

Introduction

The BRAM Block is a configurable memory module that attaches to a variety of BRAM Interface Controllers.

The BRAM Block structural HDL is generated by the EDK design tools based on the configuration of the BRAM interface controller IP. All BRAM Block parameters are automatically calculated and assigned by the Platgen and Simgen EDK tools.

Features

- Fully automated generation and configuration of HDL through EDK Platgen and Simgen tools.
- Number of BRAM primitives utilized is a function of the configuration parameters for: memory address range, number of byte-write enables, the data width, and the targeted architecture
- Both Port A and Port B of the memory block can be connected to independent BRAM Interface Controllers: LMB (Local Memory Bus), PLB (Processor Local Bus), and OCM (On-Chip Memory).
 - ◆ Supports byte, half-word, word, and doubleword transfers provided the correct number of byte-write enables have been configured

| LogiCORE IP Facts | | | | |
|-----------------------------|--|------|-----|------------|
| Core Specifics | | | | |
| Supported Device Family | Spartan®-3A/3A DSP, Spartan-3, Spartan-3E, Automotive Spartan 3/3E/3A/3A DSP, Spartan-6, Virtex®-4 /4Q/4QV, Virtex-5/5FX, Virtex-6/6CX | | | |
| Resources Used | I/O | LUTs | FFs | Block RAMs |
| | n | n | n | n |
| Special Features | Add if applicable | | | |
| Provided with Core | | | | |
| Documentation | Product Specification | | | |
| Design File Formats | VHDL | | | |
| Constraints File | .ucf (user constraints file) | | | |
| Verification | VHDL Test Bench | | | |
| Instantiation Template | VHDL Wrapper | | | |
| Additional Items | N/A | | | |
| Design Tool Requirements | | | | |
| Xilinx Implementation Tools | ISE® 12.1 | | | |
| Verification | Mentor Graphics ModelSim: v6.5c and above | | | |
| Simulation | Mentor Graphics ModelSim: v6.5.c and above | | | |
| Synthesis | XST | | | |
| Support | | | | |
| Provided by Xilinx, Inc. | | | | |

Functional Description

The BRAM Block is a structural design that instantiates a number of RAMB primitives, depending on specific factors. An example is shown in [Figure 1](#).

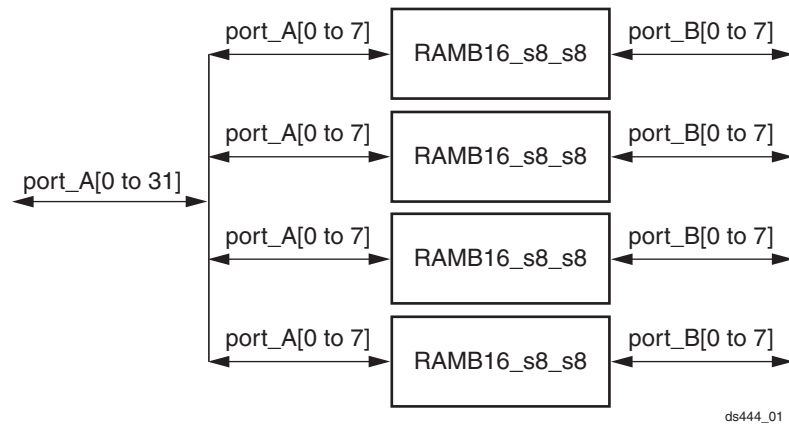


Figure 1: Example of BRAM Block Implementation with Four RAMB16 Primitives

BRAM Block I/O Signals

The I/O signals for the BRAM Block are shown in [Figure 1](#) and described in [Table 1](#). Note that the data in/out signals are named the opposite of their actual direction. This is because the signals are referenced from the point of view of the BRAM controller so that “data in” on the controller connects to “data in” on the BRAM block, and “data out” on the controller connects to “data out” on the BRAM block.

Table 1: BRAM Block I/O Signals

| Signal Name | Interface | I/O | Initial State | Description |
|------------------------------------|-----------|-----|---------------|--|
| BRAM_Rst_A | Port A | I | | BRAM Reset, Active High |
| BRAM_Clk_A | Port A | I | | BRAM Clock |
| BRAM_EN_A | Port A | I | | BRAM Enable, Active High |
| BRAM_WEN_A | Port A | I | | BRAM Write Enable, Active High |
| BRAM_Addr_A (0:C_PORT_AWIDTH-1) | Port A | I | | BRAM Address |
| BRAM_Din_A (0:C_PORT_DWIDTH-1) | Port A | O | 0 | BRAM Data Output . Note signal name is referenced from the point of view of the controller (on which it is an input). |
| BRAM_Dout_A (0:C_PORT_DWIDTH-1) | Port A | I | | BRAM Data Input . Note signal name is referenced from the point of view of the controller (on which it is an output). |
| BRAM_Rst_B | Port B | I | | BRAM Reset, Active High |
| BRAM_Clk_B | Port B | I | | BRAM Clock |
| BRAM_EN_B | Port B | I | | BRAM Enable, Active High |
| BRAM_WEN_B | Port B | I | | BRAM Write Enable, Active High |

Table 1: BRAM Block I/O Signals (Cont'd)

| Signal Name | Interface | I/O | Initial State | Description |
|------------------------------------|-----------|-----|---------------|--|
| BRAM_Addr_B (0:C_PORT_AWIDTH-1) | Port B | I | | BRAM Address |
| BRAM_Din_B (0:C_PORT_DWIDTH-1) | Port B | O | 0 | BRAM Data Output . Note signal name is referenced from the point of view of the controller (on which it is an input). |
| BRAM_Dout_B (0:C_PORT_DWIDTH-1) | Port B | I | | BRAM Data Input . Note signal name is referenced from the point of view of the controller (on which it is an output). |

BRAM Block Parameters

Table 2: BRAM block Parameters

| Parameter Name | Feature/Description | Allowable Values | Tool Calculated | VHDL Type |
|----------------|--|---|-----------------|-----------|
| C_PORT_AWIDTH | Port A and B Address Width | 9 – 17 | Yes | integer |
| C_PORT_DWIDTH | Port A and B Data Width | 32, 64, 128 | Yes | integer |
| C_NUM_WE | Number of Write Enables (Byte Enables for Write) | 1, 2, 4, 8, 16 | Yes | integer |
| C_FAMILY | Target FPGA family of bram_block | spartan3, spartan3e, spartan3a, spartan3adsp, aspartan3, aspartan3e, aspartan3a, aspartan3adsp, spartan6, virtex4, qvirtex4, qvirtex4, virtex5, virtex5fx, virtex6, virtex6cx | Yes | string |
| C_MEMSIZE | Size of BRAM(s) in bytes | 2048 - 524288 | Yes | integer |

Allowable Parameter Combinations

For architectures that support the RAMB16 primitive, the minimum value for C_MEMSIZE is 8192 (8kB) and the maximum value is 262144 (256kB).

For architectures that support the RAMB16BWE primitive, the minimum value for C_MEMSIZE is 2048 (2kB) and the maximum value is 262144 (256kB).

For architectures that support the RAMB36 primitive, the minimum value for C_MEMSIZE is 4096 (4kB) and the maximum value is 524288 (512kB).

Parameter - Port Dependencies

The width of many of the BRAM Block signals depends on the number of memories in the system and the width of the various data and address buses. The dependencies between the BRAM design parameters and I/O signals are shown in [Table 3](#).

Table 3: Parameter-Port Dependencies

| Name | Affects | Depends | Relationship Description |
|--------------------------|-------------------------------|--|---|
| Design Parameters | | | |
| C_PORT_DWIDTH | BRAM_Din_A/B BRAM_Dout_A/B | 0 to C_PORT_DWIDTH -1 0 to C_PORT_DWIDTH -1 | Width of BRAM interface Width of BRAM interface |
| C_PORT_AWIDTH | BRAM_Addr_A/B | 0 to C_PORT_AWIDTH -1 | Width of the BRAM Address Bus |
| C_NUM_WE | BRAM_WEN_A/B | 0 to C_NUM_WE -1 | Number of byte enable signals |
| I/O Signals | | | |
| BRAM_Addr_A | | C_PORT_AWIDTH | Width varies with the width of the BRAM Address Bus |
| BRAM_Din_A | | C_PORT_DWIDTH | Width varies with the width of the BRAM Data Bus |
| BRAM_Dout_A | | C_PORT_DWIDTH | Width varies with the width of the BRAM Data Bus |
| BRAM_WEN_A | | C_NUM_WE | Width varies with the number of byte-write enable signals |
| BRAM_Addr_B | | C_PORT_AWIDTH | Width varies with the width of the BRAM Address Bus. |
| BRAM_Din_B | | C_PORT_DWIDTH | Width varies with the width of the BRAM Data Bus. |
| BRAM_Dout_B | | C_PORT_DWIDTH | Width varies with the width of the BRAM Data Bus. |
| BRAM_WEN_B | | C_NUM_WE | Width varies with the number of byte-write enable signals |

Design Implementation

Design Tools

The BRAM Block design is generated by the EDK tools.

XST is the synthesis tool used for synthesizing the BRAM Block. The EDIF netlist output from XST is then input to the Xilinx Alliance tool suite for actual device implementation.

The BRAM Block is a structural design that instantiates a number of RAMB primitives. The number of block RAM primitives instantiated in a BRAM Block depends on the following factors

- Each primitive is either 4kb or 16kb depending on architecture. The address range (C_PORT_AWIDTH) times the accessed data width (C_PORT_DWIDTH) defines the total number of bits required.
- Instantiated RAMB primitives can be configured to have at most a 32 bit wide data interface. Thus a 64 bit interface will require at least 2 BRAM primitives in parallel.
- RAMB primitive in architectures prior to Virtex-4 only have a single write enable per port. Thus if byte-write enable is required on a 32 bit data port (C_NUM_WE=4), these architectures will use a minimum of 4 BRAM primitives.

Target Technology

The target technology is an FPGA listed in the [Supported Device Family](#) field of the LogiCORE Facts table.

Device Utilization and Performance Benchmarks

The device utilization depends on the configured BRAM Block size in relation to the RAMB primitive resources of the targeted device. See the user guide for the respective FPGA family for details on RAMB primitive performance and available resources.

Supported BRAM Memory Configurations

Typical BRAM memory configurations that are supported by this BRAM Controller are: For Virtex-4, Spartan-3, and Spartan-6 devices, see [Table 4](#). For Virtex-5 and Virtex-6 devices, see [Table 5](#). **The BRAM instantiations are not provided by this core. They are part of the bram_block module generated by the EDK XPS tools during embedded system creation.**

Table 4: Supported BRAM Memory sizes for Virtex-4, Spartan-3, and Spartan-6 FPGAs

| Native Data Width Size (bits) | Supported Memory Sizes (Bytes) / BRAM Memory Configuration (Depth x Width) | Number of BRAM primitives (18Kbit ea.) required | Number of BRAM_Addr bits required | Typical BRAM_Addr(0:31) bit usage for BRAM width |
|-------------------------------|--|---|-----------------------------------|--|
| C_PORT_DWIDTH = 32 | | | | |
| 32 | 8K / (2,048x32) | 4 ⁽¹⁾ | 11 | BRAM_Addr(17:29) |
| 32 | 16K / (4,096x32) | 8 | 12 | BRAM_Addr(16:29) |
| 32 | 32K / (8,192x32) | 16 | 13 | BRAM_Addr(15:29) |
| 32 | 64K / (16,384x32) | 32 | 14 | BRAM_Addr(14:29) |
| C_PORT_DWIDTH = 64 | | | | |
| 64 | 16K / (2,048x64) | 8 ⁽²⁾ | 11 | BRAM_Addr(18:28) |
| 64 | 32K / (4,096x64) | 16 | 12 | BRAM_Addr(17:28) |
| 64 | 64K / (8,192x64) | 32 | 13 | BRAM_Addr(16:28) |
| 64 | 128K / (16,384x64) | 64 | 14 | BRAM_Addr(15:28) |
| C_PORT_DWIDTH = 128 | | | | |
| 128 | 32K / (2,048x128) | 16 ⁽³⁾ | 11 | BRAM_Addr(16:27) |
| 128 | 64K / (4,096x128) | 32 | 12 | BRAM_Addr(15:27) |
| 128 | 128K / (8,192x128) | 64 | 13 | BRAM_Addr(14:27) |
| 128 | 256K / (16,384x128) | 128 | 14 | BRAM_Addr(13:27) |

Notes:

1. A minimum of 4 BRAM primitives are required to maintain byte write capability for a 32-bit native data width BRAM array.
2. A minimum of 8 BRAM primitives are required to maintain byte write capability for a 64-bit native data width BRAM array.
3. A minimum of 16 BRAM primitives are required to maintain byte write capability for a 128-bit native data width BRAM array.

Table 5: Supported BRAM Memory sizes for Virtex-5 and Virtex-6 FPGAs

| Native Data Width Size (bits) | Supported Memory Sizes (Bytes) / BRAM Memory Configuration (Depth x Width) | Number of BRAM primitives (36Kbit ea.) required | Number of BRAM_Addr bits required | Typical BRAM_Addr(0:31) bit usage for 64-bit wide Memory (8 byte lanes) |
|-------------------------------|--|---|-----------------------------------|---|
| C_NATIVE_DWIDTH = 32 | | | | |
| 32 | 4K / (1,024x32) | 1 ⁽¹⁾ | 10 | BRAM_Addr(20:29) |
| 32 | 8K / (2,048x32) | 2 | 11 | BRAM_Addr(19:29) |
| 32 | 16K / (4,096x32) | 4 | 12 | BRAM_Addr(18:29) |
| 32 | 32K / (8,192x32) | 8 | 13 | BRAM_Addr(17:29) |
| 32 | 64K / (16,384x32) | 16 | 14 | BRAM_Addr(16:29) |
| 32 | 128K / (32,768x32) | 32 | 15 | BRAM_Addr(15:29) |
| 32 | 256K / (65,536x32) | 64 | 16 | BRAM_Addr(14:29) |
| C_NATIVE_DWIDTH = 64 | | | | |
| 64 | 8K / (1,024x64) | 2 ⁽¹⁾ | 10 | BRAM_Addr(19:28) |
| 64 | 16K / (2,048x64) | 4 | 11 | BRAM_Addr(18:28) |
| 64 | 32K / (4,096x64) | 8 | 12 | BRAM_Addr(17:28) |
| 64 | 64K / (8,192x64) | 16 | 13 | BRAM_Addr(16:28) |
| 64 | 128K / (16,384x64) | 32 | 14 | BRAM_Addr(15:28) |
| 64 | 256K / (32,768x64) | 64 | 15 | BRAM_Addr(14:28) |
| 64 | 512K / (65,536x64) | 128 | 16 | BRAM_Addr(13:28) |
| C_NATIVE_DWIDTH = 128 | | | | |
| 128 | 16K / (1,024x128) | 4 ⁽¹⁾ | 10 | BRAM_Addr(18:27) |
| 128 | 32K / (2,048x128) | 8 | 11 | BRAM_Addr(17:27) |
| 128 | 64K / (4,096x128) | 16 | 12 | BRAM_Addr(16:27) |
| 128 | 128K / (8,192x128) | 32 | 13 | BRAM_Addr(15:27) |
| 128 | 256K / (16,384x128) | 64 | 14 | BRAM_Addr(14:27) |
| 128 | 512K / (32,768x128) | 128 | 15 | BRAM_Addr(13:27) |
| 128 | 1024K / (65,536x128) | 256 | 16 | BRAM_Addr(12:27) |

Notes:

1. Virtex-5 and Virtex-6 BRAM primitives have up to 4 byte enables per primitive.

Specification Exceptions

Not applicable.

Reference Documents

None

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Revision History

The following table shows the revision history for this document:

| Date | Version | Description of Revisions |
|----------|---------|--|
| 01/07/03 | 1.3 | Update for EDK SP3 |
| 07/22/03 | 1.4 | Update to new template |
| 08/03/04 | 1.5 | Update with new families and added detail on BRAM primitive usage. |
| 8/13/04 | 1.5.1 | Updated for Gmm: reviewed and corrected trademarks. |
| 4/2/05 | 1.6 | Updated for EDK 7.1.1 SP1; updated supported device family listing. |
| 8/1/05 | 1.7 | Converted to new DS template; no content edited. |
| 12/1/05 | 1.8 | Added Spartan®-3E to supported device listing. |
| 8/21/06 | 1.9 | C_MEMSIZE description corrected |
| 4/24/09 | 2.0 | Replaced references to supported device families and tool name(s) with hyperlink to PDF file; converted to current DS template. |
| 12/2/09 | 2.1 | Listed supported devices families in LogiCORE Table; added Spartan-6 and Virtex-6 support, converted to new DS template. |
| 3/2/10 | 2.2 | Incorporated CR480244; added " Supported BRAM Memory Configurations " section, Table 4 , and Table 5 . |
| 3/1/11 | 2.3 | Corrected errors in Revision History Table; no technical content changed. |

Notice of Disclaimer

Xilinx is providing this product documentation, hereinafter "Information," to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.