Memory Usage Guide for iCE40 Devices

Technical Note

FPGA-TN-02002-1.7
September 2020
Disclaimers
Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer’s responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice’s product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.
Contents

Acronyms in This Document ..................................................................................................................... 5
1. Introduction ........................................................................................................................................ 6
2. Memories in iCE40 Devices .................................................................................................................. 6
3. iCE40 sysMEM Embedded Block RAM ............................................................................................... 7
   3.1. Signals .......................................................................................................................................... 8
   3.2. Timing Diagram .............................................................................................................................. 9
   3.3. Memory Initialization ..................................................................................................................... 10
   3.4. Write Operations ........................................................................................................................... 10
   3.5. Read Operations ............................................................................................................................. 10
   3.6. EBR Considerations ....................................................................................................................... 10
      3.6.1. Read Data Register Undefined Immediately After Configuration ...................................... 10
      3.6.2. Pre-loading EBR Data ............................................................................................................. 10
      3.6.3. EBR Contents Preserved During Configuration ................................................................... 11
      3.6.4. Low-Power Setting ................................................................................................................. 11
4. iCE40 sysMEM Embedded Block RAM Memory Primitives ................................................................. 11
   4.1. SB_RAM256x16 ............................................................................................................................... 13
   4.2. SB_RAM512x8 ............................................................................................................................... 15
   4.3. SB_RAM1024x4 ............................................................................................................................. 17
   4.4. SB_RAM2048x2 ............................................................................................................................. 19
   4.5. SB_RAM40_4K ............................................................................................................................... 21
5. EBR Utilization Summary in iCEcube2 Design Software ...................................................................... 23
Appendix A. Standard HDL Code References .......................................................................................... 24
   A.1. Single-Port RAM ........................................................................................................................... 24
   A.2. Dual Port RAM ............................................................................................................................... 25
References .................................................................................................................................................... 27
Technical Support Assistance .................................................................................................................... 28
Revision History ......................................................................................................................................... 29
Figures

Figure 2.1. Typical Layout of an iCE40 Device..................................................................................6
Figure 3.1. sysMEM Embedded Block RAM ...................................................................................7
Figure 3.2. EBR Module Timing Diagram.........................................................................................9
Figure 4.1. SB_RAM256x16 Primitive ..........................................................................................13
Figure 4.2. SB_RAM512x8 Primitive ...............................................................................................15
Figure 4.3. SB_RAM1024x4 Primitive .............................................................................................17
Figure 4.4. SB_RAM2048x2............................................................................................................19
Figure 4.5. SB_RAM40_4K...............................................................................................................21
Figure 5.1. iCEcube2 Design Software Report File ........................................................................23

Tables

Table 3.1. EBR Signal Descriptions .................................................................................................8
Table 4.1. EBR Configurations and Primitive Names ......................................................................11
Table 4.2. Naming Convention for RAM Primitives ....................................................................11
Table 4.3. Address and Data Mapping for the Initialize RAM Configuration (Write Data) .........12
Table 4.4. Address and Data Mapping for the Initialize RAM Configuration (Read Data) ........12
Table 4.5. SB_RAM256x16 Based Primitives ................................................................................14
Table 4.6. SB_RAM512x8 Based Primitives ..................................................................................16
Table 4.7. SB_RAM1024x4 Based Primitives ................................................................................18
Table 4.8. SB_RAM2048x2 Based Primitives ................................................................................20
Table 4.9. SB_RAM40_4K Naming Convention Rules ..................................................................21
Table 4.10. SB_RAM40_4K Signal Descriptions ..........................................................................21
Table 4.11. SB_RAM40_4K Primitive Parameter Descriptions ......................................................22
# Acronyms in This Document

A list of acronyms used in this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBR</td>
<td>Embedded Block RAM</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>RAM</td>
<td>Random-Access Memory</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
</tr>
<tr>
<td>VHDL</td>
<td>Verilog Hardware Description Language</td>
</tr>
</tbody>
</table>
1. Introduction

This technical note discusses memory usage for the iCE40™ device family (iCE40 LP/HX, iCE40 LM, iCE40 Ultra™, iCE40 UltraLite™, iCE40 UltraPlus™). It is intended to be used as a guide to the high-speed synchronous RAM Blocks and the iCE40 sysMEM™ Embedded Block RAM (EBR). The EBR is the embedded block RAM of the device, each 4 Kbit in size. The iCE40 device architecture provides resources for memory-intensive applications. Single-Port RAM, Dual-Port RAM and FIFO can be constructed using the EBRs. The EBRs can be utilized by instantiating software primitives as described later in this document. Apart from primitive instantiation, the iCEcube2™ design software infers generic codes as EBRs.

2. Memories in iCE40 Devices

iCE40 devices contain an array of EBRs. Figure 2.1 shows the placement of EBRs in a typical iCE40 device (does not represent true numbers of design elements).

![Figure 2.1. Typical Layout of an iCE40 Device](image-url)
3. iCE40 sysMEM Embedded Block RAM

Each iCE40 device includes multiple high-speed synchronous EBRs, each 4Kbit in size. A single iCE40 device integrates between eight and 32 such blocks. Each EBR is a 256-word deep by 16-bit wide, two-port register file, as illustrated in Figure 3.1. The input and output connections to and from an EBR feed into the programmable interconnect resources.

![Figure 3.1. sysMEM Embedded Block RAM](image)

Using programmable logic resources, an EBR implements a variety of logic functions, each with configurable input and output data widths.

- Random-access memory (RAM)
  - Single-port RAM with a common address, enable, and clock control lines
  - Two-port RAM with separate read and write control lines, address inputs, and enable
- Register file and scratchpad RAM
- First-In, First-Out (FIFO) memory for data buffering applications
- 256-deep by 16-wide ROM with registered outputs; contents loaded during configuration
- Counters, sequencers

As shown in Figure 3.1, an EBR has separate write and read ports, each with independent control signals. Table 3.1 lists the signals for both ports. Additionally, the write port has an active-low bit-line write-enable control; optionally mask write operations on individual bits. By default, input and output data is 16 bits wide, although the data width is configurable using programmable logic and, if needed, multiple EBRs.

The WCLK and RCLK inputs optionally connect to one of the following clock sources:

- The output from any one of the eight Global Buffers, or
- A connection from the general-purpose interconnect fabric
3.1. Signals

Table 3.1 lists the signal names, direction, and function of each connection to the EBR block.

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDATA[15:0]</td>
<td>Input</td>
<td>Write Data input.</td>
</tr>
<tr>
<td>MASK[15:0]</td>
<td>Input</td>
<td>Masks write operations for individual data bit-lines. 0 = write bit; 1 = don’t write bit</td>
</tr>
<tr>
<td>WADDR[7:0]</td>
<td>Input</td>
<td>Write Address input. Selects one of 256 possible RAM locations.</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>Write Enable input.</td>
</tr>
<tr>
<td>WCLKE</td>
<td>Input</td>
<td>Write Clock Enable input.</td>
</tr>
<tr>
<td>WCLK</td>
<td>Input</td>
<td>Write Clock input. Default rising-edge, but with falling-edge option.</td>
</tr>
<tr>
<td>RDATA[15:0]</td>
<td>Output</td>
<td>Read Data output.</td>
</tr>
<tr>
<td>RADDR[7:0]</td>
<td>Input</td>
<td>Read Address input. Selects one of 256 possible RAM locations.</td>
</tr>
<tr>
<td>RE*</td>
<td>Input</td>
<td>Read Enable input. Only available for SB_RAM256x16 configurations.</td>
</tr>
<tr>
<td>RCLKE</td>
<td>Input</td>
<td>Read Clock Enable input.</td>
</tr>
<tr>
<td>RCLK</td>
<td>Input</td>
<td>Read Clock input. Default rising-edge, but with falling-edge option.</td>
</tr>
</tbody>
</table>

*Note: Read Enable (RE) is available only for SB_RAM256x16 configuration. For other configurations (x2/ x4/ x8), RDATA output of SB_RAM40_4K can change, even if RE is active low. To use the RE functionality for other configurations, you can gate the Read Address RADDR with the RE signal when generating the ADDR signals.
3.2. Timing Diagram

Figure 3.2 shows the timing diagram for the EBR memory module.

Figure 3.2. EBR Module Timing Diagram

Note: Internal timing values are considered in the iCEcube2 software’s place and route. WE and WCLKE have to be valid for the address to be clocked in at the clock edge.
3.3. Memory Initialization

sysMEM memories can be initialized as needed. The initialization can be achieved through HDL (Verilog or VHDL) by specifying the initial values or through an initialization file (.mem file).

Refer to the iCEcube2 User Guide (under the Help menu) for more information on initializing memories. The Initializing Inferred RAM section covers the process of initializing memory by providing initial values or using mem file. The Memory Initializer section provides the DOS commands that can be used to initialize various memories using .mem files.

3.4. Write Operations

By default, all EBR write operations are synchronized to the rising edge of WCLK although the clock is invertible as shown in Figure 3.1. When the WCLKE signal is low, the clock to the EBR block is disabled, keeping the EBR in its lowest power mode.

To write data into the EBR block, perform the following operations:
- Supply a valid address on the WADDR[7:0] address input port
- Supply valid data on the WDATA[15:0] data input port
- To write or mask selected data bits, set the associated MASK input port accordingly. For example, write operations on data bit D[i] are controlled by the associated MASK[i] input.
  - MASK[i] = 0: Write operations are enabled for data line WDATA[i]
  - MASK[i] = 1: Mask write operations are disabled for data line WDATA[i]
- Enable the EBR write port (WE = 1)
- Enable the EBR write clock (WCLKE = 1)
- Apply a rising clock edge on WCLK (assuming that the clock is not inverted)

3.5. Read Operations

By default, all EBR read operations are synchronized to the rising edge of RCLK although the clock is invertible as shown in Figure 3.1.

To read data from the EBR block, perform the following operations:
- Supply a valid address on the RADDR[7:0] address input port
- Enable the EBR read port (RE = 1)
- Enable the EBR read clock (RCLKE = 1)
- Apply a rising clock edge on RCLK

After the clock edge, the EBR contents located at the specified address (RADDR) appear on the RDATA output port

3.6. EBR Considerations

3.6.1. Read Data Register Undefined Immediately After Configuration

Unlike the flip-flops in the Programmable Logic Blocks and Programmable I/O pins, the RDATA port is not automatically reset after configuration. Consequently, immediately following configuration and before the first valid Read Data operation, the initial RDATA read value is undefined.

3.6.2. Pre-loading EBR Data

The data contents for an EBR block can be optionally pre-loaded during iCE40 configuration. If not pre-loaded during configuration, then the EBR contents must be initialized by the iCE40 application before the EBR contents are valid. EBR initialization data can be done in the RTL code. Pre-loading the EBR data in the configuration bitstream increases the size of the configuration image accordingly.
3.6.3. EBR Contents Preserved During Configuration

EBR contents are preserved (write protected) during configuration, assuming that voltage supplies are maintained throughout. Consequently, data can be passed between multiple iCE40 configurations by leaving it in an EBR block and then skipping pre-loading during the subsequent reconfiguration.

3.6.4. Low-Power Setting

To place an EBR block in its lowest power mode, keep WCLKE = 0 and RCLKE = 0. In other words, when not actively using an EBR block, disable the clock inputs.

4. iCE40 sysMEM Embedded Block RAM Memory Primitives

This section lists the iCE40 sysMEM EBR software primitives that can be instantiated in the RTL. Different EBRs are used in different configurations. Each EBR has separate write and read ports, each with independent control signals. Each EBR can be configured into a RAM block of size 256x16, 512x8, 1024x4 or 2048x2. The data contents of the EBR can optionally be pre-loaded during iCE40 device configuration by specifying the initialization data in the primitive instantiation. Table 4.3 and Table 4.4 shows how the address and data connection are mapped according to RAM configuration.

Table 4.1 lists the supported dual port synchronous RAM configurations, each of 4 Kbits in size. The RAM blocks can be directly instantiated in the top module and taken through the iCEcube2 software flow.

### Table 4.1. EBR Configurations and Primitive Names

<table>
<thead>
<tr>
<th>Block RAM Configuration</th>
<th>Block RAM Configuration and Size</th>
<th>WADDR Port Size (Bits)</th>
<th>WDATA Port Size (Bits)</th>
<th>RADDR Port Size (Bits)</th>
<th>RDATA Port Size (Bits)</th>
<th>MASK Port Size (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM256x16</td>
<td>256 x 16 (4K)</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>16 [15:0]</td>
</tr>
<tr>
<td>SB_RAM256x16NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM256x16NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM512x8</td>
<td>512 x 8 (4K)</td>
<td>9 [8:0]</td>
<td>8 [7:0]</td>
<td>9 [8:0]</td>
<td>8 [7:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM512x8NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM512x8NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM512x8NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM1024x4NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM1024x4NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM1024x4NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM2048x2</td>
<td>2048 x 2 (4K)</td>
<td>11 [10:0]</td>
<td>2 [1:0]</td>
<td>11 [10:0]</td>
<td>2 [1:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM2048x2NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM2048x2NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM2048x2NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For iCE40 EBR primitives with a negative-edged read or write clock, the base primitive name is appended with a ‘N’ and a ‘R’ or ‘W’ depending on the clock that is affected (see Table 4.2 for the 256x16 RAM block configuration).

### Table 4.2. Naming Convention for RAM Primitives

<table>
<thead>
<tr>
<th>RAM Primitive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM256x16</td>
<td>Positive-edged read clock, positive-edged write clock</td>
</tr>
<tr>
<td>SB_RAM256x16NR</td>
<td>Negative-edged read clock, positive-edged write clock</td>
</tr>
<tr>
<td>SB_RAM256x16NW</td>
<td>Positive-edged read clock, negative-edged write clock</td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</td>
<td>Negative-edged read clock, negative-edged write clock</td>
</tr>
</tbody>
</table>
### Table 4.3. Address and Data Mapping for the Initialize RAM Configuration (Write Data)

<table>
<thead>
<tr>
<th>Mode</th>
<th>WADDR</th>
<th>Configuration</th>
<th>Data Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NA</td>
<td>256 x 16</td>
<td>15,14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>512 x 8</td>
<td>14,14,12,12,10,10, 8, 8, 6, 6, 4, 4, 2, 2, 0, 0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1024 x 4</td>
<td>13,13,13,13, 9, 9, 9, 5, 5, 5, 1, 1, 1</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2048 x 2</td>
<td>11,11,11,11,11,11,11, 3, 3, 3, 3, 3, 3</td>
</tr>
</tbody>
</table>

### Table 4.4. Address and Data Mapping for the Initialize RAM Configuration (Read Data)

<table>
<thead>
<tr>
<th>Mode</th>
<th>RADDR[10:8]</th>
<th>Configuration</th>
<th>Data Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xxx</td>
<td>256x16</td>
<td>15,14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0</td>
</tr>
<tr>
<td>1</td>
<td>xx0</td>
<td>512x8</td>
<td>14,12,10, 8, 6, 4, 2, 0</td>
</tr>
<tr>
<td></td>
<td>xx1</td>
<td></td>
<td>15,13,11, 9, 7, 5, 3, 1</td>
</tr>
<tr>
<td>2</td>
<td>x00</td>
<td>1024x4</td>
<td>12, 8, 4, 0</td>
</tr>
<tr>
<td></td>
<td>x01</td>
<td></td>
<td>13, 9, 5, 1</td>
</tr>
<tr>
<td></td>
<td>x10</td>
<td></td>
<td>14,10, 6, 2</td>
</tr>
<tr>
<td></td>
<td>x11</td>
<td></td>
<td>15,11, 7, 3</td>
</tr>
<tr>
<td>3</td>
<td>000</td>
<td>2048x2</td>
<td>8, 0</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td></td>
<td>9, 1</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td></td>
<td>10, 2</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td></td>
<td>11, 3</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td></td>
<td>12, 4</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td></td>
<td>13, 5</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td></td>
<td>14, 6</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
<td>15, 7</td>
</tr>
</tbody>
</table>
4.1. SB_RAM256x16

Verilog Instantiation

SB_RAM256x16 ram256x16_inst (  
  .RDATA(RDATA_c[15:0]),  
  .RADDR(RADDR_c[7:0]),  
  .RCLK(RCLK_c),  
  .RCLKE(RCLKE_c),  
  .RE(RE_c),  
  .WADDR(WADDR_c[7:0]),  
  .WCLK(WCLK_c),  
  .WCLKE(WCLKE_c),  
  .WDATA(WDATA_c[15:0]),  
  .WE(WE_c),  
  .MASK(MASK_c[15:0])  
);

defparam ram256x16_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
VHDL Instantiation

```
ram256x16_inst : SB_RAM256x16 generic map ( 
  INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000", 
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
) 
port map ( 
  RDATA => RDATA_c, 
  RADDR => RADDR_c, 
  RCLK => RCLK_c, 
  RCLKE => RCLKE_c, 
  RE => RE_c, 
  WADDR => WADDR_c, 
  WCLK=> WCLK_c, 
  WCLKE => WCLKE_c, 
  WDATA => WDATA_c, 
  MASK => MASK_c, 
  WE => WE_c 
);
```

Table 4.5 is a complete list of SB_RAM256x16 based primitives.

**Table 4.5. SB_RAM256x16 Based Primitives**

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM256x16</td>
<td>SB_RAM256x16 //Positive edged clock RCLK WCLK</td>
</tr>
<tr>
<td></td>
<td>(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM256x16NR</td>
<td>SB_RAM256x16NR // Negative edged Read Clock – i.e. RCLKN</td>
</tr>
<tr>
<td></td>
<td>(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM256x16NW</td>
<td>SB_RAM256x16NW // Negative edged Write Clock – i.e. WCLK</td>
</tr>
<tr>
<td></td>
<td>(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</td>
<td>SB_RAM256x16NRNW // Negative edged Read and Write – i.e. RCLKN WRCLKN</td>
</tr>
<tr>
<td></td>
<td>(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
</tbody>
</table>
4.2. SB_RAM512x8

Verilog Instantiation

SB_RAM512x8 inst 
  .RDAT(RDATA_c[7:0]),
  .RADDR(RADDR_c[8:0]),
  .RCLK(RCLK_c),
  .RCLKE(RCLKE_c),
  .RE(RE_c),
  .WADDR(WADDR_c[8:0]),
  .WCLK(WCLK_c),
  .WCLKE(WCLKE_c),
  .WDATA(WDATA_c[7:0]),
  .WE(WE_c)
);

defparam ram512x8 inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8 inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_B =
256’h0000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_C =
256’h0000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_D =
256’h0000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_E =
256’h0000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_F =
256’h0000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation

ram512x8_inst : SB_RAM512x8 generic map {
    INIT_0 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_1 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_2 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_3 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_4 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_5 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_6 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_7 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_8 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_9 => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_A => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_B => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_C => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_D => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_E => X"0000000000000000000000000000000000000000000000000000000000",
    INIT_F => X"0000000000000000000000000000000000000000000000000000000000"
} port map (
    RDATA => RDATA_c,
    RADDR => RADDR_c,
    RCLK => RCLK_c,
    RCLKE => RCLKE_c,
    RE => RE_c,
    WADDR => WADDR_c,
    WCLK=> WCLK_c,
    WCLKE => WCLKE_c,
    WDATA => WDATA_c,
    WE => WE_c
);
WE => WE_c
);

Table 4.6 is a complete list of SB_RAM512x8 based primitives.

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM512x8</td>
<td>SB_RAM512x8 //Positive edged clock RCLK WCLK (RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM512x8NR</td>
<td>SB_RAM512x8NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM512x8NW</td>
<td>SB_RAM512x8NW // Negative edged Write Clock – i.e. WCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM512x8NRNW</td>
<td>SB_RAM512x8NRNW // Negative edged Read and Write – i.e. RCLKN WRCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
</tbody>
</table>
4.3. SB_RAM1024x4

Verilog Instantiation

```
SB_RAM1024x4 ram1024x4_inst (  
  .RDATA(RDATA_c[3:0]),  
  .RADDR(RADDR_c[9:0]),  
  .RCLK(RCLK_c),  
  .RCLKE(RCLKE_c),  
  .RE(RE_c),  
  .WADDR(WADDR_c[9:0]),  
  .WCLK(WCLK_c),  
  .WCLKE(WCLKE_c),  
  .WDATA(WDATA_c[3:0]),  
  .WE(WE_c)
);
```
```vhdl
defparam raml0124x4_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam raml0124x4_inst.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam raml0124x4_inst.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam raml0124x4_inst.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam raml0124x4_inst.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation

Ram1024X4_inst : SB_RAM1024x4 generic map ( 
  INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000" 
) 

port map ( 
  RDATA => RDATA_c,
  RADDR => RADDR_c,
  RCLK => RCLK_c,
  RCLKE => RCLKE_c,
  RE => RE_c,
  WADDR => WADDR_c,
  WCLK=> WCLK_c,
  WCLKE => WCLKE_c,
  WDATA => WDATA_c,
  WE => WE_c
); 

Table 4.7 is a complete list of SB_RAM1024x4 based primitives.

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM1024x4</td>
<td>SB_RAM1024x4 //Positive edged clock RCLK WCLK (RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM1024x4NR</td>
<td>SB_RAM1024x4NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM1024x4NW</td>
<td>SB_RAM1024x4NW // Negative edged Write Clock – i.e. WCLKN (RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM1024x4NRNW</td>
<td>SB_RAM1024x4NRNW // Negative edged Read and Write – i.e. RCLKN WRCILN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
</tbody>
</table>
4.4. SB_RAM2048x2

Verilog Instantiation

```verilog
SB_RAM2048x2 ram2048x2_inst {
    .RDATA(RDATA_c[1:0]),
    .RADDR(RADDR_c[10:0]),
    .RCLK(RCLK_c),
    .RCLKE(RCLKE_c),
    .RE(RE_c),
    .WADDR(WADDR_c[10:0]),
    .WCLK(WCLK_c),
    .WCLKE(WCLKE_c),
    .WDATA(WDATA_c[10:0]),
    .WE(WE_c)
};

defparam ram2048x2_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
```

Figure 4.4. SB_RAM2048x2
defparam ram2048x2_inst .INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation

Ram2048x2_inst : SB_RAM2048x2

generic map (    
  INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",    
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"  )

call map (    
  RDATA => RDATA_c,    
  RADDR => RADDR_c,    
  RCLK => RCLK_c,    
  RCLKE => RCLKE_c,    
  RE => RE_c,    
  WADDR => WADDR_c,    
  WCLK => WCLK_c,    
  WCLKE => WCLKE_c,    
  WE => WE_c  )

Table 4.8 is a complete list of the SB_RAM2048x2 based primitives.

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM2048x2</td>
<td>SB_RAM2048x2 // Positive edged clock RCLK WCLK (RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM2048x2NR</td>
<td>SB_RAM2048x2NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM2048x2NW</td>
<td>SB_RAM2048x2NW // Negative edged Write Clock – i.e. WCLKN (RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
<tr>
<td>SB_RAM2048x2NRRN</td>
<td>SB_RAM2048x2NRRN // Negative edged Read and Write – i.e. RCLKN WRCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);</td>
</tr>
</tbody>
</table>
4.5. SB_RAM40_4K

SB_RAM40_4K is the basic physical RAM primitive which can be instantiated and configured to different depths and data ports. The SB_RAM40_4K block has a size of 4 Kbits with separate write and read ports, each with independent control signals. By default, input and output data is 16 bits wide, although the data width is configurable using the READ_MODE and WRITE_MODE parameters. The data contents of the SB_RAM40_4K block are optionally pre-loaded during iCE device configuration.

Table 4.9. SB_RAM40_4K Naming Convention Rules

<table>
<thead>
<tr>
<th>RAM Primitive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM40_4K</td>
<td>Positive-edged read clock, positive-edged write clock</td>
</tr>
<tr>
<td>SB_RAM40_4KNR</td>
<td>Negative-edged read clock, positive-edged write clock</td>
</tr>
<tr>
<td>SB_RAM40_4KNW</td>
<td>Positive-edged read clock, negative-edged write clock</td>
</tr>
<tr>
<td>SB_RAM40_4KNRNW</td>
<td>Negative-edged clock, negative-edged write clock</td>
</tr>
</tbody>
</table>

![Figure 4.5. SB_RAM40_4K](image)

Table 4.10 lists the signals for both ports.

Table 4.10. SB_RAM40_4K Signal Descriptions

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDATA[15:0]</td>
<td>Input</td>
<td>Write Data input.</td>
</tr>
<tr>
<td>MASK[15:0]</td>
<td>Input</td>
<td>Bit-line Write Enable input, active low. Applicable only when WRITE_MODE parameter is set to '0'.</td>
</tr>
<tr>
<td>WADDR[7:0]</td>
<td>Input</td>
<td>Write Address input. Selects up to 256 possible locations.</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>Write Enable input, active high.</td>
</tr>
<tr>
<td>WCLK</td>
<td>Input</td>
<td>Write Clock input, rising-edge active.</td>
</tr>
<tr>
<td>WCLKE</td>
<td>Input</td>
<td>Write Clock Enable input.</td>
</tr>
<tr>
<td>RDATA[15:0]</td>
<td>Output</td>
<td>Read Data output.</td>
</tr>
<tr>
<td>RADDR[7:0]</td>
<td>Input</td>
<td>Read Address input. Selects one of 256 possible locations.</td>
</tr>
<tr>
<td>RE</td>
<td>Input</td>
<td>Read Enable input, active high.</td>
</tr>
<tr>
<td>RCLK</td>
<td>Input</td>
<td>Read Clock input, rising-edge active.</td>
</tr>
<tr>
<td>RCLKE</td>
<td>Input</td>
<td>Read Clock Enable input.</td>
</tr>
</tbody>
</table>

Table 4.11 on the next page describes the parameter values to infer the desired RAM configuration.
Table 4.11. SB_RAM40_4K Primitive Parameter Descriptions

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_0, ... ,INIT_F</td>
<td>RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16 x 256=4096 total bits)</td>
<td>INIT_0 to INIT_F</td>
<td>Initialize the RAM with predefined value</td>
</tr>
<tr>
<td>WRITE_MODE</td>
<td>Sets the RAM block write port configuration</td>
<td>0</td>
<td>256 x 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>512 x 18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1024 x 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>2048 x 2</td>
</tr>
<tr>
<td>READ_MODE</td>
<td></td>
<td>0</td>
<td>256 x 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>512 x 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1024 x 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>2048 x 2</td>
</tr>
</tbody>
</table>

Verilog Instantiation

```verilog
// Physical RAM Instance without Pre Initialization
SB_RAM40_4K ram40_4kinst_physical (  
    .RDATA(RDATA),  
    .RADDR(RADDR),  
    .WADDR(WADDR),  
    .MASK(MASK),  
    .WDATA(WDATA),  
    .RCLKE(RCLKE),  
    .RCLK(RCLK),  
    .RE(RE),  
    .WCLKE(WCLKE),  
    .WCLK(WCLK),  
    .WE(WE)  
);  
defparam ram40_4kinst_physical.READ_MODE=0;  
defparam ram40_4kinst_physical.WRITE_MODE=0;  
```

VHDL Instantiation

```vhdl
-- Physical RAM Instance without Pre Initialization  
ram40_4kinst_physical : SB_RAM40_4K  
generic map (  
    READ_MODE => 0,  
    WRITE_MODE=> 0  
)  
port map (  
    RDATA=>RDATA,  
    RADDR=>RADDR,  
    WADDR=>WADDR,  
    MASK=>MASK,  
    WDATA=>WDATA,  
    RCLKE=>RCLKE,  
    RCLK=>RCLK,  
    RE=>RE,  
    WCLKE=>WCLKE,  
    WCLK=>WCLK,  
    WE=>WE  
);  
```
5. EBR Utilization Summary in iCEcube2 Design Software

The placer.log file in the iCEcube2 design software shows the device utilization summary. The Final Design Statistics and Device Utilization Summary sections show the number of EBRs (or RAMs) used against the total number. Figure 5.1 shows the EBR usage when one SB_RAM256x16 was instantiated in the design.

![Device/Operating Condition](image1)

![Final Design Statistics](image2)

![Device Utilization Summary](image3)

**Figure 5.1. iCEcube2 Design Software Report File**
Appendix A. Standard HDL Code References

This appendix contains standard HDL and VHDL codes for popular memory elements, which can be used to infer a sysMEM EBR automatically. Standard HDL coding techniques do not require you to know the details of the block RAMs of the device and are inferred automatically.

A.1. Single-Port RAM

Verilog

```verilog
module ram (din, addr, write_en, clk, dout); // 512x8
    parameter addr_width = 9;
    parameter data_width = 8;
    input [addr_width-1:0] addr;
    input [data_width-1:0] din;
    input write_en, clk;
    output [data_width-1:0] dout;
    reg [data_width-1:0] mem[(1<<addr_width)-1:0];
    always @(posedge clk)
        begin
            if (write_en)
                mem[addr] <= din;
            dout = mem[addr]; // Output register controlled by clock.
        end
endmodule
```

VHDL

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
    generic(
        addr_width : natural := 9;--512x8
        data_width : natural := 8);
    port (
        addr : in std_logic_vector (addr_width - 1 downto 0);
        write_en : in std_logic;
        clk : in std_logic;
        din : in std_logic_vector (data_width - 1 downto 0);
        dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
    type mem_type is array ((2**addr_width) - 1 downto 0) of
        std_logic_vector(data_width - 1 downto 0);
    signal mem : mem_type;
    begin
        process (clk)
        begin
            if (clk'event and clk = '1') then
                if (write_en = '1') then
                    mem(conv_integer(addr)) <= din; -- Using write address bus.
                end if;
                dout <= mem(conv_integer(addr));
            end if;
        end process;
    end rtl;
```
A.2. Dual Port RAM

Verilog

```verilog
module ram (din, write_en, waddr, wclk, raddr, rclk, dout); // 512x8
    parameter addr_width = 9;
    parameter data_width = 8;
    input [addr_width-1:0] waddr, raddr;
    input [data_width-1:0] din;
    input write_en, wclk, rclk;
    output reg [data_width-1:0] dout;
    reg [data_width-1:0] mem [(1<<addr_width)-1:0];

    always @(posedge wclk) // Write memory.
    begin
        if (write_en)
            mem[waddr] <= din; // Using write address bus.
    end

    always @(posedge rclk) // Read memory.
    begin
        dout <= mem[raddr]; // Using read address bus.
    end
endmodule
```

VHDL

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
    generic (
        addr_width : natural := 9;--512x8
        data_width : natural := 8);
    port (
        write_en : in std_logic;
        waddr : in std_logic_vector (addr_width - 1 downto 0);
        wclk : in std_logic;
        raddr : in std_logic_vector (addr_width - 1 downto 0);
        rclk : in std_logic;
        din : in std_logic_vector (data_width - 1 downto 0);
        dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
    type mem_type is array ((2** addr_width) - 1 downto 0) of
        std_logic_vector(data_width - 1 downto 0);
    signal mem : mem_type;
begin
    process (wclk)
    begin
        if (wclk'event and wclk = '1') then
            if (write_en = '1') then
                mem(conv_integer(waddr)) <= din;
            end if;
    end process;
end rtl;
```

```-- Output register controlled by clock.
```
-- Using write address bus.
end if;
end if;
end process;

process (rclk) -- Read memory.
begin
  if (rclk'event and rclk = '1') then
    dout <= mem(conv_integer(raddr));
    -- Using read address bus.
  end if;
end process;
end rtl;
References
For more information, refer to the following documents:

- iCE40 LP/HX Family Data Sheet (FPGA-DS-02029)
- iCE40 LM Family Data Sheet (FPGA-DS-02043)
- Programming Cables User’s Guide (FPGA-DS-02024)
Technical Support Assistance
Submit a technical support case via www.latticesemi.com/techsupport.
Revision History

Revision 1.7, September 2020

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disclaimers</td>
<td>Added this section.</td>
</tr>
<tr>
<td>iCE40 sysMEM Embedded Block RAM</td>
<td>Update the note in Figure 3.2. EBR Module Timing Diagram.</td>
</tr>
</tbody>
</table>
| iCE40 sysMEM Embedded Block RAM Memory Primitives | Added the following tables:  
  - Table 4.3. Address and Data Mapping for the Initialize RAM Configuration (Write Data)  
  - Table 4.4. Address and Data Mapping for the Initialize RAM Configuration (Read Data) |
| References | Updated document numbers of referenced data sheets and user guide. |

Revision 1.6, August 2017

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
</table>
| All | Added Acronyms in This Document section.  
  - Changed document number from TN1250 to FPGA-TN-02002.  
  - Updated document template. |

Revision 1.5, June 2016

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Updated Introduction section. Added iCE40 UltraPlus to introductory paragraph.</td>
</tr>
<tr>
<td>Signals</td>
<td>Updated Signals section. Added footnote to RE signal in Table 3.1. EBR Signal Descriptions.</td>
</tr>
<tr>
<td>Timing Diagram</td>
<td>Updated Timing Diagram section. Revised Figure 3.2. EBR Module Timing Diagram.</td>
</tr>
<tr>
<td>Memory Initialization</td>
<td>Added Memory Initialization section.</td>
</tr>
</tbody>
</table>
| iCE40 sysMEM Embedded Block RAM Memory Primitives | Updated SB_RAM512x8 section. Minor correction in Verilog Instantiation and VHDL Instantiation.  
  - Updated SB_RAM1024x4 section. Minor correction in VHDL Instantiation.  
  - Revised SB_RAM512x8 Verilog Instantiation and VHDL Instantiation.  
  - Revised SB_RAM1024x4 Verilog Instantiation and VHDL Instantiation.  
  - Revised SB_RAM2048x2 Verilog Instantiation. |
| Technical Support Assistance | Updated Technical Support Assistance section. |

Revision 1.4, January 2015

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Added support for iCE40 UltraLite.</td>
</tr>
</tbody>
</table>

Revision 1.3, June 2014

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Added support for iCE40 Ultra.</td>
</tr>
</tbody>
</table>

Revision 1.2, December 2013

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>iCE40 sysMEM Embedded Block RAM</td>
<td>Added information to Table 3.1. EBR Signal Descriptions</td>
</tr>
</tbody>
</table>
### Revision 1.1, October 2013

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Removed iCE40 Family EBRs table.</td>
</tr>
<tr>
<td>Technical Support Assistance</td>
<td>Updated Technical Support Assistance information.</td>
</tr>
</tbody>
</table>

### Revision 1.0, September 2012

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>