic function be *decode*(addr*[j:k], i*), where 0 <= *i* < $2^{(j-k+1)}$.

Let *D* be the depth of an LSRAM block in the array of blocks, starting at 0.

LSRAM Block Port Depth x Width	*M, N*
2Kx9	11
1Kx18	10

A_BLK, A_REN Connections

The **A_BLK** signal on the generated component is connected to the A port block select input (**A_BLK**) for each LSRAM block according to the block depth within the component and synchronized with **A_CLK.** When ECC Pipeline is enabled, the component level **A_BLK** is OR'd with pipelined version of component **A_BLK**, while the component level **A_REN** is AND'd with component level **A_BLK**..

Depth	**A_BLK[2]**	**A_BLK[1]**	**A_BLK[0]**	**A_REN**
A_MSB < *M*	**A_BLK**	1	1	**A_REN**
A_MSB = *M*	**A_BLK**	1	*decode*(**A_ADDR**[*M*:*M*], *D*%2)	1
A_MSB = *M*+1	**A_BLK**	*decode* (**A_ADDR**[*M*+1:*M*+1], (*D*/2)%2)	*decode* (**A_ADDR**[*M*:*M*], *D*%2)	1
A_MSB > *M*+1	**A_BLK** & *decode* (**A_ADDR**[*A_MSB*:*M*+2], *D*/4)	*decode* (**A_ADDR**[*M*+1:*M*+1], (*D*/2)%2)	*decode* (**A_ADDR**[*M*:*M*], *D*%2)	1

Note that **A_REN** is available on the top-level generated component only when there is no depth cascading (*A_MSB* < *M*). De-asserting **A_REN** will hold the previous read-data. De-asserting **A_BLK** will generate zeros on the read-data.

B_BLK, B_REN Connections

The **B_BLK** signal on the generated component is connected to the B port block select input (**B_BLK**) for each LSRAM block according to the block depth within the component and synchronized with **B_CLK**. When ECC Pipeline is enabled, the component level **B_BLK** is OR'd with pipelined version of component **B_BLK**, while the component level **B_REN** is AND'd with component level **B_BLK**.

Depth	B_BLK[2]	B_BLK[1]	B_BLK[0]	B_REN
B_MSB < *N*	**B_BLK**	1	1	**B_REN**
B_MSB = *N*	**B_BLK**	1	*decode*(**B_ADDR**[*N*:*N*], *D*%2)	1
B_MSB = *N*+1	**B_BLK**	*decode*(**B_ADDR**[*N*+1:*N*+1], (*D*/2)%2)	*decode*(**B_ADDR**[*N*:*N*], *D*%2)	1
B_MSB > *N*+1	**B_BLK** & *decode*(**B_ADDR**[*A_MSB*:*N*+2], *D*/4)	*decode*(**B_ADDR**[*N*+1:*N*+1], (*D*/2)%2)	*decode*(**B_ADDR**[*N*:*N*], *D*%2)	1

Note that **B_REN** is available on the top-level generated component only when there is no depth cascading (*B_MSB* < *N*). De-asserting **B_REN** will hold the previous read-data. De-asserting **B_BLK** will generate zeros on the read-data.

A_DIN, B_DIN Connections

The **A_DIN** and **B_DIN** bits on the generated component is partitioned into slices based on the width of the data on the respective port of each LSRAM block. Each bit of **A_DIN** is connected to all blocks in a slice at every depth and synchronized with **A_CLK**. Each bit of **B_DIN** is connected to all blocks in a slice at every depth and synchronized with **B_CLK**.

A_DOUT Logic

The **A_DOUT** bits on the generated component is partitioned into slices based on the width of the data on the A port of each LSRAM block. Each bit of read-data on the A-port from all blocks in a slice at every depth is OR'd together to generate a bit of **A_DOUT**. The **A_DOUT** bits are synchronized with **A_CLK** according to the following latency.

ECC Pipeline	ECC	A_DOUT Pipeline	A_DOUT Latency
No	No	No	0
No	No	Yes	1
No	Yes	No	0
No	Yes	Yes	1
Yes	Yes	No	1
Yes	Yes	Yes	2

B_DOUT Logic

The **B_DOUT** bits on the generated component is partitioned into slices based on the width of the data on the B port of each LSRAM block. Each bit of read-data on the B-port from all blocks in a slice at every depth is OR'd together to generate a bit of **B_DOUT**. The **B_DOUT** bits are synchronized with **B_CLK** according to the following latency.

ECC Pipeline	ECC	B_DOUT Pipeline	B_DOUT Latency
No	No	No	0
No	No	Yes	1
No	Yes	No	0
No	Yes	Yes	1
Yes	Yes	No	1
Yes	Yes	Yes	2

A_SB_CORRECT, A_DB_DETECT Logic

The **A_SB_CORRECT** and **A_DB_DETECT** outputs are synchronized with **A_CLK** according to the above **A_DOUT** latency. The **A_SB_CORRECT** flags of each LSRAM block are gated by its (**A_BLK** and **A_REN**) signals pipelined one more than the **A_DOUT** latency value and then OR'd together to generate the **A_SB_CORRECT** output of the component. Similarly, the **A_DB_DETECT** flags of each LSRAM block are gated by its (**A_BLK** and **A_REN**) signals pipelined one more than the **A_DOUT** latency value and then OR'd together to generate the **A_DB_DETECT** output of the component. As a result, both the outputs are zero whenever **A_BLK** or **A_REN** is de-asserted.

B_SB_CORRECT, B_DB_DETECT Logic

The **B_SB_CORRECT** and **B_DB_DETECT** outputs are synchronized with **B_CLK** according to the above **B_DOUT** latency. The **B_SB_CORRECT** flags of each LSRAM block are gated by its (**B_BLK** and **B_REN)** signals pipelined one more than the **B_DOUT** latency value and then OR'd together to generate the **B_SB_CORRECT** output of the component. Similarly, the **B_DB_DETECT** flags of each LSRAM block are gated by its (**B_BLK** and **B_REN**) signals pipelined one more than the **B_DOUT** latency value and then OR'd together to generate the **B_DB_DETECT** output of the component. As a result, both the outputs are zero whenever **B_BLK** or **B_REN** is de-asserted.

3 – Implementation Rules

Caveats for Dual-Port Large SRAM generation

- The core configurator only supports depth cascading up to 32 blocks.
- The software returns a configuration error for unsupported configurations.

Note

- All unused inputs must be grounded.
- ARST_N does not reset the memory contents. It resets only the pipeline registers for Read Data.
- Writing different data to the same address using both ports in Dual-Port Large SRAM is undefined and should be avoided.
- Writing to and reading from the same address is undefined and should be avoided. There is no collision prevention or detection. However, correct data is expected to be written into the memory.
- Read from both ports at the same location is allowed.

4 – RAM Content Manager

The RAM Content Manager enables you to specify the contents of your memory so that you can avoid the simulation cycles required for initializing the memory, which reduces simulation runtime.

The RAM core generator takes away much of the complexity required in the generation of large memory that utilize one or more RAM blocks on the device. The configurator uses one or more memory blocks to generate a RAM matching your configuration. In addition, it also creates the surrounding cascading logic.

The configurator cascades RAM blocks in three different ways.

- Cascaded deep (e.g. 2 blocks of 1024x18 to create a 2048x18)
- Cascaded wide (e.g. 2 blocks of 1024x18 to create a 1024x36)
- Cascaded wide and deep (e.g. 4 blocks of 1024x18 to create a 2048x36, in a 2 blocks width-wise by 2 blocks depth-wise configuration)

Specify memory content in terms of your total memory size. The configurator must partition your memory file appropriately such that the right content goes to the right block RAM when multiple blocks are cascaded.

Supported Formats

The Microsemi implementation of these formats interprets data sets in bytes. This means that if the memory width is 7 bits, every 8th bit in the data set is ignored. Or, if the data width is 9, two bytes are assigned to each memory address and the upper 7 bits of each 2-byte pair are ignored.

The following examples illustrate how the data is interpreted for various word sizes:

For the given data: FF 11 EE 22 DD 33 CC 44 BB 55 (where 55 is the MSB and FF is the LSB)

For 32-bit word size:

```
0x22EE11FF (address 0)
0x44CC33DD (address 1)
0x000055BB (address 2)
```

For 16-bit word size:

```
0x11FF (address 0)
0x22EE (address 1)
0x33DD (address 2)
0x44CC (address 3)
0x55BB (address 4)
```

For 8-bit word size:

```
0xFF (address 0)
0x11 (address 1)
0xEE (address 2)
0x22 (address 3)
0xDD (address 4)
0x33 (address 5)
0xCC (address 6)
0x44 (address 7)
0xBB (address 8)
0x55 (address 9)
```

For 9-bit word size:

```
0x11FF -> 0x01FF (address 0)
0x22EE -> 0x00EE (address 1)
0x33DD -> 0x01DD (address 2)
0x44CC -> 0x00CC (address 3)
0x55BB -> 0x01BB (address 4)
```

Notice that for 9-bit, that the upper 7-bits of the 2-bytes are ignored.

INTEL-HEX

Industry standard file. Extensions are HEX and IHX. For example, file2.hex or file3.ihx.

A standard format created by Intel. Memory contents are stored in ASCII files using hexadecimal characters. Each file contains a series of records (lines of text) delimited by new line, '\n', characters and each record starts with a ':' character. For more information regarding this format, refer to the Intel-Hex Record Format Specification document available on the web (search Intel Hexadecimal Object File for several examples).

The Intel Hex Record is composed of five fields and arranged as follows:

```
:llaaaatt[dd...]cc
```

Where:

- : is the start code of every Intel Hex record
- ll is the byte count of the data field
- aaaa is the 16-bit address of the beginning of the memory position for the data. Address is big endian.
- tt is record type, defines the data field:
  - 00 data record
  - 01 end of file record
  - 02 extended segment address record
  - 03 start segment address record (ignored by Microsemi SoC tools)
  - 04 extended linear address record
  - 05 start linear address record (ignored by Microsemi SoC tools)
- [dd...] is a sequence of n bytes of the data; n is equivalent to what was specified in the ll field
- cc is a checksum of count, address, and data

Example Intel Hex Record:

```
:0300300002337A1E
```

MOTOROLA S-record

Industry standard file. File extension is S, such as file4.s

This format uses ASCII files, hex characters, and records to specify memory content in much the same way that Intel-Hex does. Refer to the Motorola S-record description document for more information on this format (search Motorola S-record description for several examples). The RAM Content Manager uses only the S1 through S3 record types; the others are ignored.

The major difference between Intel-Hex and Motorola S-record is the record formats, and some extra error checking features that are incorporated into Motorola S.

In both formats, memory content is specified by providing a starting address and a data set. The upper bits of the data set are loaded into the starting address and leftovers overflow into the adjacent addresses until the entire data set has been used.

The Motorola S-record is composed of 6 fields and arranged as follows:

```
Stllaaaa[dd...]cc
```

Where:

- S is the start code of every Motorola S-record
- t is record type, defines the data field
- ll is the byte count of the data field
- aaaa is a 16-bit address of the beginning of the memory position for the data. Address is big endian.

- [dd...] is a sequence of n bytes of the data; n is equivalent to what was specified in the ll field
- cc is the checksum of count, address, and data

Example Motorola S-Record:

```
S10a0000112233445566778899FFFA
```

RAM Content Manager Functionality

To open the RAM Content Manager, after specifying your RAM configurations (set your Read and Write Depth and Width), select the **Initialize RAM for Simulation** checkbox, and then click **Customize RAM Content**. The RAM Content Manager appears (Figure 4-1).

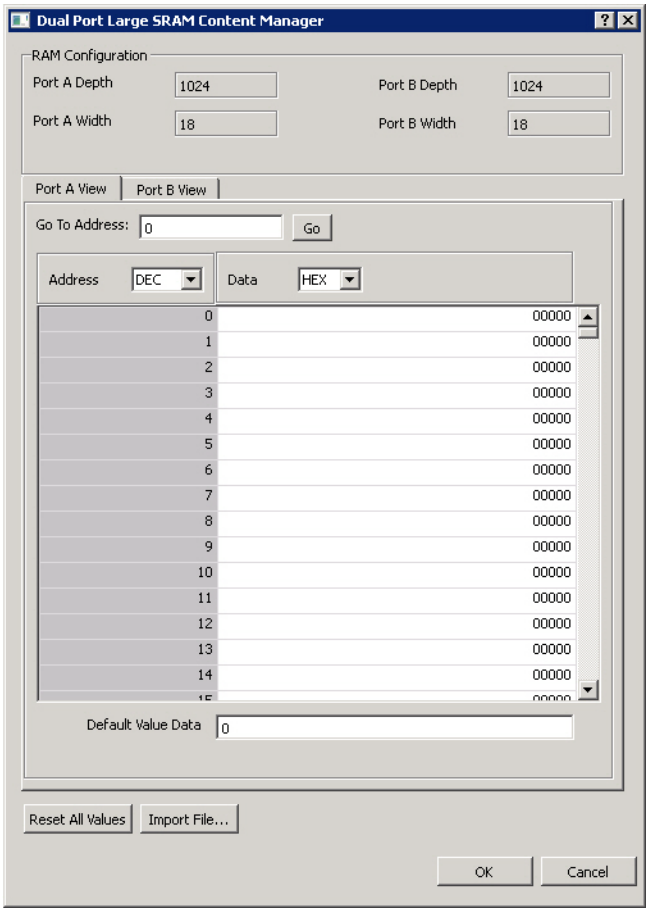

***Figure 4-1* • Customize RAM Content for Simulation**

RAM Configuration

Write Depth and Write Width - As specified in the RAM core generator dialog box (not editable).

Read Depth and Read Width - As specified in the RAM core generator dialog box (not editable).

Write Port View / Read Port View

Go To Address - Enables you to go to a specific address in the manager. Each memory block has many addresses; it is often difficult to scroll through and find a specific one. This task is simplified by enabling you to type in a specific address. The number display format (Hex, Bin, Dec) is controlled by the value you set in the drop-down menu above the Address column.

Address - The Address column lists the address of a memory location. The drop-down menu specifies the number format for your address list (hexadecimal, binary, or decimal).

Data - Enables you to control the data format and data value in the manager. Click the value to change it.

Note that the dialogs show all data with the MSB down to LSB. For example, if the row showed 0xAABB for a 16-bit word size, the AA would the MSB and BB would be LSB.

Default Data Value - The value given to memory addresses that have not been explicitly initialized (by importing content or editing manually). When changed, all default values in the manager are updated to match the new value. The number display format (Hex, Bin, Dec) is controlled by the value you set in the drop-down menu above the Data column.

Reset All Values - Resets the Data values.

Import File - Opens the Import Memory Content dialog box; enables you to select a memory content file (Intel-Hex) to load. File extensions are set to *.hex for Intel-Hex files during import.

OK - Closes the manager and saves all the changes made to the memory and its contents.

Cancel - Closes the manager, cancels all your changes in this instance of the manager, and returns the memory back to the state it held before the manager was opened.

MEMFILE (RAM Content Manager output file)

Transfer of RAM data (from the RAM Content Manager) to test equipment is accomplished via MEM files. The contents of your RAM is first organized into the logical layer and then reorganized to fit the hardware layer. Then it is stored in MEM files that are read by other systems and used for testing.

The MEM files are named according to the logical structure of RAM elements created by the configurator. In this scheme the highest order RAM blocks are named CORE_R0C0.mem, where "R" stands for row and "C" stands for column. For multiple RAM blocks, the naming continues with CORE_R0C1, CORE_R0C2, CORE_R1C0, etc.

The data intended for the RAM is stored as ASCII 1s and 0s within the file. Each memory address occupies one line. Words from logical layer blocks are concatenated or split in order to make them fit efficiently within the hardware blocks. If the logical layer width is less than the hardware layer, two or more logical layer words are concatenated to form one hardware layer word. In this case, the lowest bits of the hardware word are made up of the lower address data bits from the logical layer. If the logical layer width is more than the hardware layer, the words are split, placing the lower bits in lower addresses.

If the logical layer words do not fit cleanly into the hardware layer words, the most significant bit of the hardware layer words is not used and defaulted to zero. This is also done when the logical layer width is 1 in order to avoid having left over memory at the end of the hardware block.

5 – Port Description

Table 5-1 lists the Dual-Port Large SRAM signals in the generated macro.

***Table 5-1* • Dual-Port Large SRAM Signals**

Port	Direction	Polarity	Description
CLK	In	Rising edge	Single clock signal that drives all three ports with the same clock
A_DIN[]	In		Port A Write data
A_ADDR[]	In		Port A Read address
A_BLK	In	Active high	Port A Enable
A_CLK	In	Rising edge	Port A clock
A_WEN	In		Port A signal to switch between Read and Write modes: Low = Read; High = Write
A_REN	In		Port A Read Data Enable (exposed only if there is no depth cascading)
A_WBYTE_EN[]	In	Active High	Port A Write Byte Enables (per byte)
A_DOUT[]	Out		Port A Read data
A_DOUT_EN	In	Active High	Port A Read data register Enable
A_DOUT_SRST_N	In	Active Low	Port A Read data register Synchronous reset
A_SB_CORRECT	Out	Active High	Port A single-bit correct flag
A_DB_DETECT	Out	Active High	Port A double-bit detect flag
B_DIN[]	In		Port B Write data
B_ADDR[]	In		Port B address
B_BLK	In	Active High	Port B Enable
B_CLK	In	Rising edge	Port B clock
B_WEN	In		Port signal to switch between Read and Write modes: Low = Read; High = Write
B_REN	In		Port B Read Data Enable (exposed only if there is no depth cascading)
B_WBYTE_EN[]	In	Active High	Port B Write Byte Enables (per byte)
B_DOUT[]	Out		Port B Read data
B_DOUT_EN	In	Active High	Port B Read data register Enable
B_DOUT_SRST_N	In	Active Low	Port B Read data register Synchronous reset
B_SB_CORRECT	Out	Active High	Port B single-bit correct flag
B_DB_DETECT	Out	Active High	Port B double-bit detect flag
ARST_N	In	Active Low	Port A and B Read data register Asynchronous reset

6 – Parameters

Table 6-1 lists the Dual-Port Large SRAM parameters in the generated macro.

Table 6-1 • **Dual-Port Large SRAM Parameters**

GENFILE Parameter	Configurator Parameter	Valid Range	Default	Description
DESIGN				Name of the generated macro
FAM		RTG4		Target family
OUTFORMAT		Verilog, VHDL		Netlist format
LPMTYPE		LPM_RAM		Macro category
DEVICE		12000	12000	Target device: RT4G150
PTYPE	PTYPE	2	2	2: Dual-port
INIT_RAM	INIT_RAM	F, T	F	Initialize RAM for simulation
CASCADE	CASCADE	0, 1	0	0: Cascading for WIDTH or Speed 1: Cascading for DEPTH or Power
CLKS	CLKS	1, 2	1	1: Single Read/Write Clock 2 : Independent Read and Write Clocks
WCLK_EDGE	CLK_EDGE	CLS=1 RISE, FALL	RISE	RISE: Rising edge Single clock FALL: Falling edge Single clock
WWIDTH	A_WIDTH	1-3762	18	Port A data width
WDEPTH	A_DEPTH	1-65536	1024	Port A address depth
RWIDTH	B_WIDTH	1-3762	18	Port B data width
RDEPTH	B_DEPTH	1- 65536	1024	Port B address depth
WE_POLARITY	A_BLK_POLARITY	0, 1, 2	2	0: Active-low Port A enable 1: Active-high Port A enable 2: Port A enable tied off to be always active
WCLK_EDGE	A_CLK_EDGE	CLKS=2 RISE, FALL	RISE	RISE: Rising edge Port A clock FALL: Falling edge Port A clock
PMODE1	A_PMODE	0, 1	0	0: Bypass Port A read data register 1: Pipeline Port A read data
WMODE1	A_WMODE	0	0	0: Hold Port A read data
A_DOUT_EN_POLARITY	A_DOUT_EN_POLARITY	PMODE1=1 0, 1, 2	2	0: Active-low Port A read data register enable 1: Active-high Port A read data register enable 2: Port A read data register enable tied off to be always active

***Table 6-1* • Dual-Port Large SRAM Parameters (continued)**

GENFILE Parameter	Configurator Parameter	Valid Range	Default	Description
A_DOUT_SRST_POLARITY	A_DOUT_SRST_POLARITY	PMODE1=1 0, 1, 2	2	0: Active-low Port A read data register Sync-reset 1: Active-high Port A read data register Sync-reset 2: Port A read data register Sync-reset tied off to be always inactive
A_REN_POLARITY	A_REN_POLARITY	WMODE1=0 0, 1, 2	2	0: Active-low Port A read data enable 1: Active-high Port A read data enable 2: Port A read data enable tied off to be always active
RE_POLARITY	B_BLK_POLARITY	0, 1, 2	2	0: Active-low Port B enable 1: Active-high Port B enable 2: Port B enable tied off to be always active
RCLK_EDGE	B_CLK_EDGE	CLKS=2 RISE, FALL	RISE	RISE: Rising edge Port B clock FALL: Falling edge Port B clock
PMODE2	B_PMODE	0, 1	0	0: Bypass Port B read data register 1: Pipeline Port B read data
WMODE2	B_WMODE	0	0	0: Hold Port B read data
B_DOUT_EN_POLARITY	B_DOUT_EN_POLARITY	PMODE2=1 0, 1, 2	2	0: Active-low Port B read data register enable 1: Active-high Port B read data register enable 2: Port B read data register enable tied off to be always active
B_DOUT_SRST_POLARITY	B_DOUT_SRST_POLARITY	PMODE2=1 0, 1, 2	2	0: Active-low Port B read data register Sync-reset 1: Active-high Port B read data register Sync-reset 2: Port B read data register Sync-reset tied off to be always inactive
B_REN_POLARITY	B_REN_POLARITY	WMODE2=0 0, 1, 2	2	0: Active-low Port B read data enable 1: Active-high Port B read data enable 2: Port B read data enable tied off to be always active
RESET_POLARITY	RESET_POLARITY	PMODE1=1 or PMODE2=1 0, 1, 2	2	0: Active-low Read data register Async-reset 1: Active-high Read data register Async-reset 2: Read data register Async-reset tied off to be always inactive
BYTEENABLES	BYTEENABLES	0, 1	0	0: Don't generate A_WBYTE_EN or B_WBYTE_EN 1: Generate A_WBYTE_EN and B_WBYTE_EN
ECC	ECC	0, 1, 2	0	0: ECC Disabled 1: ECC Pipelined 2: ECC Non-pipelined
CLOCK_PN	CLOCK_PN	CLKS=1	CLK	Single clock Port name
DATAA_IN_PN	DATAA_IN_PN		A_DIN	Port A write data Port name
ADDRESSA_PN	ADDRESSA_PN		A_ADDR	Port A address Port name
BLKA_PN	BLKA_PN	A_BLK_POLARITY<2	A_BLK	Port A enable Port name

***Table 6-1* • Dual-Port Large SRAM Parameters (continued)**

GENFILE Parameter	Configurator Parameter	Valid Range	Default	Description
CLKA_PN	CLKA_PN	CLKS=2	A_CLK	Port A clock Port name
RWA_PN	RWA_PN		A_WEN	Port A Write enable Port name
DATAA_OUT_PN	DATAA_OUT_PN		A_DOUT	Port A read data Port name
A_DOUT_EN_PN	A_DOUT_EN_PN	PMODE1=1	A_DOUT_EN	Port A read data register enable Port name
A_DOUT_SRST_PN	A_DOUT_SRST_PN	PMODE1=1	A_DOUT_SRST_N	Port A read data register Sync-reset Port name
A_REN_PN	A_REN_PN	A_REN_POLARITY< 2	A_REN	Port A Read data enable Port name
DATAB_IN_PN	DATAB_IN_PN		B_DIN	Port B write data Port name
ADDRESSB_PN	ADDRESSB_PN		B_ADDR	Port B address Port name
BLKB_PN	BLKB_PN	B_BLK_POLARITY<2	B_BLK	Port B enable Port name
CLKB_PN	CLKB_PN	CLKS=2	B_CLK	Port B clock Port name
RWB_PN	RWB_PN		B_WEN	Port B Write enable Port name
DATAB_OUT_PN	DATAB_OUT_PN		B_DOUT	Port B read data Port name
B_DOUT_EN_PN	B_DOUT_EN_PN	PMODE2=1	B_DOUT_EN	Port B read data register enable Port name
B_DOUT_SRST_PN	B_DOUT_SRST_PN	PMODE2=1	B_DOUT_SRST_N	Port B read data register Sync-reset Port name
B_REN_PN	B_REN_PN	B_REN_POLARITY< 2	B_REN	Port B Read data enable Port name
RESET_PN	RESET_PN	PMODE1=1 or PMODE2=1	ARST_N	Read data registers Async-reset Port name