Single Port Block Memory

May 28, 1999 Product Specification

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
E-mail: coregen@xilinx.com
URL: www.xilinx.com

Features

- Supports RAM, ROM and Write Only functions
- Supports data widths from 1 to 1024 bits
- Supports memory depths from 16 to 1M words
- Uses Virtex™ block memory for performance and efficiency
- Allows power-on memory content to be defined
- Fully synchronous
- Drop-in modules for the Virtex™ family
- Available in Xilinx CORE Generator™ System

Functional Description

The Block RAM takes an N-bit data value and an M-bit address. When the Block RAM is disabled (EN inactive) the memory configuration and output value remain unaltered. When enabled (EN asserted) all memory operations occur on the active edge of the clock input (CLK). During a write operation (WE asserted) the data value is stored at the location selected by the address. If the memory is not in reset mode (RST inactive) the data value will also appear at the module's output. During a read operation (WE and RST inactive) the memory contents at the location selected by the address will appear at the module's output. While in reset mode (RST asserted) the module's outputs are all LOW, although memory write operations may still take place. (Asserting RST has no effect on memory contents.)

The initial contents of the memory (i.e. the data stored in the memory immediately after device configuration) may also be specified.

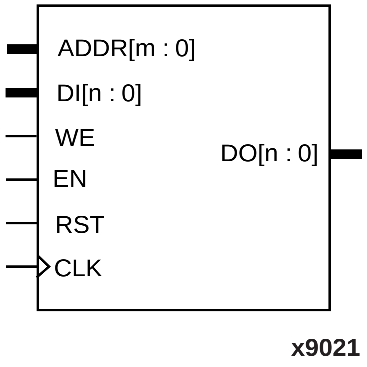
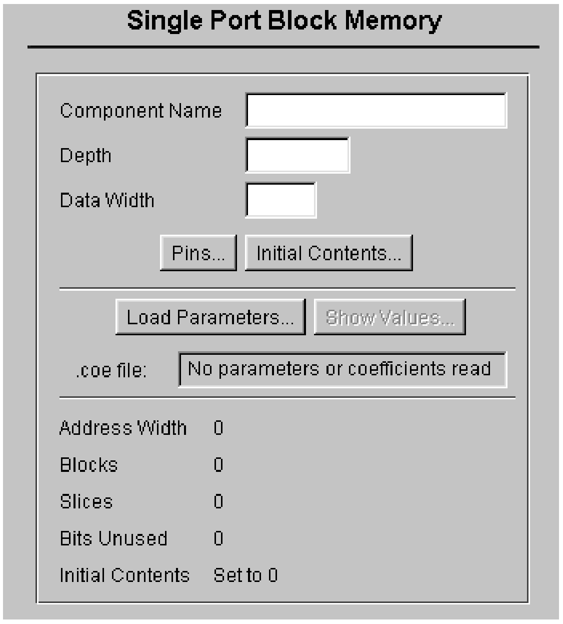


Figure 1: Core Schematic Symbol

Pinout

Port names for the core module are shown in Figure 1 and described in Table 1. The inclusion of some ports on the module is optional; exclusion of these ports will alter the


Single Port Block Memory

Component Name

Depth

Data Width

.coe file: No parameters or coefficients read

Address Width	0
Blocks	0
Slices	0
Bits Unused	0
Initial Contents	Set to 0

Figure 2: Parameterization Window

function of the module (e.g., by excluding the DI and WE ports the Block RAM becomes a Block ROM). The optional ports are marked in Table 1.

CORE Generator Parameters

The CORE Generator parameterization window for this module is shown in Figure 2. The Parameters tab of this window is divided into a number of sections. The interactive section of the window contains parameters that the user can define. These are as follows:

- **Component Name:** Enter a name for the output files generated for this module.
- **Depth:** Select the number of words in the memory. Values should be 16, 32, 64, 128 or multiples of 256. Invalid entries will be set to the nearest value greater in depth. The absolute maximum number of words is 1048576 (or 1M), but useful cores should not exceed the number of Block RAM primitives available in the required device. (1M words would require more primitives than are available in any current device.)
- **Data Width:** Select the data bit width. Values must be greater than 1. There is no upper limit but useful cores should not exceed the number of Block RAM primitives available in the required device.
- **Pins…:** Press the button to display the Pins dialog box on the screen.
- **Initial Contents…:** Press the button to display the Initial Contents dialog box on the screen.

The Pins dialog box is shown in Figure 4. The box contains parameter fields that the user can define, relating to the optional pins and the control pin polarity. These fields are as follows:

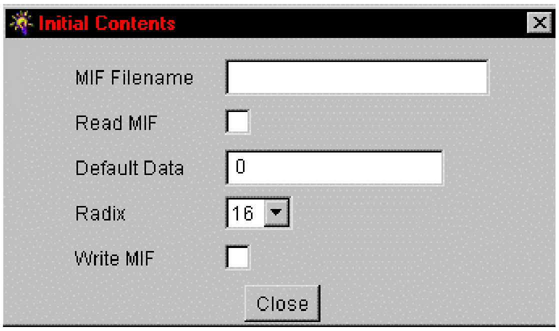
- **Write Port:** Check the box to include the DI and WE ports on the module; uncheck the box to remove them.
- **Read Port:** Check the box to include the DO port on the module; uncheck the box to remove it.
- **Enable:** Check the box to include the EN port on the module; uncheck the box to remove it.
- **Output Reset:** Check the box to include the RST port on the module; uncheck the box to remove it.
- **WE Active:** Choose between an active High and active Low WE port. This choice will only be available if the Write Port box is checked.
- **Clock On:** Choose whether the CLK port is active on the rising edge or falling edge.
- **EN Active:** Choose between an active High and active Low EN port. This choice will only be available if the Enable box is checked.
- **RST Active:** Choose between an active High and active Low RST port. This choice will only be available if the Output Reset box is checked.

Table 1: Core Signal Pinout

Signal	Signal Direction	Description
ADDR[m:0]	Input	ADDRESS – memory location to which data will be written or from which data will be read.
DI[n:0] (Optional)	Input	DATA INPUT – data to be written into memory.
WE (Optional)	Input	WRITE ENABLE – control signal used to allow transfer of data input into memory. Active state of WE is user defined (default is active High).
EN (Optional)	Input	ENABLE – control signal used to allow any memory operations to take place. Active state of EN is user defined (default is active High).
RST (Optional)	Input	RESET – control signal used to force the modules outputs LOW. Active state of RST is user defined (default is active High). RST does not affect memory contents.
CLK	Input	CLOCK – when memory is enabled, control and data inputs are registered and new output data formed on active clock edge. Active edge of clock is user defined (default is rising edge).
DO[n:0] (Optional)	Output	DATA OUTPUT – when the memory is enabled this port reflects data stored at the location selected by the address during read mode, data input during write mode or LOW during reset mode. When memory is disabled, maintains the previous output data value.

The Initial Contents dialog box is shown in Figure 3. The box contains parameter fields that the user can define, relating to the data stored in the memory directly after device configuration. These fields are as follows:

- **MIF Filename:** Enter the name of a file to which the memories initial contents will be read from or to, which they will be written. This defaults to the modules component name. If no extension is given this will default to .mif.



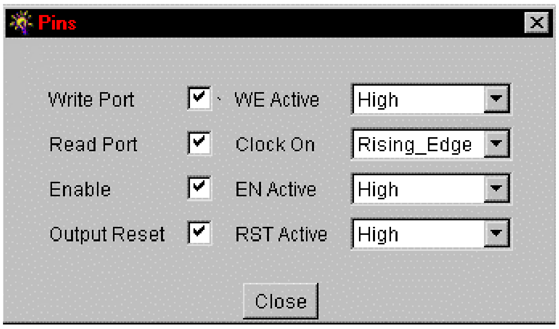
MIF Filename	
Read MIF	☐
Default Data	0
Radix	16
Write MIF	☐

Figure 3: Initial Contents Dialog Box

- **Read MIF:** Check this box if the named MIF file exists and the initial memory contents are to be read from this file.
- **Default Data:** Enter the initial value to be stored in any memory location not specified by another means. When no value is entered this field defaults to 0. Values may be entered in Binary, Decimal or Hex, as defined by the Radix entry.
- **Radix:** Choose the radix of the Default Data value. Valid entries are 2,10 and 16. If the memory's initial contents have been defined by an entry in a coefficient file (.coe – see below) the radix will become fixed.
- **Write MIF:** Check this box if the initial contents of the memory are to be written to the named MIF file. In most cases this parameter will be unchecked; the file will be written by default where necessary.
- **Close:** Press the button to close the dialog box and return to the parameterization window.

For further information regarding the memory's initial contents refer to the *Specifying Memory Contents* section.

The .coe file section of the window allows the user to load parameter values from a coefficient file, including initial and default memory contents. The .coe file section fields are:



Write Port	✓	WE Active	High
Read Port	✓	Clock On	Rising_Edge
Enable	✓	EN Active	High
Output Reset	✓	RST Active	High

Figure 4: Pins Dialog Box

```
Component_Name=ram256x16;
Data_Width=16;
Depth=256;
Radix=16;
Default_Data=F;
Memory_Initialization_Vector=123,  456,
 aaaa;
```

Figure 5: An example COE file for Single Port Block RAM

- **Load Init Values:** Specifies the file that contains the parameter values for the memory.
- **Show Invalid Values:** Display any initial content values that are invalid given the chosen parameters, once they have been loaded.
- **.coe file:** Displays the name of the coefficient file. This field is read only.

The report section of the window provides feedback about the memory with the parameter values as set by the user. These fields are:

- **Address Width:** Shows the number of bits needed to address all of the words in the memory.
- **Blocks*:** Shows the number of Block RAM primitives required to implement the specified Depth and Data width.
- **Slices:** For some memory depths address decoding and output multiplexing external to the Block RAM primitives is required. This field shows an estimate of the number of Virtex™ slices used for the extra logic.
- **Bits Unused:** For some memory depths and data widths the Block RAM primitives (needed to construct the user's specified memory) may not be 100% utilized. This field shows the number of Block RAM bits which are unused.

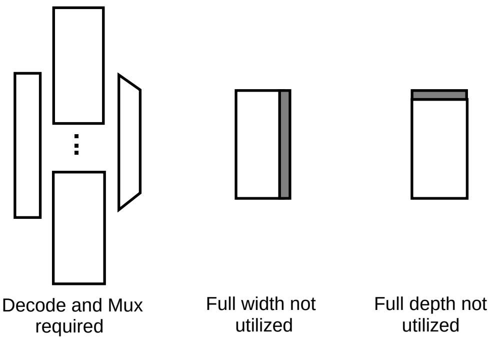


Figure 6: Symbol Annotations and Their Meaning

- **Initial Contents:** Shows how the initial contents of the memory are specified either: To be read from (MIF) file, Loaded from coe file or Set to (default value).

Finally, the Symbol section of the window updates itself in response to the user's parameter selections. The user's Depth and Data width settings are reflected on the ADDR, DIand DO bus widths. Active Low control signals are illustrated with a circle on the pin. Other annotations on the symbol and their meanings are shown in Figure 6.

***Note:** Ensure that the target device has sufficient Block RAM primitives to accommodate the specified memory, including any primitives used elsewhere in the application. (Each primitive is equivalent to 4096 bits of storage. Table 2 shows how many primitives are available in each device.)

Specifying Memory Contents

The initial contents of the memory can be assigned by specifying the desired information in a text file known as a memory initialization file (MIF). MIF files may take any root file name and extension, although the extension will normally be ".mif".

The MIF file consists of one line per memory location, starting from 0 and running consecutively. The file must not contain more lines than there are memory locations, but may contain fewer. In the latter case all other locations will be initialized to the default value. Each individual line must contain the value for that location in binary format, with exactly one binary digit per bit in the memory's width.

To specify a MIF file to use as the initial memory contents press the "Initial Contents…" button, enter the file's name in the MIF Filename field and then check the Read MIF box. At this point the default data value and radix can be entered in their respective fields, if required. The MIF file will be read during core generation.

The initial contents of the memory can also be assigned by specifying the desired information in a separate text file called a COE file. In addition to the initial memory contents, all the parameters visible on the parameterization window may be assigned values in the COE file. COE files may take any root file name but must end with the extension ".coe".

To select and load a COE file, press the "Load Init Values…" button on the parameterization window and choose the desired file from the dialog box. Any field on the parameterization window that is assigned a value in the COE file will lose its previous value when the COE file is loaded. Changing a parameter value that was previously loaded from a COE file causes the COE file's name to be highlighted in red, indicating that the settings have changed since the file was loaded. If any of the initialization values are inconsistent with the other parameters specified, an error is issued. The inconsistent data can then be reviewed by pressing the "Show Invalid Values…" button, which will now be highlighted in red.

For a detailed description of the COE file syntax, please refer to the Xilinx CORE Generator User Guide. The COE keywords supported by the Single Port Block Memory module are shown in the Parameter File Information table at the end of this data sheet. An example COE file is shown in Figure 5.

When specifying the initial contents for a memory in a COE file the keywords **DEFAULT_DATA**, **MEMORY_INITIALIZATION_VECTOR**, **RADIX** and **READ_MIF** may be used. The **DEFAULT_DATA** keyword allows a value to be assigned to all memory locations with a single statement. If not set, the **DEFAULT_DATA** value is 0. (In general, data values that require fewer than **DATA_WIDTH** bits to express will be padded with "0"s at their most significant end. In the example below the **DEFAULT_DATA** value "F" is assumed to be "000F".) The **DEFAULT_DATA** value is overridden by the **MEMORY_INITIALIZATION_VECTOR** but only for the memory locations covered by the vector.

The **MEMORY_INITIALIZATION_VECTOR** takes the form of a sequence of comma separated values, one value per memory location, terminated by a semi-colon. Any amount of white space, including new lines, can be included in the vector to enhance readability. The format of an individual value in the vector will depend on the **RADIX** value, which can be "2", "10" or "16", (the default value is 16). The vector values must be consistent with the **RADIX** value and must fall within the range of 0 to 2^{DATA_WIDTH}-1. Values must not be negative. Additionally, the initial values for the memory can be read from a memory initialization file (.mif) by setting the **READ_MIF** parameter. When **READ_MIF** is true the **MEMORY_INITIALIZATION_VECTOR** values are over written with values from the .mif file. This allows the initial memory contents to be developed in conjunction with the HDL behavioral models and then used when generating the final module. The name of the .mif file can be specified with the **MIF_FILENAME** parameter. This name is used for reading and generating the MIF file, and its root name defaults to the **COMPONENT_NAME** parameter values.

HDL Simulation

The behavioural models (VHDL and Verilog) for the memory components initialize their contents by reading a memory initialization file (MIF). This file (specified by the **MIF_FILENAME** parameter) is written into the project directory when the module is generated. Both the Verilog and the VHDL behavioural models read the same .mif file. The C_READ_MIF generic (or parameter for Verilog) must be set to '1' in order for the model to read the .mif file. If C_READ_MIF is set to '1' and the .mif file is not present in the simulator project directory before the simulation begins, an error will be issued. If C_READ_MIF is set to '0' the memory will be initialized with the default data value as specified by the C_DEFAULT_DATA generic (or parameter).

For backward compatibility .mif files can also be generated from the behavioural models and read in during module generation (see previous section). This function is available by setting the generatemif signal within the behavioural model to 1, during simulation. The memory contents will be written to the file (specified by the **MIF_FILENAME** parameter, overwriting the previous contents) on every inactive clock edge until the generatemif signal is set to 0. As writing the .mif file may be time consuming it is advised to use this function sparingly.

Note: As the VHDL behavioural models must be VHDL-87 and VHDL-93 compliant, the default VHDL model does not contain this feature. User's wishing to use this feature should compile either, the mem_init_file_pack.vhd87 (for VHDL-87 compilers) or the mem_init_file_pack.vhd93 (for VHDL-93 compilers) package, into XilinxCoreLib. Compilation of these files directly supersedes the mem_init_file_pack package and the .mif file reading and writing procedures, compiled from mem_init_file_pack.vhd by default, with VHDL-87 or VHDL-93 equivalents. Both packages can be downloaded from:

http:\\www.xilinx.com\ipcenter

Core Resource Utilization

The number of Block RAM primitives required is dependent on the values of the depth and data width fields selected in the CORE generator parameterization window, and is equal to:

(depth*data_width)/4096.

Table 2 details the smallest device in the Virtex™ family, that provides a particular number of primitives.

For some memory depths extra logic is required to decode the address and multiplex the outputs from various primitives. Virtex™ CLB slices are used to provide this functionality. The number of slices required depends on the way that the depth is constructed from differing primitives, the data width and the way that the decode and mux are constructed. Calculating the number of slices used is therefore a non-trivial task and the best way to obtain an estimate for a specified memory is to enter its parameters into the parameterization window and read the report section.

Ordering Information

This macro comes free with the Xilinx CORE Generator™ System. For additional information contact your local Xilinx sales representative, or e-mail requests to: coregen@xilinx.com.

Table 2: Block RAM Primitives Available by Device

Blocks	Minimum Device
1	XCV50
2	
3	
4	
5	
6	
7	
8	
9	XCV100
10	
11	XCV150
12	
13	XCV200
14	
15	XCV300
16	
17	XCV400
18	
19	
20	
21	XCV600
22	
23	
24	
25	XCV800
26	
27	
28	
29	XCV1000
30	
31	
32	

Table 3: Parameter File Information

Parameter Type	Type	Notes
Component_Name	String	
Depth	Integer	16,32,64,128 or multi-ples of 256 (up to 1M)
Data_Width	Integer	> 1
Write_Port	Boolean	Default = true
Read_Port	Boolean	Default = true
Enable	Boolean	Default = true
Output_Reset	Boolean	Default = true
WE_Active	String	High or Low Default = High
EN_Active	String	High or Low Default = High
RST_Active	String	High or Low Default = High
Clock_On	String	Rising_Edge or Falling_Edge Default = Rising_Edge
Radix	Integer	2,10 or 16 Default = 16
Default_Data	Integer	Default = 0
Memory_Initializa-tion_Vector*	Integer List	Comma seperated and semi-colon terminated
Write_MIF	Boolean	Default = false
Read_MIF	Boolean	Default = false
MIF_FileName	String	Default =Component_Name
*used in COE and batch files only.		