RTG4 FPGA
Two-Port Large SRAM Configuration

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.

5-02-00797-2/08.18

Table of Contents

Introduction

A Two-Port Large SRAM enables write access on one port and read access on the other port (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 Read and Write aspect ratios.

Two-Port Large SRAM is synchronous for read and write operations, setting up the addresses as well as writing and reading the data. The memory write and read operations are triggered at the rising edge of the clock.

An optional pipeline register is available at the read data port to improve the clock-to-out delay.

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

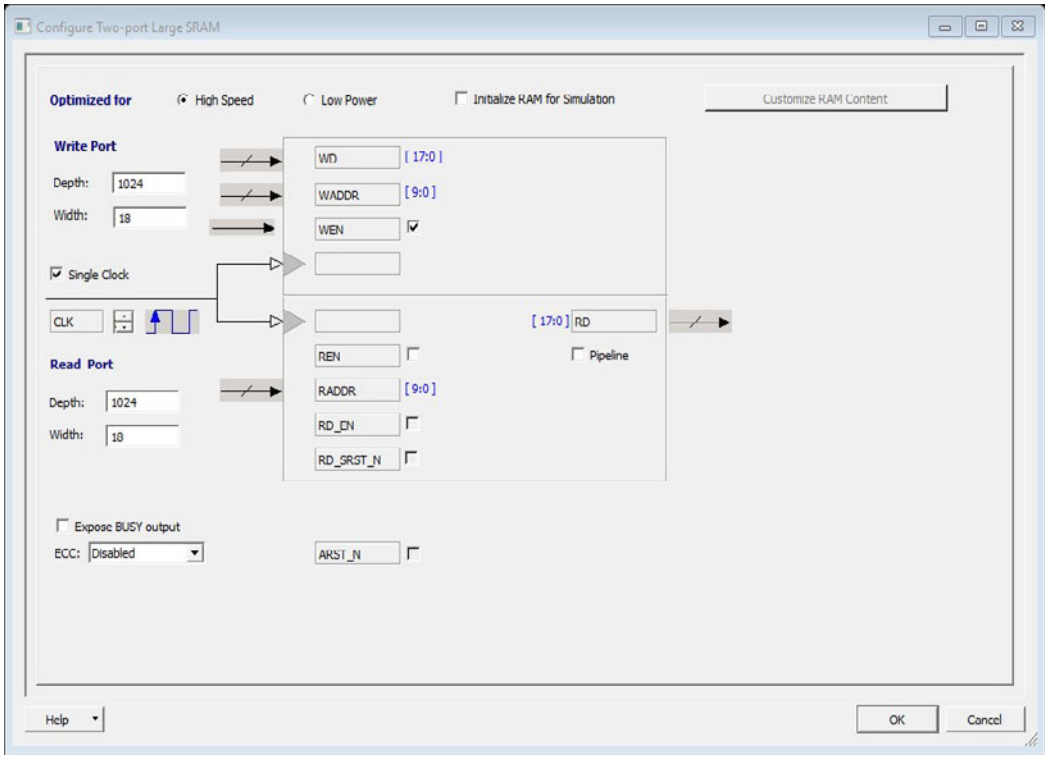

***Figure 1* • Two-Port Large SRAM Configurator**

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.

Write Depth/Width and Read Depth/Width

The depth range for each port is 1-65536. The width range for each port is 1-7524.

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

Single Clock (CLK) or Independent Write and Read Clocks (WCLK, RCLK)

The default configuration for Two-Port Large SRAM is a Single clock (CLK) to drive WCLK and RCLK with the same clock. Uncheck the Single clock checkbox to drive independent clocks (one each for Write and Read).

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

Write Enable (WEN)

Asserting WEN writes the data WD into the RAM at the address WADDR on the next rising edge of WCLK. Un-checking the WEN option ties the signal to the active state and removes it from the generated macro; click the signal arrow to toggle its polarity.

Read Enable (REN)

De-asserting REN holds the previous Read data (RD).

Asserting the REN reads the RAM at the read address RADDR onto the input of the RD register on the next rising edge of RCLK.

The default configuration for REN is unchecked, which ties the signal to the active state and removes it from the generated macro. Click the checkbox to insert that signal on the generated macro; click the signal arrow to toggle its polarity.

Pipeline for Read Data Output

Click the Pipeline checkbox to enable pipelining for Read data (RD). This is a static selection and cannot be changed dynamically by driving it with a signal.

Turning off pipelining of Read data also disables the configuration options of the RD_EN, RD_SRST_N and ARST_N signals.

Register Enable (RD_EN)

The pipeline register for RD has an active high, enable input. The default configuration is to tie this signal to the active state and remove it from the generated macro. Click the signal's checkbox to insert that signal on the generated macro; click the signal arrow to toggle its polarity.

Synchronous Reset (RD_SRST_N)

The pipeline register for RD has an active low, synchronous reset input. The default configuration is to tie this signal to the inactive state and remove it from the generated macro. Click the signal's checkbox to insert that signal on the generated macro; click the signal arrow to toggle its polarity.

Asynchronous Reset (ARST_N)

The pipeline register for RD has 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 signal's checkbox to insert that signal on the generated macro; click the signal arrow to toggle its polarity.

Expose Busy Output

Check the Expose BUSY output checkbox to enable it. When enabled, the SmartDebug BUSY signal is promoted as a top-level port.

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 36 bits, 18 bits or 9 bits width. Alternatively, both ports could be configured to 12 bits width.

When ECC is enabled (Pipelined or Non-Pipelined), both ports have word widths equal to 36 bits or 18 bits.

RD Register Truth Table

Table 1-1 describes the functionality of the control signals on the RD register.

***Table 1-1* • RD Register Truth Table**

ARST_N	RCLK	RD_EN	RD_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 – Implementation Rules

Caveats for Two-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 resgister for RD.
- Writing to and reading from the same address is undefined and should be avoided. There is no collision prevention or detection.

3 – 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 configuration (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 3-1).

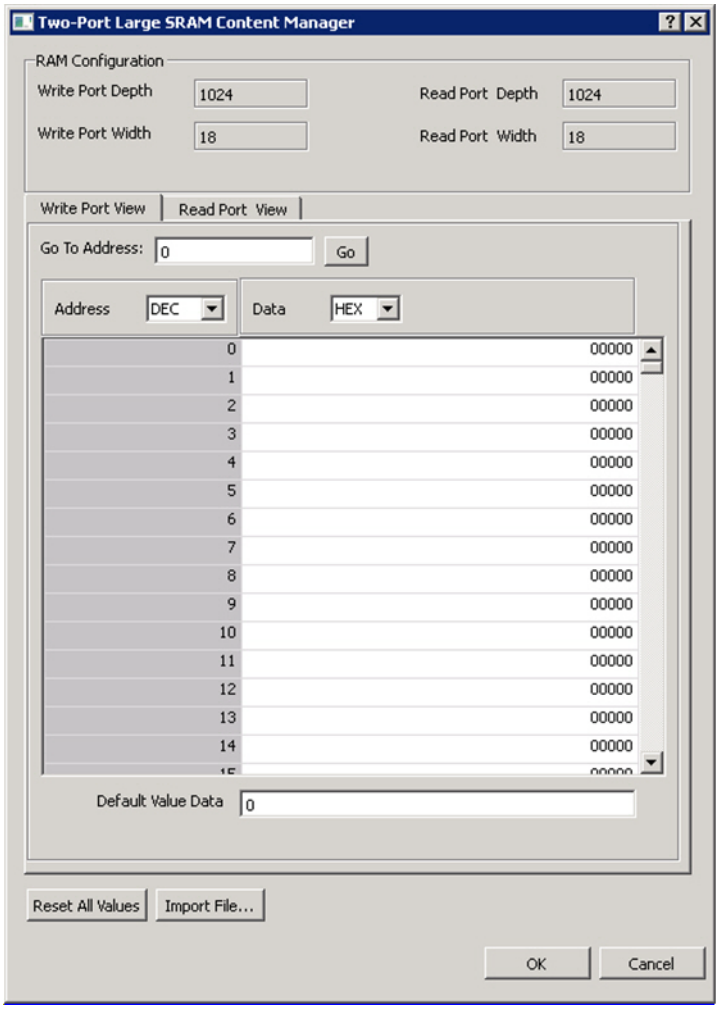

***Figure 3-1* • Customize RAM Content for Simulation**

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 from File - Opens the Import Memory Content dialog box; enables you to select a memory content file (Intel-Hex) to load. Intel-Hex file extensions are set to *.hex 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 leftover memory at the end of the hardware block.

4 – Port Description

Table 4-1 lists the Two-Port Large SRAM signals in the generated macro.

***Table 4-1* • Two-Port Large SRAM Signals**

Port	Direction	Default Polarity	Description
CLK	In	Rising Edge	Single clock to drive both WCLK and RCLK
WD[]	In		Write data
WADDR[]	In		Write address
WEN	In	Active high	Write port enable
WCLK	In	Rising edge	Write clock
RCLK	In	Rising edge	Read clock
REN	In	Active high	Read data enable
RADDR[]	In		Read address
RD[]	Out		Read data
RD_EN	In	Active high	Read data register enable
RD_SRST_N	In	Active low	Read data register Synchronous reset
ARST_N	In	Active low	Read data register Asynchronous reset
SB_CORRECT	Out	Active High	Single-bit correct flag
DB_DETECT	Out	Active High	Double-bit detect flag
BUSY	Out	Active High	Busy signal

5 – Parameters

Table 5-1 lists the Micro SRAM parameters in the generated macro.

***Table 5-1* • Two-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	1, 2	1	1: Two-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	CLKS=1 RISE, FALL	RISE	RISE: Rising edge Single clock FALL: Falling edge Single clock
WWIDTH	WWIDTH	1-7524	18	Write data width
WDEPTH	WDEPTH	1-65536	1024	Write address depth
RWIDTH	RWIDTH	1-7524	18	Read data output width
RDEPTH	RDEPTH	1- 65536	1024	Read address depth
WE_POLARITY	WE_POLARITY	0, 1, 2	1	0: Active-low Write port enable 1: Active-high Write port enable 2: Write port enable tied off to be always active
WCLK_EDGE	WCLK_EDGE	CLKS=2 RISE, FALL	RISE	RISE: Rising edge Write clock FALL: Falling edge Write clock
RCLK_EDGE	RCLK_EDGE	CLKS=2 RISE, FALL	RISE	RISE: Rising edge Read clock FALL: Falling edge Read clock
RE_POLARITY	RE_POLARITY	0, 1, 2	2	0: Active-low Read port enable 1: Active-high Read port enable 2: Read port enable tied off to be always active
PMODE2	RPMODE	0, 1	0	0: Bypass Read data register 1: Pipeline Read data
A_DOUT_EN_POLARITY	A_DOUT_EN_POLARITY	PMODE2=1 0, 1, 2	2	0: Active-low Read data register enable 1: Active-high Read data register enable 2: Read data register enable tied off to be always active

***Table 5-1* • Two-Port Large SRAM Parameters (continued)**

GENFILE Parameter	Configurator Parameter	Valid Range	Default	Description
A_DOUT_SRST_POLARITY	A_DOUT_SRST_POLARITY	PMODE2=1 0, 1, 2	2	0: Active-low Read data register Sync-reset 1: Active-high Read data register Sync-reset 2: Read data register Sync-reset tied off to be always inactive
RESET_POLARITY	ARST_N_POLARITY	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
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
DATA_IN_PN	DATA_IN_PN		WD	Write data Port name
WADDRESS_PN	WADDRESS_PN		WADDR	Write address Port name
WE_PN	WE_PN	WE_POLARITY<2	WEN	Write port enable Port name
WCLOCK_PN	WCLOCK_PN	CLKS=2	WCLK	Write clock Port name
RCLOCK_PN	RCLOCK_PN	CLKS=2	RCLK	Read clock Port name
RE_PN	RE_PN	RE_POLARITY<2	REN	Read port enable Port name
RADDRESS_PN	RADDRESS_PN		RADDR	Read address Port name
DATA_OUT_PN	DATA_OUT_PN		RD	Read data Port name
A_DOUT_EN_PN	A_DOUT_EN_PN	PMODE2=1	RD_EN	Read data register enable Port name
A_DOUT_SRST_PN	A_DOUT_SRST_PN	PMODE2=1	RD_SRST_N	Read data register Sync-reset Port name
RESET_PN	RESET_PN	PMODE2=1	ARST_N	Read data register Async-reset Port name
BUSY_FLAG	BUSY_FLAG	0, 1	0	0: Don't generate BUSY 1: Generate BUSY