Xilinx disclose this user guide, manual, release note, and/or specification (the “Documentation”) to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation 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. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU “AS-IS” WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© Copyright 2002-2011 Xilinx Inc. All Rights Reserved. XILINX, the Xilinx logo, the Brand Window and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.
Chapter 1

Introduction

This HDL guide is part of the ISE® documentation collection. A separate version of this guide is available if you prefer to work with schematics.

This guide contains the following:

• Introduction.
• Descriptions of each available macro.
• A list of design elements supported in this architecture, organized by functional categories.
• Descriptions of each available primitive.

About Design Elements

This version of the Libraries Guide describes the valid design elements for this architecture, and includes examples of instantiation code for each element. Instantiation templates are also supplied in a separate ZIP file, which you can find in your installation directory under ISE/doc/usenglish/isehelp.

Design elements are divided into three main categories:

• **Macros** - These elements are in the UniMacro library in the Xilinx tool, and are used to instantiate primitives that are complex to instantiate by just using the primitives. The synthesis tools will automatically expand the unimacros to their underlying primitives.

• **Primitives** - Xilinx components that are native to the FPGA you are targeting. If you instantiate a primitive in your design, after the translation process (ngdbuild) you will end up with the exact same component in the back end. For example, if you instantiate the Virtex®-5 element known as ISERDES_NODELAY as a user primitive, after you run translate (ngdbuild) you will end up with an ISERDES_NODELAY in the back end as well. If you were using ISERDES in a Virtex-5 device, then this will automatically retarget to an ISERDES_NODELAY for Virtex-5 in the back end. Hence, this concept of a “primitive” differs from other uses of that term in this technology.

CORE Generator maintains software libraries with hundreds of functional design elements (UniMacros and primitives) for different device architectures. New functional elements are assembled with each release of development system software. In addition to a comprehensive Unified Library containing all design elements, this guide is one in a series of architecture-specific libraries.
Design Entry Methods

For each design element in this guide, Xilinx evaluates four options for using the design element, and recommends what we believe is the best solution for you. The four options are:

- **Instantiation** - This component can be instantiated directly into the design. This method is useful if you want to control the exact placement of the individual blocks.

- **Inference** - This component can be inferred by most supported synthesis tools. You should use this method if you want to have complete flexibility and portability of the code to multiple architectures. Inference also gives the tools the ability to optimize for performance, area, or power, as specified by the user to the synthesis tool.

- **Coregen & Wizards** - This component can be used through CORE Generator or other Wizards. You should use this method if you want to build large blocks of any FPGA primitive that cannot be inferred. When using this flow, you will have to re-generate your cores for each architecture that you are targeting.

- **Macro Support** - This component has a UniMacro that can be used. These components are in the UniMacro library in the Xilinx tool, and are used to instantiate primitives that are too complex to instantiate by just using the primitives. The synthesis tools will automatically expand UniMacros to their underlying primitives.
This section describes the unimacros that can be used with this architecture. The unimacros are organized alphabetically.

The following information is provided for each unimacro, where applicable:

- Name of element
- Brief description
- Schematic symbol
- Logic table (if any)
- Port descriptions
- Design Entry Method
- Available attributes
- Example instantiation code
- For more information
**BRAM_SDP_MACRO**

Macro: Simple Dual Port RAM

**Introduction**

FPGA devices contain several block RAM memories that can be configured as general-purpose 18Kb or 9Kb RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. Both read and write operations are fully synchronous to the supplied clock(s) of the component. However, read and write ports can operate fully independently and asynchronously to each other, accessing the same memory array. Byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM.

**Note** This element, must be configured so that read and write ports have the same width.

**Port Description**

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width (Bits)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>Output</td>
<td>See Configuration Table</td>
<td>Data output bus addressed by RDADDR.</td>
</tr>
<tr>
<td><strong>Input Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td>Input</td>
<td>See Configuration Table</td>
<td>Data input bus addressed by WRADDR.</td>
</tr>
<tr>
<td>WRADDR, RDADDR</td>
<td>Input</td>
<td>See Configuration Table</td>
<td>Write/Read address input buses.</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>See Configuration Table</td>
<td>Byte-Wide Write enable.</td>
</tr>
<tr>
<td>WREN, RDEN</td>
<td>Input</td>
<td>1</td>
<td>Write/Read enable</td>
</tr>
</tbody>
</table>
## Configuration Table

<table>
<thead>
<tr>
<th>DATA_WIDTH</th>
<th>BRAM_SIZE</th>
<th>ADDR</th>
<th>WE</th>
</tr>
</thead>
<tbody>
<tr>
<td>36 - 19</td>
<td>18Kb</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>9Kb</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>18 - 10</td>
<td>18Kb</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>9Kb</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>9 - 5</td>
<td>18Kb</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9Kb</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4 - 3</td>
<td>18Kb</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9Kb</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18Kb</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9Kb</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18Kb</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9Kb</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

## Design Entry Method

This unimacro can be instantiated only. It is a parameterizable version of the primitive. Consult the Configuration Table above to correctly configure it to meet your design needs.

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Available support</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

## Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAM_SIZE</td>
<td>String</td>
<td>18Kb, 9Kb</td>
<td>9Kb</td>
<td>Configures RAM as 18Kb or 9Kb memory.</td>
</tr>
<tr>
<td>DO_REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>A value of 1 enables to the output registers to the RAM enabling quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read in one clock cycle but will have slower clock to out timing.</td>
</tr>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 72-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial value on the output after configuration.</td>
</tr>
</tbody>
</table>
### Chapter 2: About Unimacros

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_WIDTH, WRITE_WIDTH</td>
<td>Integer</td>
<td>1-72</td>
<td>36</td>
<td>Specifies size of DI/DO bus. READ_WIDTH and WRITE_WIDTH must be equal.</td>
</tr>
<tr>
<td>INIT_FILE</td>
<td>String</td>
<td>String representing file name and location</td>
<td>None</td>
<td>Name of the file containing initial values.</td>
</tr>
<tr>
<td>SIM_COLLISION_CHECK</td>
<td>String</td>
<td>ALL, &quot;WARNING_ONLY&quot;, &quot;GENERATE_X_ONLY&quot;, &quot;NONE&quot;</td>
<td>ALL</td>
<td>Allows modification of the simulation behavior if a memory collision occurs. The output is affected as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;ALL&quot; - Warning produced and affected outputs/memory location go unknown (X).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;WARNING_ONLY&quot; - Warning produced and affected outputs/memory retain last value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;GENERATE_X_ONLY&quot; - No warning. However, affected outputs/memory go unknown (X).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;NONE&quot; - No warning and affected outputs/memory retain last value.</td>
</tr>
<tr>
<td>Note</td>
<td></td>
<td></td>
<td></td>
<td>Setting this to a value other than &quot;ALL&quot; can allow problems in the design go unnoticed during simulation. Care should be taken when changing the value of this attribute. Please see the Synthesis and Simulation Design Guide for more information.</td>
</tr>
<tr>
<td>SIM_MODE</td>
<td>String</td>
<td>&quot;SAFE&quot;, &quot;FAST&quot;</td>
<td>&quot;SAFE&quot;</td>
<td>This is a simulation only attribute. It will direct the simulation model to run in performance-oriented mode when set to &quot;FAST.&quot; Please see the Synthesis and Simulation Design Guide for more information.</td>
</tr>
<tr>
<td>SRVAL</td>
<td>Hexadecimal</td>
<td>Any 72-Bit Value</td>
<td>All zeroes</td>
<td>Specifies the output value of on the DO port upon the assertion of the synchronous reset (RST) signal.</td>
</tr>
<tr>
<td>INIT_00 to INIT_7F</td>
<td>Hexadecimal</td>
<td>Any 256-Bit Value</td>
<td>All zeroes</td>
<td>Allows specification of the initial contents of the 16Kb or 32Kb data memory array.</td>
</tr>
<tr>
<td>INITP_00 to INITP_0F</td>
<td>Hexadecimal</td>
<td>Any 256-Bit Value</td>
<td>All zeroes</td>
<td>Allows specification of the initial contents of the 2Kb or 4Kb parity data memory array.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- BRAM_SDP_MACRO: Simple Dual Port RAM
```

---

**Spartan-6 Libraries Guide for HDL Designs**

8  
[www.xilinx.com](http://www.xilinx.com)  
UG615 (v 13.3) October 26, 2011
Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

-- Note - This Unimacro model assumes the port directions to be "downto".
-- Simulation of this model with "to" in the port directions could lead to erroneous results.

BRAM_SDP_MACRO_inst : BRAM_SDP_MACRO

generic map (
BRAM_SIZE => "18Kb", -- Target BRAM, "9Kb" or "18Kb"
DEVICE => "SPARTAN6", -- Target device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
WRITE_WIDTH => 0, -- Valid values are 1-36
READ_WIDTH => 0, -- Valid values are 1-36
DO_REGS => 0, -- Optional output register (0 or 1)
INIT_FILE => "NONE", -- Optional output register (0 or 1)
SIM_COLLISION_CHECK => "ALL", -- Collision check enable "ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE"
SRVAL => X"00000000000000000000000000000000", -- Set/Reset value for port output
INIT => X"00000000000000000000000000000000", -- Initial values on output port

-- The following INIT_xx declarations specify the initial contents of the RAM
INIT_00 => X"00000000000000000000000000000000",
INIT_01 => X"00000000000000000000000000000000",
INIT_02 => X"00000000000000000000000000000000",
INIT_03 => X"00000000000000000000000000000000",
INIT_04 => X"00000000000000000000000000000000",
INIT_05 => X"00000000000000000000000000000000",
INIT_06 => X"00000000000000000000000000000000",
INIT_07 => X"00000000000000000000000000000000",
INIT_08 => X"00000000000000000000000000000000",
INIT_09 => X"00000000000000000000000000000000",
INIT_1A => X"00000000000000000000000000000000",
INIT_1B => X"00000000000000000000000000000000",
INIT_1C => X"00000000000000000000000000000000",
INIT_1D => X"00000000000000000000000000000000",
INIT_1E => X"00000000000000000000000000000000",
INIT_1F => X"00000000000000000000000000000000",

-- The next set of INIT_xx are for "18Kb" configuration only
INIT_20 => X"00000000000000000000000000000000",
INIT_21 => X"00000000000000000000000000000000",
Chapter 2: About Unimacros

--- The next set of INITP_xx are for the parity bits
INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",

--- The next set of INITP_xx are for "18Kb" configuration only
INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",

port map (DO => DO, -- Output read data port, width defined by READ_WIDTH parameter
DI => DI, -- Input write data port, width defined by WRITE_WIDTH parameter
RDADDR => RDADDR, -- Input read address, width defined by read port depth
RDCLK => RDCLK, -- 1-bit input read clock
RDEN => RDEN, -- 1-bit input read port enable
REGCE => REGCE, -- 1-bit input read output register enable
RST => RST, -- 1-bit input reset
WE => WE, -- Input write enable, width defined by write port depth
WRADDR => WRADDR, -- Input write address, width defined by write port depth
WRCLK => WRCLK, -- 1-bit input write clock
WREN => WREN -- 1-bit input write port enable
);

-- End of BRAM_SDP_MACRO_inst instantiation

Verilog Instantiation Template

// BRAM_SDP_MACRO: Simple Dual Port RAM
// Spartan-6

+---------------------------------------------------------------------+
<table>
<thead>
<tr>
<th>READ_WIDTH</th>
<th>BRAM_SIZE</th>
<th>READ Depth</th>
<th>RDADDR Width</th>
<th>WRITE_WIDTH</th>
<th>WRADDR Width</th>
<th>WE Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-36</td>
<td>&quot;18Kb&quot;</td>
<td>512</td>
<td>9-bit</td>
<td>4-bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-18</td>
<td>&quot;9Kb&quot;</td>
<td>512</td>
<td>9-bit</td>
<td>2-bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-18</td>
<td>&quot;18Kb&quot;</td>
<td>1024</td>
<td>10-bit</td>
<td>2-bit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
+---------------------------------------------------------------------+
BRAM_SOP_MACRO #
.BRAM_SIZE("18Kb"), // Target BRAM, "9Kb" or "18Kb"
.DEVICE("SPARTAN6"), // Target device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
.WRITE_WIDTH(0), // Valid values are 1-36
.READ_WIDTH(0), // Valid values are 1-36
.DO_REG(0), // Optional output register (0 or 1)
.INIT_FILE ("NONE"),
.SIM_COLLISION_CHECK ("ALL"), // Collision check enable "ALL", "WARNING_ONLY",
"GENERATE_X_ONLY" or "NONE"
SRVAL(72'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_1A(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_1B(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_1C(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_1D(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_1E(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_1F(256'h0000000000000000000000000000000000000000000000000000000000000000),

// The next set of INIT_xx are for "18Kb" configuration only
.INIT_20(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_21(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_22(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_23(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_24(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_25(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_26(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_27(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_28(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_29(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_2A(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_2B(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_2C(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_2D(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_2E(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_2F(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_30(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_31(256'h0000000000000000000000000000000000000000000000000000000000000000),

Chapter 2: About Unimacros

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
**BRAM_SINGLE_MACRO**

**Macro: Single Port RAM**

**Introduction**

FPGA devices contain several block RAM memories that can be configured as general-purpose 18Kb or 9Kb RAM/ROM memories. These single-port, block RAM memories offer fast and flexible storage of large amounts of on-chip data. Byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM.

**Port Description**

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>Output</td>
<td>See Configuration Table below.</td>
<td>Data output bus addressed by ADDR.</td>
</tr>
<tr>
<td><strong>Input Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td>Input</td>
<td>See Configuration Table below.</td>
<td>Data input bus addressed by ADDR.</td>
</tr>
<tr>
<td>ADDR</td>
<td>Input</td>
<td>See Configuration Table below.</td>
<td>Address input bus.</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>See Configuration Table below.</td>
<td>Byte-Wide Write enable.</td>
</tr>
<tr>
<td>EN</td>
<td>Input</td>
<td>1</td>
<td>Write/Read enables.</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Output registers synchronous reset.</td>
</tr>
<tr>
<td>REGCE</td>
<td>Input</td>
<td>1</td>
<td>Output register clock enable input (valid only when DO_REG=1)</td>
</tr>
</tbody>
</table>
Chapter 2: About Unimacros

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock input.</td>
</tr>
</tbody>
</table>

Configuration Table

<table>
<thead>
<tr>
<th>WRITE_WIDTH</th>
<th>BRAM_SIZE</th>
<th>ADDR</th>
<th>WE</th>
</tr>
</thead>
<tbody>
<tr>
<td>37 - 72</td>
<td>18Kb</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>36 - 19</td>
<td></td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>18 - 10</td>
<td></td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>9 - 5</td>
<td></td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>4 - 3</td>
<td></td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>36 - 19</td>
<td>9Kb</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>18-10</td>
<td></td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>9 - 5</td>
<td></td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>4 - 3</td>
<td></td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>13</td>
<td>1</td>
</tr>
</tbody>
</table>

Design Entry Method

This unimacro can be instantiated only. It is a parameterizable version of the primitive. Consult the above Configuration Table in correctly configuring this element to meet your design needs.

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAM_SIZE</td>
<td>String</td>
<td>18Kb, 9Kb</td>
<td>9Kb</td>
<td>Configures RAM as 18Kb or 9Kb memory.</td>
</tr>
<tr>
<td>DO_REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>A value of 1 enables the output registers to the RAM enabling quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read in one clock cycle but will have slower clock to out timing.</td>
</tr>
<tr>
<td>READ_WIDTH</td>
<td>Integer</td>
<td>1 - 36</td>
<td>1</td>
<td>Specifies size of output bus.</td>
</tr>
<tr>
<td>WRITE_WIDTH</td>
<td>Integer</td>
<td>1 - 36</td>
<td>1</td>
<td>Specifies size of input bus.</td>
</tr>
<tr>
<td>INIT_FILE</td>
<td>String</td>
<td>String representing file name and location</td>
<td>NONE</td>
<td>Name of the file containing initial values.</td>
</tr>
</tbody>
</table>
### Attribute Value Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITE_MODE</td>
<td>String</td>
<td>READ_FIRST, WRITE_FIRST, NO_CHANGE</td>
<td>WRITE_FIRST</td>
<td>Specifies write mode to the memory</td>
</tr>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 72-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial value on the output after configuration.</td>
</tr>
<tr>
<td>SRVAL</td>
<td>Hexadecimal</td>
<td>Any 72-Bit Value</td>
<td>All zeros</td>
<td>Specifies the output value of on the DO port upon the assertion of the synchronous reset (RST) signal.</td>
</tr>
<tr>
<td>SIM_MODE</td>
<td>String</td>
<td>&quot;SAFE&quot;, &quot;FAST&quot;</td>
<td>&quot;SAFE&quot;</td>
<td>This is a simulation only attribute. It will direct the simulation model to run in performance-oriented mode when set to &quot;FAST.&quot; Please see the Synthesis and Simulation Design Guide for more information.</td>
</tr>
<tr>
<td>INIT_00 to INIT_FF</td>
<td>Hexadecimal</td>
<td>Any 256-Bit Value</td>
<td>All zeros</td>
<td>Allows specification of the initial contents of the 16Kb or 32Kb data memory array.</td>
</tr>
<tr>
<td>INTP_00 to INTP_0F</td>
<td>Hexadecimal</td>
<td>Any 256-Bit Value</td>
<td>All zeros</td>
<td>Allows specification of the initial contents of the 2Kb or 4Kb parity data memory array.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- BRAM_SINGLE_MACRO: Single Port RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
-- Note - This Unimacro model assumes the port directions to be "downto".
-- Simulation of this model with "to" in the port directions could lead to erroneous results.

-- READ_WIDTH BRAM_SIZE READ Depth WRITE Width ADDR Width WE Width
-- 19-36 "18Kb" 512 9-bit 4-bit
-- 10-18 "18Kb" 1024 10-bit 2-bit
-- 10-18 "9Kb" 512 9-bit 2-bit
-- 5-9 "18Kb" 2048 11-bit 1-bit
-- 3-4 "18Kb" 4096 12-bit 1-bit
-- 3-4 "9Kb" 2048 11-bit 1-bit
-- 2 "18Kb" 8192 13-bit 1-bit
-- 2 "9Kb" 4096 12-bit 1-bit
-- 1 "18Kb" 16384 14-bit 1-bit
-- 1 "9Kb" 8192 13-bit 1-bit

BRAM_SINGLE_MACRO_inst : BRAM_SINGLE_MACRO

generic map (  
    BRAM_SIZE => "18Kb", -- Target BRAM, "9Kb" or "18Kb"  
    DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"  
    DO_REG => 0, -- Optional output register (0 or 1)  
    INIT => X"0000000000", -- Initial values on output port  
    INIT_FILE => "NONE",  
    WRITE_WIDTH => 0, -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="18Kb")  
    READ_WIDTH => 0, -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="18Kb")  
    SRVAL => X"0000000000", -- Set/Reset value for port output  
    WRITE_MODE => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NOCHANGE"  
    -- The following INIT_xx declarations specify the initial contents of the RAM  
);
Chapter 2: About Unimacros

-- The next set of INIT_xx are for "18Kb" configuration only
INIT_00 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_01 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_02 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_03 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_04 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_05 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_06 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_07 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_08 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_09 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0A -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0B -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0C -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0D -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0E -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0F -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_10 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_11 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_12 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_13 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_14 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_15 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_16 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_17 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_18 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_19 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1A -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1B -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1C -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1D -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1E -> X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1F -> X"0000000000000000000000000000000000000000000000000000000000000000",

-- The next set of INITP_xx are for the parity bits
INITP_00 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_01 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_02 -> X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_03 -> X"0000000000000000000000000000000000000000000000000000000000000000"
-- The next set of INITP_xx are for "18Kb" configuration only
INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000"

port map (  
  DO => DO, -- Output data, width defined by READ_WIDTH parameter  
  ADDR => ADDR, -- Input address, width defined by read/write port depth  
  CLK => CLK, -- 1-bit input clock  
  DI => DI, -- Input data port, width defined by WRITE_WIDTH parameter  
  EN => EN, -- 1-bit input RAM enable  
  REGCE => REGCE, -- 1-bit input output register enable  
  RST => RST, -- 1-bit input reset  
  WE => WE -- Input write enable, width defined by write port depth  
);

-- End of BRAM_SINGLE_MACRO_inst instantiation

Verilog Instantiation Template

// BRAM_SINGLE_MACRO: Single Port RAM
// Spartan-6

//-------------------------------------------------------------------------------
// READ_WIDTH  |  WRITE_WIDTH  |  BRAM_SIZE  |  READ Depth  |  ADDR Width  |  WRITE Depth  |  WE Width  \
//-------------------------------------------------------------------------------
//  19-36       |  512         |  9-bit     |  4-bit       |             |             |           \
//  10-18      |  1024        |  10-bit    |  2-bit       |             |             |           \
//  10-18      |  512         |  9-bit     |  2-bit       |             |             |           \
//  5-9        |  2048        |  11-bit    |  1-bit       |             |             |           \
//  5-9        |  1024        |  10-bit    |  1-bit       |             |             |           \
//  3-4        |  4096        |  12-bit    |  1-bit       |             |             |           \
//  3-4        |  2048        |  11-bit    |  1-bit       |             |             |           \
//  2          |  8192        |  13-bit    |  1-bit       |             |             |           \
//  2          |  4096        |  12-bit    |  1-bit       |             |             |           \
//  1          |  16384       |  14-bit    |  1-bit       |             |             |           \
//  1          |  8192        |  13-bit    |  1-bit       |             |             |           \
//-------------------------------------------------------------------------------

BRAM_SINGLE_MACRO #(  
  .BRAM_SIZE("18Kb"), // Target BRAM, "9Kb" or "18Kb"  
  .DEVICE("SPARTAN6"), // Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"  
  .DO_REG(0), // Optional output register (0 or 1)  
  .INIT(36'h0000000000), // Initial values on output port  
  .INIT_FILE ("NONE")/,  
  .WRITE_WIDTH(0), // Valid values are 1-36 (19-36 only valid when BRAM_SIZE="18Kb")  
  .READ_WIDTH(0), // Valid values are 1-36 (19-36 only valid when BRAM_SIZE="18Kb")  
  .SRVAL(36'h0000000000), // Set/Reset value for port output  
  .WRITE_MODE("WRITE_FIRST"), // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"  
  .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),  
  .INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000)  
);
Chapter 2: About Unimacros

About writing HDL or UG615

```vhdl
 BRAM_SINGLE_MACRO_inst (
   .DO (DO),  // Output data, width defined by READ_WIDTH parameter
   .ADDR (ADDR),  // Input address, width defined by read/write port depth
   .CLK (CLK),  // 1-bit input clock
   .DI (DI),  // Input data port, width defined by WRITE_WIDTH parameter
   .EN (EN),  // 1-bit input RAM enable
   .REGCE (REGCE),  // 1-bit input output register enable
   .RST (RST),  // 1-bit input reset
   .WE (WE)  // Input write enable, width defined by write port depth
 );
```

// End of BRAM_SINGLE_MACRO_inst instantiation
For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
Chapter 2: About Unimacros

**BRAM_TDP_MACRO**

Macro: True Dual Port RAM

Introduction

FPGA devices contain several block RAM memories that can be configured as general-purpose 18kb or 9kb RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. Both read and write operations are fully synchronous to the supplied clock(s) of the component. However, READ and WRITE ports can operate fully independently and asynchronous to each other, accessing the same memory array. Byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM.

Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data output bus addressed by ADDRA.</td>
</tr>
<tr>
<td>DOA</td>
<td>Output</td>
<td>See Configuration Table below.</td>
<td>Data output bus addressed by ADDRA.</td>
</tr>
<tr>
<td>DOB</td>
<td>Output</td>
<td>See Configuration Table below.</td>
<td>Data output bus addressed by ADDRB.</td>
</tr>
<tr>
<td>DIA</td>
<td>Input</td>
<td>See Configuration Table below.</td>
<td>Data input bus addressed by ADDRA.</td>
</tr>
</tbody>
</table>
### Chapter 2: About Unimacros

#### Name | Direction | Width | Function
--- | --- | --- | ---
DIB | Input | See Configuration Table below. | Data input bus addressed by ADDRB.
ADDRA, ADDRB | Input | See Configuration Table below. | Address input buses for Port A, B.
WEA, WEB | Input | See Configuration Table below. | Write enable for Port A, B.
ENA, ENB | Input | 1 | Write/Read enables for Port A, B.
RSTA, RSTB | Input | 1 | Output registers synchronous reset for Port A, B.
REGCEA, REGCEB | Input | 1 | Output register clock enable input for Port A, B (valid only when DO_REG=1)
CLKA, CLKB | Input | 1 | Write/Read clock input for Port A, B.

#### Configuration Table

<table>
<thead>
<tr>
<th>WRITE_WIDTH_A/B-DIA/DIB</th>
<th>BRAM_SIZE</th>
<th>ADDRA/B</th>
<th>WEA/B</th>
</tr>
</thead>
<tbody>
<tr>
<td>36 - 19</td>
<td>18Kb</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>18 - 10</td>
<td></td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>9 - 5</td>
<td></td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>4 - 3</td>
<td></td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>18 - 10</td>
<td>9Kb</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>9 - 5</td>
<td></td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>4 - 3</td>
<td></td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>13</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Design Entry Method

This unimacro can be instantiated only. It is a parameterizable version of the primitive. Consult the Configuration Table above to correctly configure it to meet your design needs.

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>Recommended</td>
</tr>
</tbody>
</table>
## Available Attributes

<table>
<thead>
<tr>
<th>Attribute(s)</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAM_SIZE</td>
<td>String</td>
<td>18Kb, 9Kb</td>
<td>9Kb</td>
<td>Configures RAM as 18Kb or 9Kb memory.</td>
</tr>
<tr>
<td>DO_REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>A value of 1 enables the output registers to the RAM enabling quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read in one clock cycle but will have slower clock to out timing.</td>
</tr>
<tr>
<td>INIT</td>
<td>Hexa-decimal</td>
<td>Any 72-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial value on the output after configuration.</td>
</tr>
<tr>
<td>INIT_FILE</td>
<td>String</td>
<td>String representing file name and location</td>
<td>NONE</td>
<td>Name of file containing initial values.</td>
</tr>
<tr>
<td>READ_WIDTH, WRITE_WIDTH</td>
<td>Integer</td>
<td>1 - 72</td>
<td>36</td>
<td>Specifies size of DI/DO bus. READ_WIDTH and WRITE_WIDTH must be equal.</td>
</tr>
</tbody>
</table>
| SIM_COLLISION_CHECK | String   | ALL, "WARNING_ONLY", "GENERATE_X_ONLY", "NONE" | ALL | Allows modification of the simulation behavior if a memory collision occurs. The output is affected as follows:  
  - "ALL" - Warning produced and affected outputs/memory location go unknown (X).  
  - "WARNING_ONLY" - Warning produced and affected outputs/memory retain last value.  
  - "GENERATE_X_ONLY" - No warning. However, affected outputs/memory go unknown (X).  
  - "NONE" - No warning and affected outputs/memory retain last value.  
  Note: Setting this to a value other than "ALL" can allow problems in the design go unnoticed during simulation. Care should be taken when changing the value of this attribute. Please see the *Synthesis and Simulation Design Guide* for more information. |
| SIM_MODE         | String   | "SAFE", "FAST"         | "SAFE"  | This is a simulation only attribute. It will direct the simulation model to run in performance-oriented mode when set to "FAST." Please see the *Synthesis and Simulation Design Guide* for more information. |
| SRVAL_A, SRVAL_B | Hexa-decimal | Any 72-Bit Value   | All zeroes | Specifies the output value of on the DO port upon the assertion of the synchronous reset (RST) signal. |
| INIT_00 to INIT_FF | Hexa-decimal | Any 256-Bit Value | All zeroes | Allows specification of the initial contents of the 16Kb or 32Kb data memory array.             |
| INITP_00 to INITP_GF | Hexa-decimal | Any 256-Bit Value | All zeroes | Allows specification of the initial contents of the 2Kb or 4Kb parity data memory array.           |

## VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```
-- BRAM_TDP_MACRO: True Dual Port RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

-- Note - This Unimacro model assumes the port directions to be "downto".
-- Simulation of this model with "to" in the port directions could lead to erroneous results.

<table>
<thead>
<tr>
<th>DATA_WIDTH_A/B</th>
<th>BRAM_SIZE</th>
<th>RAM Depth</th>
<th>ADDRA/B Width</th>
<th>WEA/B Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-36</td>
<td>&quot;18Kb&quot;</td>
<td>512</td>
<td>9-bit</td>
<td>4-bit</td>
</tr>
<tr>
<td>10-18</td>
<td>&quot;18Kb&quot;</td>
<td>1024</td>
<td>10-bit</td>
<td>2-bit</td>
</tr>
<tr>
<td>10-18</td>
<td>&quot;9Kb&quot;</td>
<td>512</td>
<td>9-bit</td>
<td>2-bit</td>
</tr>
<tr>
<td>5-9</td>
<td>&quot;18Kb&quot;</td>
<td>2048</td>
<td>11-bit</td>
<td>1-bit</td>
</tr>
<tr>
<td>5-9</td>
<td>&quot;9Kb&quot;</td>
<td>1024</td>
<td>10-bit</td>
<td>1-bit</td>
</tr>
<tr>
<td>3-4</td>
<td>&quot;18Kb&quot;</td>
<td>4096</td>
<td>12-bit</td>
<td>1-bit</td>
</tr>
<tr>
<td>3-4</td>
<td>&quot;9Kb&quot;</td>
<td>2048</td>
<td>11-bit</td>
<td>1-bit</td>
</tr>
<tr>
<td>2</td>
<td>&quot;18Kb&quot;</td>
<td>8192</td>
<td>13-bit</td>
<td>1-bit</td>
</tr>
<tr>
<td>2</td>
<td>&quot;9Kb&quot;</td>
<td>4096</td>
<td>12-bit</td>
<td>1-bit</td>
</tr>
<tr>
<td>1</td>
<td>&quot;18Kb&quot;</td>
<td>16384</td>
<td>14-bit</td>
<td>1-bit</td>
</tr>
<tr>
<td>1</td>
<td>&quot;9Kb&quot;</td>
<td>8192</td>
<td>12-bit</td>
<td>1-bit</td>
</tr>
</tbody>
</table>

BRAM_TDP_MACRO_inst : BRAM_TDP_MACRO
generic map {
  BRAM_SIZE => "18Kb", -- Target BRAM, "9Kb" or "18Kb"
  DEVICE => SPR6, -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
  DOA_REG => 0, -- Optional port A output register (0 or 1)
  DOB_REG => 0, -- Optional port B output register (0 or 1)
  INIT_A = X"0000000000", -- Initial values on A output port
  INIT_B = X"0000000000", -- Initial values on B output port
  INIT_FILE => "NONE",
  READ_WIDTH_A => 0, -- Valid values are 1-36
  READ_WIDTH_B => 0, -- Valid values are 1-36
  SIM_COLLISION_CHECK => "ALL", -- Collision check enable "ALL", "WARNING ONLY",
  -- "GENERATE_XONLY" or "NONE"
  SRVAL_A => X"0000000000", -- Set/Reset value for A port output
  SRVAL_B => X"0000000000", -- Set/Reset value for B port output
  WRITE_MODE_A => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"
  WRITE_MODE_B => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"
  WRITE_WIDTH_A => 0, -- Valid values are 1-36
  WRITE_WIDTH_B => 0, -- Valid values are 1-36
-- The following INIT_xx declarations specify the initial contents of the RAM
  INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",

-- The next set of INIT_xx are for "18Kb" configuration only
INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",

-- The next set of INIT_xx are for the parity bits
INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",

-- The next set of INIT_xx are for "18Kb" configuration only
INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",

port map (DOA => DOA, DOB => DOB, ADDR => ADDR, ADDRB => ADDRB, CLKA => CLKA, CLKB => CLKB, DIA => DIA, DIB => DIB, ENA => ENA, ENB => ENB, REGCEA => REGCEA, REGCEB => REGCEB, RSTA => RSTA, RSTB => RSTB, WEA => WEA, WEB => WEB);

-- End of BRAM_TDP_MACRO_inst instantiation
Verilog Instantiation Template

Verilog code for a dual-port RAM configuration.

```verilog
// BRAM_TDP_MACRO: True Dual Port RAM
// Spartan-6

// // DATA_WIDTH_A/B BRAM_SIZE RAM Depth ADDR/A/B Width WEA/B Width
// // 19-36 "18Kb" 512 9-bit 4-bit
// // 10-18 "18Kb" 1024 10-bit 2-bit
// // 1-8 "9Kb" 512 9-bit 2-bit
// // 1-4 "9Kb" 2048 12-bit 2-bit
// // 1-4 "9Kb" 2048 11-bit 1-bit
// // 2 "18Kb" 8192 14-bit 1-bit
// // 2 "9Kb" 4096 12-bit 1-bit
// // 1 "18Kb" 16384 14-bit 1-bit
// // 1 "9Kb" 8192 12-bit 1-bit

BRAM_TDP_MACRO #(
    // BRAM_SIZE("18Kb"), // Target BRAM: "9Kb" or "18Kb"
    // DEVICE("Spartan6"), // Target device: "Virtex5", "Virtex6", "Spartan6"
    // .DOB_REG(0), // Optional port B output register (0 or 1)
    .INIT_A(36'h00000000), // Initial values on port A output port
    .INIT_B(36'h00000000), // Initial values on port B output port
    .INIT_FILE("NONE"),
    .READ_WIDTH_A(0), // Valid values are 1-36
    .READ_WIDTH_B(0), // Valid values are 1-36
    .SIM_COLLISION_CHECK("ALL"), // Collision check enable "ALL", "WARNING_ONLY",
    // "GENERATE_X_ONLY" or "NONE"
    .SRVAL_A(36'h00000000), // Set/Reset value for port A output
    .SRVAL_B(36'h00000000), // Set/Reset value for port B output
    .WRITE_MODE_A("WRITE_FIRST"), // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
    .WRITE_MODE_B("WRITE_FIRST"), // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
    .WRITE_WIDTH_A(0), // Valid values are 1-36
    .WRITE_WIDTH_B(0), // Valid values are 1-36
)
```

Other configuration parameters include:
- INIT_00 to INIT_0E
- INIT_0F to INIT_00
- INIT_10 to INIT_1F
- INIT_1A to INIT_10
- INIT_1B to INIT_1F
- INIT_1C to INIT_10

These values represent initial states for the registers and RAM addresses.
Chapter 2: About Unimacros

// The next set of INIT_xx are for "18Kb" configuration only
.INIT_00(256'h00000000000000000000000000000000),
.INIT_01(256'h00000000000000000000000000000000),
.INIT_02(256'h00000000000000000000000000000000),
.INIT_03(256'h00000000000000000000000000000000),

// The next set of INITP_10 are for the parity bits
.INITP_00(256'h00000000000000000000000000000000),
.INITP_01(256'h00000000000000000000000000000000),
.INITP_02(256'h00000000000000000000000000000000),
.INITP_03(256'h00000000000000000000000000000000),

// The next set of INITP_xx are for "18Kb" configuration only
.INITP_04(256'h00000000000000000000000000000000),
.INITP_05(256'h00000000000000000000000000000000),
.INITP_06(256'h00000000000000000000000000000000),
.INITP_07(256'h00000000000000000000000000000000),
)

BRAM_TDP_MACRO_inst ()
.DO( (DOA), // Output port-A data, width defined by READ_WIDTH_A parameter
.DOB(DOB), // Output port-B data, width defined by READ_WIDTH_B parameter
.ADDRA(ADDRA), // Input port-A address, width defined by Port A depth
.ADDRB(ADDRB), // Input port-B address, width defined by Port B depth
.CLKA(CLKA), // 1-bit input port-A clock
.CLKB(CLKB), // 1-bit input port-B clock
.DIA(DIA), // Input port-A data, width defined by WRITE_WIDTH_A parameter
.DIB(DIB), // Input port-B data, width defined by WRITE_WIDTH_B parameter
.ENA(ENA), // 1-bit input port-A enable
.ENB(ENB), // 1-bit input port-B enable
.REGCEA(REGCEA), // 1-bit input port-A output register enable
.REGCEB(REGCEB), // 1-bit input port-B output register enable
.RSTA(RSTA), // 1-bit input port-A reset
.RSTB(RSTB), // 1-bit input port-B reset
.WEA(WEA), // Input port-A write enable, width defined by Port A depth
.WEB(WEB), // Input port-B write enable, width defined by Port B depth
);

// End of BRAM_TDP_MACRO_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
ADDMA
CC_MACRO

Macro: Adder/Multiplier/Accumulator

Introduction
The ADDMACC_MACRO simplifies the instantiation of the DSP48 block when used as a pre-add, multiply accumulate function. It features parameterizable input and output widths and latency that ease the integration of DSP48 block into HDL.

Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT</td>
<td>Output</td>
<td>Variable width, equals the value of the WIDTH_A attribute plus the value of the WIDTH_B attribute.</td>
<td>Primary data output.</td>
</tr>
<tr>
<td>PREADD1</td>
<td>Input</td>
<td>Variable, see WIDTH_PREADD attribute.</td>
<td>Preadder data input.</td>
</tr>
<tr>
<td>PREADD2</td>
<td>Input</td>
<td>Variable, see WIDTH_PREADD attribute.</td>
<td>Preadder data input</td>
</tr>
<tr>
<td>MULTIPLIER</td>
<td>Input</td>
<td>Variable, see WIDTH_MULTIPLIER attribute.</td>
<td>Multiplier data input</td>
</tr>
<tr>
<td>CARRYIN</td>
<td>Input</td>
<td>1</td>
<td>Carry input</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Clock enable</td>
</tr>
<tr>
<td>LOAD</td>
<td>Input</td>
<td>1</td>
<td>Load</td>
</tr>
<tr>
<td>LOAD_DATA</td>
<td>Input</td>
<td>Variable, see WIDTH_PRODUCT attribute.</td>
<td>In a DSP slice, when LOAD is asserted, loads P with A*B+LOAD_DATA.</td>
</tr>
</tbody>
</table>
### Design Entry Method

This unimacro can be instantiated only. It is a parameterizable version of the primitive.

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIDTH_PREADD</td>
<td>Integer</td>
<td>1 to 24</td>
<td>24</td>
<td>Controls the width of PREADD1 and PREADD2 inputs.</td>
</tr>
<tr>
<td>WIDTH_MULTIPLIER</td>
<td>Integer</td>
<td>1 to 18</td>
<td>18</td>
<td>Controls the width of MULTIPLIER input.</td>
</tr>
<tr>
<td>WIDTH_PRODUCT</td>
<td>Integer</td>
<td>1 to 48</td>
<td>48</td>
<td>Controls the width of MULTIPLIER output.</td>
</tr>
<tr>
<td>LATENCY</td>
<td>Integer</td>
<td>0, 1, 2, 3, 4</td>
<td>3</td>
<td>Number of pipeline registers</td>
</tr>
<tr>
<td>DEVICE</td>
<td>String</td>
<td>&quot;VIRTEX6&quot;, &quot;SPARTAN6&quot;</td>
<td>&quot;VIRTEX6&quot;</td>
<td>Target hardware architecture.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- ADDMACC_MACRO: Add and Multiple Accumulate Function implemented in a DSP48E
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

ADDMACC_MACRO_inst : ADDMACC_MACRO
generic map (  
  DEVICE => "SPARTAN6", -- Target Device: "VIRTEX6", "SPARTAN6"  
  LATENCY => 4, -- Desired clock cycle latency, 1-4  
  WIDTH_PREADD => 25, -- Pre-Adder input bus width, 1-25  
  WIDTH_MULTIPLIER => 18, -- Multiplier input bus width, 1-18  
  WIDTH_PRODUCT => 48) -- MACC output width, 1-48
port map (  
  PRODUCT => PRODUCT, -- MACC result output, width defined by WIDTH_PRODUCT generic  
  MULTIPLIER => MULTIPLIER, -- Multiplier data input, width determined by WIDTH_MULTIPLIER generic
```

---

**Chapter 2: About Unimacros**

**Name** | **Direction** | **Width** | **Function**
--- | --- | --- | ---
RST | Input | 1 | Synchronous Reset
Chapter 2: About Unimacros

Verilog Instantiation Template

// ADDMACC_MACRO: Variable width & latency - Pre-Add -> Multiplier -> Accumulate
// function implemented in a DSP48E
// Spartan-6

ADDMACC_MACRO #(
  .DEVICE("SPARTAN6"), // Target Device: "VIRTEX6", "SPARTAN6"
  .LATENCY(4), // Desired clock cycle latency, 0-4
  .WIDTH_MULTIPLIER(18), // Pre-adder input width, 1-18
  .WIDTH_PRODUCT(48) // MACC output width, 1-48
) ADDMACC_MACRO_inst ( // MACC result output, width defined by WIDTH_PRODUCT parameter
  .PRODUCT(PRODUCT), // Pre-adder data input, width determined by WIDTH_PREADDR generic
  .PREADDR1 => PREADDR1, -- Preadder data input, width determined by WIDTH_PREADDR generic
  .PREADDR2 => PREADDR2, -- Preadder data input, width determined by WIDTH_PREADDR generic
  .CARRYIN => CARRYIN, -- 1-bit carry-in input
  .CE => CE, -- 1-bit input clock enable
  .CLK => CLK, -- 1-bit clock input
  .LOAD => LOAD, -- 1-bit accumulator load input
  .LOAD_DATA => LOAD_DATA, -- Accumulator load data input, width defined by WIDTH_PRODUCT generic
  .RST => RST -- 1-bit input active high synchronous reset
);

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
**ADDSUB_MACRO**

**Macro: Adder/Subtractor**

The ADDSUB_MACRO simplifies the instantiation of the DSP48 block when used as a simple adder/subtractor. It features parameterizable input and output widths and latency that ease the integration of the DSP48 block into HDL.

**Port Description**

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width (Bits)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARRYOUT</td>
<td>Output</td>
<td>1</td>
<td>Carry Out</td>
</tr>
<tr>
<td>RESULT</td>
<td>Output</td>
<td>Variable, see WIDTH attribute.</td>
<td>Data output bus addressed by RDADDR.</td>
</tr>
<tr>
<td><strong>Input Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDSUB</td>
<td>Input</td>
<td>1</td>
<td>When high, RESULT is an addition. When low, RESULT is a subtraction.</td>
</tr>
<tr>
<td>A</td>
<td>Input</td>
<td>Variable, see WIDTH attribute.</td>
<td>Data input to add/sub.</td>
</tr>
<tr>
<td>B</td>
<td>Input</td>
<td>Variable, see WIDTH attribute.</td>
<td>Data input to add/sub</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Clock Enable</td>
</tr>
<tr>
<td>CARRYIN</td>
<td>Input</td>
<td>1</td>
<td>Carry In</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Synchronous Reset</td>
</tr>
</tbody>
</table>

**Design Entry Method**

This unimacro can be instantiated only. It is a parameterizable version of the primitive.
### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE</td>
<td>String</td>
<td>“VIRTEX6”, “SPARTAN6”</td>
<td>“VIRTEX6”</td>
<td>Target hardware architecture.</td>
</tr>
<tr>
<td>LATENCY</td>
<td>Integer</td>
<td>0, 1, 2</td>
<td>2</td>
<td>Number of pipeline registers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1 - PREG = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 2 - AREG = BREG = CREG = PREG</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Integer</td>
<td>1-48</td>
<td>48</td>
<td>A, B, and RESULT port width; internal customers can override B and RESULT port widths using other parameters</td>
</tr>
<tr>
<td>WIDTH_RESULT</td>
<td>Integer</td>
<td>1-48</td>
<td>48</td>
<td>Result port width override.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- ADDSUB_MACRO: Variable width & latency - Adder / Subtractor implemented in a DSP48E
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
ADDSUB_MACRO_inst : ADDSUB_MACRO
generic map (
  DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
  LATENCY => 2, -- Desired clock cycle latency, 0-2
  WIDTH => 48) -- Input / Output bus width, 1-48
port map (  
  CARRYOUT => CARRYOUT, -- 1-bit carry-out output signal
  RESULT => RESULT, -- Add/sub result output, width defined by WIDTH generic
  A => A, -- Input A bus, width defined by WIDTH generic
  ADD_SUB => ADD_SUB, -- 1-bit add/sub input, high selects add, low selects subtract
  B => B, -- Input B bus, width defined by WIDTH generic
  CARRYIN => CARRYIN, -- 1-bit carry-in input
  CE => CE, -- 1-bit clock enable input
  CLK => CLK, -- 1-bit clock input
  RST => RST -- 1-bit active high synchronous reset
);
-- End of ADDSUB_MACRO_inst instantiation
```
Chapter 2: About Unimacros

Verilog Instantiation Template

// ADDSUB_MACRO: Variable width & latency - Adder / Subtractor implemented in a DSP48E
// Spartan-6
ADD_SUB_MACRO #(  
  .DEVICE("SPARTAN6"), // Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
  .LATENCY(2), // Desired clock cycle latency, 0-2
  .WIDTH(48) // Input / output bus width, 1-48
) ADDSUB_MACRO_inst (  
  .CARRYOUT(CARRYOUT), // 1-bit carry-out output signal
  .RESULT(RESULT), // Add/sub result output, width defined by WIDTH parameter
  .A(A), // Input A bus, width defined by WIDTH parameter
  .ADD_SUB(ADD_SUB), // 1-bit add/sub input, high selects add, low selects subtract
  .B(B), // Input B bus, width defined by WIDTH parameter
  .CARRYIN(CARRYIN), // 1-bit carry-in input
  .CE(CE), // 1-bit clock enable input
  .CLK(CLK), // 1-bit clock input
  .RST(RST) // 1-bit active high synchronous reset
);

// End of ADDSUB_MACRO_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
COUNTER_LOAD_MACRO

Macro: Loadable Counter

Introduction

The COUNTER_LOAD_MACRO simplifies the instantiation of the DSP48 block when used as dynamic loading up/down counter. It features parameterizable output width and count by values that ease the integration of the DSP48 block into HDL.

Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Ports</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>Output</td>
<td>Variable, see WIDTH_DATA attribute.</td>
<td>Counter output.</td>
</tr>
<tr>
<td>Input Ports</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Clock Enable.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Input</td>
<td>Variable, see WIDTH_DATA attribute.</td>
<td>When asserted, loads the counter from LOAD_DATA (two-clock latency).</td>
</tr>
<tr>
<td>LOAD_DATA</td>
<td>Input</td>
<td>Variable, see WIDTH_DATA attribute.</td>
<td>In a DSP slice, asserting the LOAD pin will force this data into the P register with a latency of 2 clocks.</td>
</tr>
<tr>
<td>DIRECTION</td>
<td>Input</td>
<td>1</td>
<td>High for Up and Low for Down (two-clock latency)</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Synchronous Reset</td>
</tr>
</tbody>
</table>

Design Entry Method

This unimacro can be instantiated only. It is a parameterizable version of the primitive.
Chapter 2: About Unimacros

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE</td>
<td>String</td>
<td>&quot;VIRTEX6&quot;, &quot;SPARTAN6&quot;</td>
<td>&quot;VIRTEX6&quot;</td>
<td>Target hardware architecture.</td>
</tr>
<tr>
<td>COUNT_BY</td>
<td>Hexadecimal</td>
<td>Any 48 bit value.</td>
<td>000000000001</td>
<td>Count by n; takes precedence over WIDTH_DATA.</td>
</tr>
<tr>
<td>WIDTH_DATA</td>
<td>Integer</td>
<td>1-48</td>
<td>48</td>
<td>Specifies counter width.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- COUNTER_LOAD_MACRO: Loadable variable counter implemented in a DSP48E
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
COUNTER_LOAD_MACRO_inst : COUNTER_LOAD_MACRO
generic map (  
  COUNT_BY => X"000000000001", -- Count by value  
  DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"  
  WIDTH_DATA => 48) -- Counter output bus width, 1-48
port map (  
  Q => Q, -- Counter output, width determined by WIDTH_DATA generic  
  CLK => CLK, -- 1-bit clock input  
  CE => CE, -- 1-bit clock enable input  
  DIRECTION => DIRECTION, -- 1-bit up/down count direction input, high is count up  
  LOAD => LOAD, -- 1-bit active high load input  
  LOAD_DATA => LOAD_DATA, -- Counter load data, width determined by WIDTH_DATA generic  
  RST => RST -- 1-bit active high synchronous reset
);
-- End of COUNTER_LOAD_MACRO_inst instantiation
```

### Verilog Instantiation Template

```verilog
// COUNTER_LOAD_MACRO: Loadable variable counter implemented in a DSP48E
// Spartan-6
COUNTER_LOAD_MACRO #(  
  .COUNT_BY(4'h000000000001), -- Count by value  
  .DEVICE("SPARTAN6"), // Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"  
  .WIDTH_DATA(48) // Counter output bus width, 1-48
) COUNTER_LOAD_MACRO_inst (  
  .Q(Q), // Counter output, width determined by WIDTH_DATA parameter  
  .CLK(CLK), // 1-bit clock input  
  .CE(CE), // 1-bit clock enable input  
  .DIRECTION(DIRECTION), // 1-bit up/down count direction input, high is count up  
  .LOAD(LOAD), // 1-bit active high load input  
  .LOAD_DATA(LOAD_DATA), // Counter load data, width determined by WIDTH_DATA parameter  
  .RST(RST) // 1-bit active high synchronous reset
);
```
// End of COUNTER_LOAD_MACRO_inst instantiation

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](https://www.xilinx.com).
MACC_MACRO

Macro: Multiplier/Accumulator

Introduction

The MACC_MACRO simplifies the instantiation of the DSP48 block when used in simple signed multiplier/accumulator mode. It features parameterizable input and output widths and latencies that ease the integration of the DSP48 block into HDL.

Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>Output</td>
<td>Variable width, equals the value of the WIDTH_A attribute plus the value of the WIDTH_B attribute.</td>
<td>Primary data output.</td>
</tr>
<tr>
<td><strong>Input Ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Input</td>
<td>Variable, see WIDTH_A attribute.</td>
<td>Multiplier data input.</td>
</tr>
<tr>
<td>B</td>
<td>Input</td>
<td>Variable, see WIDTH_B attribute.</td>
<td>Multiplier data input.</td>
</tr>
<tr>
<td>CARRYIN</td>
<td>Input</td>
<td>1</td>
<td>Carry input.</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Clock enable.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Input</td>
<td>1</td>
<td>Load.</td>
</tr>
<tr>
<td>LOAD_DATA</td>
<td>Input</td>
<td>Variable width, equals the value of the WIDTH_A attribute plus the value of the WIDTH_B attribute.</td>
<td>In a DSP slice, when LOAD is asserted, loads P with A*B+LOAD_DATA.</td>
</tr>
</tbody>
</table>
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Synchronous Reset.</td>
</tr>
<tr>
<td>ADDSUB</td>
<td>Input</td>
<td>1</td>
<td>High sets accumulator in addition mode; low sets accumulator in subtraction mode.</td>
</tr>
</tbody>
</table>

### Design Entry Method

This unimacro can be instantiated only. It is a parameterizable version of the primitive.

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIDTH_A</td>
<td>Integer</td>
<td>1 to 18</td>
<td>18</td>
<td>Controls the width of A input.</td>
</tr>
<tr>
<td>WIDTH_B</td>
<td>Integer</td>
<td>1 to 18</td>
<td>18</td>
<td>Controls the width of B input.</td>
</tr>
<tr>
<td>LATENCY</td>
<td>Integer</td>
<td>0, 1, 2, 3, 4</td>
<td>3</td>
<td>Number of pipeline registers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1 - MREG == 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 2 - AREG == BREG == 1 and MREG == 1 or MREG == 1 and PREG == 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 3 - AREG == BREG == 1 and MREG == 1 and PREG == 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 4 - AREG == BREG == 2 and MREG == 1 and PREG == 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE</td>
<td>“VIRTEX5”, “VIRTEX6”, “SPARTAN6”</td>
<td>“VIRTEX6”</td>
<td>Target hardware architecture.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- MACC_MACRO: Multiple Accumulate Function implemented in a DSP48E
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

MACC_MACRO_inst : MACC_MACRO
generic map ( 
  DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
  LATENCY => 3, -- Desired clock cycle latency, 1-4
  WIDTH_A => 25, -- Multiplier A-input bus width, 1-25
  WIDTH_B => 18, -- Multiplier B-input bus width, 1-18
  WIDTH_P => 48) -- Accumulator output bus width, 1-48
port map ( 
P => P, -- MACC output bus, width determined by WIDTH_P generic
A => A, -- MACC input A bus, width determined by WIDTH_A generic
ADDSUB => ADDSUB, -- 1-bit add/sub input, high selects add, low selects subtract
B => B, -- MACC input B bus, width determined by WIDTH_B generic

Spartan-6 Libraries Guide for HDL Designs
UG615 (v 13.3) October 26, 2011 www.xilinx.com
Chapter 2: About Unimacros

CARRYIN => CARRYIN, -- 1-bit carry-in input to accumulator
CE => CE, -- 1-bit active high input clock enable
CLK => CLK, -- 1-bit positive edge clock input
LOAD => LOAD, -- 1-bit active high input load accumulator enable
LOAD_DATA => LOAD_DATA, -- Load accumulator input data,
          -- width determined by WIDTH_P generic
RST => RST -- 1-bit input active high reset

-- End of MACC_MACRO_inst instantiation

Verilog Instantiation Template

// MACC_MACRO: Multiply Accumulate Function implemented in a DSP48E
// Spartan-6

MACC_MACRO #(
    .DEVICE("SPARTAN6"), // Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
    .LATENCY(3), // Desired clock cycle latency, 1-4
    .WIDTH_A(18), // Multiplier A-input bus width, 1-18
    .WIDTH_B(18), // Multiplier B-input bus width, 1-18
    .WIDTH_P(48) // Accumulator output bus width, 1-48
) MACC_MACRO_inst(
    .P(P), // MACC output bus, width determined by WIDTH_P parameter
    .A(A), // MACC input A bus, width determined by WIDTH_A parameter
    .ADDSUB(ADDSUB), // 1-bit add/sub input, high selects add, low selects subtract
    .B(B), // MACC input B bus, width determined by WIDTH_B parameter
    .CARRYIN(CARRYIN), // 1-bit carry-in input to accumulator
    .CE(CE), // 1-bit active high input clock enable
    .CLK(CLK), // 1-bit positive edge clock input
    .LOAD(LOAD), // 1-bit active high input load accumulator enable
    .LOAD_DATA(LOAD_DATA), // Load accumulator input data, width determined by WIDTH_P parameter
    .RST(RST) // 1-bit input active high reset
);

// End of MACC_MACRO_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
MULT_MACRO

Macro: Multiplier

Introduction

The MULT_MACRO simplifies the instantiation of the DSP48 block when used as a simple signed multiplier. It features parameterizable input and output widths and latencies that ease the integration of the DSP48 block into HDL.

Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Input</td>
<td>Variable, see WIDTH_A attribute.</td>
<td>Multiplier data input.</td>
</tr>
<tr>
<td>B</td>
<td>Input</td>
<td>Variable, see WIDTH_B attribute.</td>
<td>Multiplier data input.</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Clock Enable.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock.</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Synchronous Reset.</td>
</tr>
<tr>
<td>P</td>
<td>Output</td>
<td>Variable width, equals the value of the WIDTH_A attribute plus the value of the WIDTH_B attribute.</td>
<td>Primary data output.</td>
</tr>
</tbody>
</table>

Design Entry Method

This unimacro can be instantiated only. It is a parameterizable version of the primitive.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>Recommended</td>
</tr>
</tbody>
</table>
Chapter 2: About Unimacros

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIDTH_A</td>
<td>Integer</td>
<td>1 to 18</td>
<td>18</td>
<td>Controls the width of A input.</td>
</tr>
<tr>
<td>WIDTH_B</td>
<td>Integer</td>
<td>1 to 18</td>
<td>18</td>
<td>Controls the width of B input.</td>
</tr>
</tbody>
</table>
| LATENCY   | Integer| 0, 1, 2, 3, 4  | 3       | Number of pipeline registers
  1 - MREG == 1
  2 - AREG == BREG == 1 and MREG == 1 or MREG == 1 and PREG == 1
  3 - AREG == BREG == 1 and PREG == 1
  4 - AREG == BREG == 2 and MREG == 1 and PREG == 1 |
| DEVICE    | String | “VIRTEX5”, “VIRTEX6”, “SPARTAN6” | “VIRTEX6” | Target hardware architecture. |

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- MULT_MACRO: Multiply Function implemented in a DSP48E
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
MULT_MACRO_inst : MULT_MACRO
generic map (   DEVICE => "SPARTAN6",   -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
                LATENCY => 3,   -- Desired clock cycle latency, 0-4
                WIDTH_A => 18,   -- Multiplier A-input bus width, 1-25
                WIDTH_B => 18)   -- Multiplier B-input bus width, 1-18
port map (   P => P,   -- Multiplier output bus, width determined by WIDTH_P generic
             A => A,   -- Multiplier input A bus, width determined by WIDTH_A generic
             B => B,   -- Multiplier input B bus, width determined by WIDTH_B generic
             CE => CE,   -- 1-bit active high input clock enable
             CLK => CLK,   -- 1-bit positive edge clock input
             RST => RST -- 1-bit input active high reset
);
-- End of MULT_MACRO_inst instantiation

Verilog Instantiation Template

// MULT_MACRO: Multiply Function implemented in a DSP48E
// Spartan-6
MULT_MACRO #(
  .DEVICE("SPARTAN6"), // Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
  .LATENCY(3), // Desired clock cycle latency, 0-4
  .WIDTH_A(18), // Multiplier A-input bus width, 1-18
  .WIDTH_B(18) // Multiplier B-input bus width, 1-18
) MULT_MACRO_inst (   .P(P), // Multiplier output bus, width determined by WIDTH_P parameter
                      .A(A), // Multiplier input A bus, width determined by WIDTH_A parameter
                      .B(B), // Multiplier input B bus, width determined by WIDTH_B parameter
                      .CE(CE), // 1-bit active high input clock enable
                      .CLK(CLK), // 1-bit positive edge clock input
.RST(RST) // 1-bit input active high reset
);

// End of MULTI_MACRO_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
Chapter 3

Functional Categories

This section categorizes, by function, the circuit design elements described in detail later in this guide. The elements (primitives and macros) are listed in alphanumeric order under each functional category.

<table>
<thead>
<tr>
<th>Advanced</th>
<th>Convenience Primitives</th>
<th>Shift Register LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic Functions</td>
<td>I/O Components</td>
<td>Slice/CLB Primitives</td>
</tr>
<tr>
<td>Clock Components</td>
<td>RAM/ROM</td>
<td></td>
</tr>
<tr>
<td>Config/BSCAN Components</td>
<td>Registers/Latches</td>
<td></td>
</tr>
</tbody>
</table>

**Advanced**

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCB</td>
<td>Primitive: Memory Control Block</td>
</tr>
<tr>
<td>PCIE_A1</td>
<td>Primitive: PCI Express</td>
</tr>
</tbody>
</table>

**Arithmetic Functions**

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP48A1</td>
<td>Primitive: Multi-Functional, Cascadable, 48-bit Output, Arithmetic Block</td>
</tr>
</tbody>
</table>

**Clock Components**

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFG</td>
<td>Convenience Primitive: Global Clock Buffer</td>
</tr>
<tr>
<td>BUFGCE</td>
<td>Convenience Primitive: Global Clock Buffer with Clock Enable</td>
</tr>
<tr>
<td>BUFGMUX</td>
<td>Primitive: Global Clock MUX Buffer</td>
</tr>
<tr>
<td>BUFGMUX_1</td>
<td>Primitive: Global Clock MUX Buffer with Output State 1</td>
</tr>
<tr>
<td>BUFH</td>
<td>Primitive: Clock buffer for a single clocking region</td>
</tr>
<tr>
<td>BUFIO2</td>
<td>Primitive: Dual Clock Buffer and Strobe Pulse</td>
</tr>
<tr>
<td>BUFIO2_2CLK</td>
<td>Primitive: Dual Clock Buffer and Strobe Pulse with Differential Input</td>
</tr>
</tbody>
</table>
## Chapter 3: Functional Categories

### Design Elements and Descriptions

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFIO2FB</td>
<td>Primitive: Feedback Clock Buffer</td>
</tr>
<tr>
<td>BUFPOLL</td>
<td>Primitive: PLL Buffer</td>
</tr>
<tr>
<td>BUFPOLL_MCB</td>
<td>Primitive: PLL Buffer for the Memory Controller Block</td>
</tr>
<tr>
<td>DCM_CLKGEN</td>
<td>Primitive: Digital Clock Manager</td>
</tr>
<tr>
<td>DCM_SP</td>
<td>Primitive: Digital Clock Manager</td>
</tr>
<tr>
<td>PLL_BASE</td>
<td>Primitive: Basic Phase Locked Loop Clock Circuit</td>
</tr>
</tbody>
</table>

### Config/BSCAN Components

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSCAN_SPARTAN6</td>
<td>Primitive: Spartan®-6 JTAG Boundary Scan Logic Control Circuit</td>
</tr>
<tr>
<td>DNA_PORT</td>
<td>Primitive: Device DNA Data Access Port</td>
</tr>
<tr>
<td>ICAP_SPARTAN6</td>
<td>Primitive: Internal Configuration Access Port</td>
</tr>
<tr>
<td>JTAG_SIM_SPARTAN6</td>
<td>Simulation: JTAG TAP Controller Simulation Model</td>
</tr>
<tr>
<td>POST_CRC_INTERNAL</td>
<td>Primitive: Post-configuration CRC error detection</td>
</tr>
<tr>
<td>SIM_CONFIG_S6</td>
<td>Simulation: Configuration Simulation Model</td>
</tr>
<tr>
<td>SIM_CONFIG_S6_SERIAL</td>
<td>Simulation: Serial Configuration Simulation Model</td>
</tr>
<tr>
<td>STARTUP_SPARTAN6</td>
<td>Primitive: Spartan®-6 Global Set/Reset, Global 3-State and Configuration Start-Up Clock Interface</td>
</tr>
<tr>
<td>SUSPEND_SYNC</td>
<td>Primitive: Suspend Mode Access</td>
</tr>
</tbody>
</table>

### Convenience Primitives

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFGCE_1</td>
<td>Convenience Primitive: Global Clock Buffer with Clock Enable and Output State 1</td>
</tr>
</tbody>
</table>

### I/O Components

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTPA1_DUAL</td>
<td>Primitive: Dual Gigabit Transceiver</td>
</tr>
<tr>
<td>IBUF</td>
<td>Primitive: Input Buffer</td>
</tr>
<tr>
<td>IBUFD</td>
<td>Primitive: Differential Signaling Input Buffer</td>
</tr>
<tr>
<td>IBUFD_DIFF_OUT</td>
<td>Primitive: Signaling Input Buffer with Differential Output</td>
</tr>
<tr>
<td>IBUFG</td>
<td>Primitive: Dedicated Input Clock Buffer</td>
</tr>
<tr>
<td>IBUFGDS</td>
<td>Primitive: Differential Signaling Dedicated Input Clock Buffer and Optional Delay</td>
</tr>
<tr>
<td>IBUFGDS_DIFF_OUT</td>
<td>Primitive: Differential Signaling Input Buffer with Differential Output</td>
</tr>
<tr>
<td>IOBUF</td>
<td>Primitive: Bi-Directional Buffer</td>
</tr>
<tr>
<td>IOBUFD</td>
<td>Primitive: 3-State Differential Signaling I/O Buffer with Active Low Output Enable</td>
</tr>
</tbody>
</table>
### Design Element Description

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IODELAY2</td>
<td>Primitive: Input and Output Fixed or Variable Delay Element</td>
</tr>
<tr>
<td>JODRP2</td>
<td>Primitive: I/O Control Port</td>
</tr>
<tr>
<td>JODRP2_MCB</td>
<td>Primitive: I/O Control Port for the Memory Controller Block</td>
</tr>
<tr>
<td>ISERDES2</td>
<td>Primitive: Input SERial/DESerializer.</td>
</tr>
<tr>
<td>KEEPER</td>
<td>Primitive: KEEPER Symbol</td>
</tr>
<tr>
<td>OBUF</td>
<td>Primitive: Output Buffer</td>
</tr>
<tr>
<td>OBUFDS</td>
<td>Primitive: Differential Signaling Output Buffer</td>
</tr>
<tr>
<td>OBUFT</td>
<td>Primitive: 3-State Output Buffer with Active Low Output Enable</td>
</tr>
<tr>
<td>OBUFTDS</td>
<td>Primitive: 3-State Output Buffer with Differential Signaling, Active-Low Output Enable</td>
</tr>
<tr>
<td>OSERDES2</td>
<td>Primitive: Dedicated IOB Output Serializer</td>
</tr>
<tr>
<td>PULLDOWN</td>
<td>Primitive: Resistor to GND for Input Pads, Open-Drain, and 3-State Outputs</td>
</tr>
<tr>
<td>PULLUP</td>
<td>Primitive: Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs</td>
</tr>
</tbody>
</table>

### RAM/ROM

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM128X1D</td>
<td>Primitive: 128-Deep by 1-Wide Dual Port Random Access Memory (Select RAM)</td>
</tr>
<tr>
<td>RAM256X1S</td>
<td>Primitive: 256-Deep by 1-Wide Random Access Memory (Select RAM)</td>
</tr>
<tr>
<td>RAM32M</td>
<td>Primitive: 32-Deep by 8-bit Wide Multi Port Random Access Memory (Select RAM)</td>
</tr>
<tr>
<td>RAM32X1D</td>
<td>Primitive: 32-Deep by 1-Wide Static Dual Port Synchronous RAM</td>
</tr>
<tr>
<td>RAM32X1S</td>
<td>Primitive: 32-Deep by 1-Wide Static Synchronous RAM</td>
</tr>
<tr>
<td>RAM32X1S_1</td>
<td>Primitive: 32-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock</td>
</tr>
<tr>
<td>RAM32X2S</td>
<td>Primitive: 32-Deep by 2-Wide Static Synchronous RAM</td>
</tr>
<tr>
<td>RAM64M</td>
<td>Primitive: 64-Deep by 4-bit Wide Multi Port Random Access Memory (Select RAM)</td>
</tr>
<tr>
<td>RAM64X1D</td>
<td>Primitive: 64-Deep by 1-Wide Dual Port Static Synchronous RAM</td>
</tr>
<tr>
<td>RAM64X1S</td>
<td>Primitive: 64-Deep by 1-Wide Static Synchronous RAM</td>
</tr>
<tr>
<td>RAM64X1S_1</td>
<td>Primitive: 64-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock</td>
</tr>
<tr>
<td>RAMB16BWER</td>
<td>Primitive: 16K-bit Data and 2K-bit Parity Configurable Synchronous Dual Port Block RAM with Optional Output Registers</td>
</tr>
</tbody>
</table>
### Chapter 3: Functional Categories

#### Design Element | Description
--- | ---
RAMB8BWER | Primitive: 8K-bit Data and 1K-bit Parity Configurable Synchronous Dual Port Block RAM with Optional Output Registers
ROM128X1 | Primitive: 128-Deep by 1-Wide ROM
ROM256X1 | Primitive: 256-Deep by 1-Wide ROM
ROM32X1 | Primitive: 32-Deep by 1-Wide ROM
ROM64X1 | Primitive: 64-Deep by 1-Wide ROM

### Registers/Latches

#### Design Element | Description
--- | ---
FDCE | Primitive: D Flip-Flop with Clock Enable and Asynchronous Clear
FDPE | Primitive: D Flip-Flop with Clock Enable and Asynchronous Reset
FDRE | Primitive: D Flip-Flop with Clock Enable and Synchronous Set
FDSE | Primitive: D Flip-Flop with Clock Enable and Synchronous Reset
IDDR2 | Primitive: Double Data Rate Input D Flip-Flop with Optional Data Alignment, Clock Enable and Programmable Synchronous or Asynchronous Set/Reset
LDCE | Primitive: Transparent Data Latch with Asynchronous Clear and Gate Enable
LDPE | Primitive: Transparent Data Latch with Asynchronous Preset and Gate Enable
ODDR2 | Primitive: Dual Data Rate Output D Flip-Flop with Optional Data Alignment, Clock Enable and Programmable Synchronous or Asynchronous Set/Reset

### Shift Register LUT

#### Design Element | Description
--- | ---
SRL16E | Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Clock Enable
SRLC32E | Primitive: 32 Clock Cycle, Variable Length Shift Register Look-Up Table (LUT) with Clock Enable
## Slice/CLB Primitives

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARRY4</td>
<td>Primitive: Fast Carry Logic with Look Ahead</td>
</tr>
<tr>
<td>CFGLUT5</td>
<td>Primitive: 5-input Dynamically Reconfigurable Look-Up Table (LUT)</td>
</tr>
<tr>
<td>LUT1</td>
<td>Macro: 1-Bit Look-Up Table with General Output</td>
</tr>
<tr>
<td>LUT1_D</td>
<td>Macro: 1-Bit Look-Up Table with Dual Output</td>
</tr>
<tr>
<td>LUT1_L</td>
<td>Macro: 1-Bit Look-Up Table with Local Output</td>
</tr>
<tr>
<td>LUT2</td>
<td>Macro: 2-Bit Look-Up Table with General Output</td>
</tr>
<tr>
<td>LUT2_D</td>
<td>Macro: 2-Bit Look-Up Table with Dual Output</td>
</tr>
<tr>
<td>LUT2_L</td>
<td>Macro: 2-Bit Look-Up Table with Local Output</td>
</tr>
<tr>
<td>LUT3</td>
<td>Macro: 3-Bit Look-Up Table with General Output</td>
</tr>
<tr>
<td>LUT3_D</td>
<td>Macro: 3-Bit Look-Up Table with Dual Output</td>
</tr>
<tr>
<td>LUT3_L</td>
<td>Macro: 3-Bit Look-Up Table with Local Output</td>
</tr>
<tr>
<td>LUT4</td>
<td>Macro: 4-Bit Look-Up Table with General Output</td>
</tr>
<tr>
<td>LUT4_D</td>
<td>Macro: 4-Bit Look-Up Table with Dual Output</td>
</tr>
<tr>
<td>LUT4_L</td>
<td>Macro: 4-Bit Look-Up Table with Local Output</td>
</tr>
<tr>
<td>LUT5</td>
<td>Primitive: 5-Input Lookup Table with General Output</td>
</tr>
<tr>
<td>LUT5_D</td>
<td>Primitive: 5-Input Lookup Table with General and Local Outputs</td>
</tr>
<tr>
<td>LUT5_L</td>
<td>Primitive: 5-Input Lookup Table with Local Output</td>
</tr>
<tr>
<td>LUT6</td>
<td>Primitive: 6-Input Lookup Table with General Output</td>
</tr>
<tr>
<td>LUT6_2</td>
<td>Primitive: Six-input, 2-output, Look-Up Table</td>
</tr>
<tr>
<td>LUT6_D</td>
<td>Primitive: 6-Input Lookup Table with General and Local Outputs</td>
</tr>
<tr>
<td>LUT6_L</td>
<td>Primitive: 6-Input Lookup Table with Local Output</td>
</tr>
<tr>
<td>MUXF7</td>
<td>Primitive: 2-to-1 Look-Up Table Multiplexer with General Output</td>
</tr>
<tr>
<td>MUXF7_D</td>
<td>Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output</td>
</tr>
<tr>
<td>MUXF7_L</td>
<td>Primitive: 2-to-1 Look-Up Table Multiplexer with Local Output</td>
</tr>
<tr>
<td>MUXF8</td>
<td>Primitive: 2-to-1 Look-Up Table Multiplexer with General Output</td>
</tr>
<tr>
<td>MUXF8_D</td>
<td>Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output</td>
</tr>
<tr>
<td>MUXF8_L</td>
<td>Primitive: 2-to-1 Look-Up Table Multiplexer with Local Output</td>
</tr>
</tbody>
</table>
About Design Elements

This section describes the design elements that can be used with this architecture. The design elements are organized alphabetically.

The following information is provided for each design element, where applicable:

• Name of element
• Brief description
• Schematic symbol (if any)
• Logic table (if any)
• Port descriptions
• Design Entry Method
• Available attributes (if any)
• Example instantiation code
• For more information
BSCAN_SPARTAN6

Primitive: Spartan®-6 JTAG Boundary Scan Logic Control Circuit

Introduction

This design element allows access to and from internal logic by the JTAG Boundary Scan logic controller. This allows for communication between the internal running design and the dedicated JTAG pins of the FPGA.

Each instance of this design element will handle one JTAG USER instruction (USER1 through USER4) as set with the JTAG_CHAIN attribute. To handle all four USER instructions, instantiate four of these elements and set the JTAG_CHAIN attribute appropriately.

Note For specific information on boundary scan for an architecture, see the Spartan-6 Configuration User Guide for this element.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTURE</td>
<td>Output</td>
<td>1</td>
<td>CAPTURE output from TAP controller.</td>
</tr>
<tr>
<td>DRCK</td>
<td>Output</td>
<td>1</td>
<td>Data register output for USER functions.</td>
</tr>
<tr>
<td>RESET</td>
<td>Output</td>
<td>1</td>
<td>Reset output for TAP controller.</td>
</tr>
<tr>
<td>RUNTEST</td>
<td>Output</td>
<td>1</td>
<td>Output signal that gets asserted when TAP controller is in Run Test Idle state.</td>
</tr>
<tr>
<td>SEL</td>
<td>Output</td>
<td>1</td>
<td>USER active output.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>Output</td>
<td>1</td>
<td>SHIFT output from TAP controller.</td>
</tr>
<tr>
<td>TCK</td>
<td>Output</td>
<td>1</td>
<td>Scan Clock output. Fabric connection to TAP Clock pin.</td>
</tr>
<tr>
<td>TDI</td>
<td>Output</td>
<td>1</td>
<td>TDI output from TAP controller.</td>
</tr>
<tr>
<td>TDO</td>
<td>Input</td>
<td>1</td>
<td>Data input for USER function.</td>
</tr>
<tr>
<td>TMS</td>
<td>Output</td>
<td>1</td>
<td>Test Mode Select output. Fabric connection to TAP.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Output</td>
<td>1</td>
<td>UPDATE output from TAP controller.</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| JTAG_CHAIN  | Integer  | 1, 2, 3, 4     | 1       | Sets the JTAG USER instruction number that this instance of the element will handle.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- BSCAN_SPARTAN6: JTAG Boundary Scan Logic Control Circuit
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

BSCAN_SPARTAN6_inst : BSCAN_SPARTAN6
generic map (    JTAG_CHAIN => 1 -- Value for USER command. Possible values: (1,2,3 or 4).
)    port map (    CAPTURE => CAPTURE, -- 1-bit output: CAPTURE output from TAP controller.
    DRCK => DRCK, -- 1-bit output: Data register output for USER functions.
    RESET => RESET, -- 1-bit output: Reset output for TAP controller.
    RUNTEST => RUNTEST, -- 1-bit output: Output signal that gets asserted when TAP controller is in Run Test -- Idle state.
    SEL => SEL, -- 1-bit output: USER active output.
    SHIFT => SHIFT, -- 1-bit output: SHIFT output from TAP controller.
    TCK => TCK, -- 1-bit output: Scan Clock output. Fabric connection to TAP Clock pin.
    TDI => TDI, -- 1-bit output: TDI output from TAP controller.
    TMS => TMS, -- 1-bit output: Test Mode Select output. Fabric connection to TAP.
    UPDATE => UPDATE, -- 1-bit output: UPDATE output from TAP controller
    TDO => TDO -- 1-bit input: Data input for USER function.
    );
```

Verilog Instantiation Template

```verilog
// BSCAN_SPARTAN6: JTAG Boundary Scan Logic Control Circuit
// Spartan-6

BSCAN_SPARTAN6 #(    .JTAG_CHAIN(1) // Value for USER command. Possible values: (1,2,3 or 4).
)
BSCAN_SPARTAN6_inst (    .CAPTURE(CAPTURE), // 1-bit output: CAPTURE output from TAP controller.
    .DRCK(DRCK), // 1-bit output: Data register output for USER functions.
    .RESET(RESET), // 1-bit output: Reset output for TAP controller.
    .RUNTEST(RUNTEST), // 1-bit output: Output signal that gets asserted when TAP controller is in Run Test // Idle state.
);```

Spartan-6 Libraries Guide for HDL Designs
UG615 (v 13.3) October 26, 2011

www.xilinx.com
Chapter 4: About Design Elements

```
.SEL(SEL),  // 1-bit output: USER active output.
.SHIFT(SHIFT),  // 1-bit output: SHIFT output from TAP controller.
.TCK(TCK),  // 1-bit output: Scan Clock output. Fabric connection to TAP Clock pin.
.TDI(TDI),  // 1-bit output: TDI output from TAP controller.
.TMS(TMS),  // 1-bit output: Test Mode Select output. Fabric connection to TAP.
.UPDATE(UPDATE),  // 1-bit output: UPDATE output from TAP controller
.TDO(TDO)  // 1-bit input: Data input for USER function.
);

// End of BSCAN_SPARTAN6_inst instantiation

For More Information

- See the Spartan-6 FPGA Configuration User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
```
**BUFG**

**Convenience Primitive: Global Clock Buffer**

This design element is a high-fanout buffer that connects signals to the global routing resources for low skew distribution of the signal. BUFGs are typically used on clock nets as well other high fanout nets like sets/resets and clock enables.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Clock buffer input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock buffer output</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFG: Global Clock Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

BUFG_inst : BUFG
port map (  
  O => O, -- 1-bit output: Clock buffer output  
  I => I -- 1-bit input: Clock buffer input  
);

-- End of BUFG_inst instantiation
```

---

Chapter 4: About Design Elements
Verilog Instantiation Template

// BUFG: Global Clock Buffer
// Spartan-6

BUFG BUFG_inst {
    .O(O), // 1-bit output: Clock buffer output
    .I(I)  // 1-bit input: Clock buffer input
};

// End of BUFG_inst instantiation

For More Information

- See the Spartan-6 FPGA Clocking Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

BUFGCE

Convenience Primitive: Global Clock Buffer with Clock Enable

Introduction

This design element is a global clock buffer with a single gated input. Its O output is "0" when clock enable (CE) is Low (inactive). When clock enable (CE) is High, the I input is transferred to the O output.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>CE</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Clock buffer input</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Clock enable input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock buffer output</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFGCE: Global Clock Buffer with Clock Enable
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

BUFGCE_inst : BUFGCE
port map (  
  O => O,   -- 1-bit output: Clock buffer output
  CE => CE, -- 1-bit input: Clock buffer select
  I => I    -- 1-bit input: Clock buffer input (S=0)
);

-- End of BUFGCE_inst instantiation
```
Verilog Instantiation Template

// BUFGCE: Global Clock Buffer with Clock Enable
// Spartan-6
BUFGCE BUFGCE_inst (
    .O(O),  // 1-bit output: Clock buffer output
    .CE(CE),  // 1-bit input: Clock buffer select
    .I(I)  // 1-bit input: Clock buffer input (S=0)
);

// End of BUFGCE_inst instantiation

For More Information

- See the Spartan-6 FPGA Clocking Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
**BUFGCE_1**

**Convenience Primitive: Global Clock Buffer with Clock Enable and Output State 1**

![BUFGCE_1](image)

**Introduction**

This design element is a multiplexed global clock buffer with a single gated input. Its O output is High (1) when clock enable (CE) is Low (inactive). When clock enable (CE) is High, the I input is transferred to the O output.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>CE</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
</tbody>
</table>

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Clock buffer input</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Clock enable input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock buffer output</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFGCE_1: Global Clock Buffer with Clock Enable and Output State 1
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
BUFGCE_1_inst : BUFGCE_1
port map (  
  O => O, -- 1-bit output: Clock buffer output
  CE => CE, -- 1-bit input: Clock buffer select
  I => I    -- 1-bit input: Clock buffer input (S=0)
);
```

Spartan-6 Libraries Guide for HDL Designs
UG615 (v 13.3) October 26, 2011  www.xilinx.com  57
-- End of BUFGCE_1_inst instantiation

Verilog Instantiation Template

// BUFGCE_1: Global Clock Buffer with Clock Enable and Output State 1
// Spartan-6
BUFGCE_1 BUFGCE_1_inst (  
.O(O), // 1-bit output: Clock buffer output  
.CE(CE), // 1-bit input: Clock buffer select  
.I(I)   // 1-bit input: Clock buffer input (S=0)  
);  
// End of BUFGCE_1_inst instantiation

For More Information

- See the Spartan-6 FPGA Clocking Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
BURUGMUX

Primitive: Global Clock MUX Buffer

Introduction

BUFGMUX is a multiplexed global clock buffer that can select between two input clocks: I0 and I1. When the select input (S) is Low, the signal on I0 is selected for output (O). When the select input (S) is High, the signal on I1 is selected for output.

BUFGMUX and BUFGMUX_1 are distinguished by the state the output assumes when that output switches between clocks in response to a change in its select input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Note  BUFGMUX guarantees that when S is toggled, the state of the output remains in the inactive state until the next active clock edge (either I0 or I1) occurs.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>I1</td>
</tr>
<tr>
<td>I0</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>I1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Clock0 input</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Clock1 input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock MUX output</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Clock select input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th></th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td></td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK_SEL_TYPE</td>
<td>String</td>
<td>“SYNC”, “ASYNC”</td>
<td>“SYNC”</td>
<td>Specifies synchronous or asynchronous clock.</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFGMUX: Global Clock Mux Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

BUFGMUX_inst : BUFGMUX
generic map
    CLK_SEL_TYPE => "SYNC" -- Glitches ("SYNC") or fast ("ASYNC") clock switch-over
) port map
    O => O, -- 1-bit output: Clock buffer output
    I0 => I0, -- 1-bit input: Clock buffer input (S=0)
    I1 => I1, -- 1-bit input: Clock buffer input (S=1)
    S => S -- 1-bit input: Clock buffer select
);  
-- End of BUFGMUX_inst instantiation
```

**Verilog Instantiation Template**

```verilog
// BUFGMUX: Global Clock Mux Buffer
// Spartan-6

BUFGMUX #(  
    .CLK_SEL_TYPE("SYNC") // Glitches ("SYNC") or fast ("ASYNC") clock switch-over
) BUFGMUX_inst (  
    .O(O), // 1-bit output: Clock buffer output
    .I0(I0), // 1-bit input: Clock buffer input (S=0)
    .I1(I1), // 1-bit input: Clock buffer input (S=1)
    .S(S) // 1-bit input: Clock buffer select
);  
// End of BUFGMUX_inst instantiation
```

**For More Information**

- See the [Spartan-6 FPGA Clocking Resources User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
Chapter 4: About Design Elements

BUFDMUX_1

Primitive: Global Clock MUX Buffer with Output State 1

Introduction

This design element is a multiplexed global clock buffer that can select between two input clocks: I0 and I1. When the select input (S) is Low, the signal on I0 is selected for output (O). When the select input (S) is High, the signal on I1 is selected for output.

This design element is distinguished from BUFGMUX by the state the output assumes when that output switches between clocks in response to a change in its select input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>I0</td>
<td>I1</td>
</tr>
<tr>
<td>I0</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>I1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Clock0 input</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Clock1 input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock MUX output</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Clock select input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFGMUX_1: Global Clock Mux Buffer with Output State 1
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
BUFGMUX_1_inst : BUFGMUX_1
generic map (
  CLK_SEL_TYPE => "SYNC" -- Glitchles ("SYNC") or fast ("ASYNC") clock switch-over
)
port map (  
  O => O, -- 1-bit output: Clock buffer output
  I0 => I0, -- 1-bit input: Clock buffer input
  I1 => I1, -- 1-bit input: Clock buffer input
  S => S -- 1-bit input: Clock buffer select
);
-- End of BUFGMUX_1_inst instantiation

Verilog Instantiation Template

// BUFGMUX_1: Global Clock Mux Buffer with Output State 1
// Spartan-6
BUFGMUX_1 #(
  .CLK_SEL_TYPE("SYNC") // Glitchles ("SYNC") or fast ("ASYNC") clock switch-over
) BUFGMUX_1_inst {
  .O(O), // 1-bit output: Clock buffer output
  .I0(I0), // 1-bit input: Clock buffer input
  .I1(I1), // 1-bit input: Clock buffer input
  .S(S) // 1-bit input: Clock buffer select
};
// End of BUFGMUX_1_inst instantiation

For More Information

• See the Spartan-6 FPGA Clocking Resources User Guide.
• See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.


**Bufh**

**Primitive:** Clock buffer for a single clocking region

![Bufh](image)

**Introduction**

The BUFH primitive is provided to allow instantiation capability to access the HCLK clock buffer resources. The use of this component requires manual placement and special consideration and thus is recommended for more advanced users. Please refer to the *Spartan-6 FPGA Clocking Resources User Guide* (UG382) for details in using this component.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Clock Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock Output</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<table>
<thead>
<tr>
<th>Method</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- Bufh: HROW Clock Buffer for a Single Clocking Region
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

BUFH_inst : BUFH
port map (  
  O => O,  -- 1-bit output: Clock output  
  I => I  -- 1-bit input: Clock input
);

-- End of BUFH_inst instantiation
```

---

**Spartan-6 Libraries Guide for HDL Designs**

UG615 (v 13.3) October 26, 2011  
www.xilinx.com  
63
Verilog Instantiation Template

// BUFH: HROW Clock Buffer for a Single Clocking Region
// Spartan-6

BUFH BUFH_inst {
  .O(O), // 1-bit output: Clock output
  .I(I) // 1-bit input: Clock input
};

// End of BUFH_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
Chapter 4: About Design Elements

BUFIO2

Primitive: Dual Clock Buffer and Strobe Pulse

**Introduction**

This primitive provides high-speed I/O clocking resources from an off-chip source intended to drive the synchronous I/O resources (ISERDES2, OSERDES2) and associated fabric resources via a BUFG with low skew. Please refer to the Spartan-6 FPGA Clocking Resources User Guide (UG382) for details in using this component.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVCLK</td>
<td>Output</td>
<td>1</td>
<td>Divided clock output</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Clock input</td>
</tr>
<tr>
<td>IOCLK</td>
<td>Output</td>
<td>1</td>
<td>Clock output</td>
</tr>
<tr>
<td>SERDESSTROBE</td>
<td>Output</td>
<td>1</td>
<td>Output SERDES Strobe (connect to ISERDES/OSERDES)</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<table>
<thead>
<tr>
<th>Method</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVIDE</td>
<td>Decimal</td>
<td>1, 2, 3, 4, 5, 6, 7, 8</td>
<td>1</td>
<td>DIVCLK divider</td>
</tr>
<tr>
<td>DIVIDE_BYPASS</td>
<td>Boolean</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Bypass the divider circuitry</td>
</tr>
<tr>
<td>L_INVERT</td>
<td>Boolean</td>
<td>FALSE, TRUE</td>
<td>FALSE</td>
<td>Invert clock</td>
</tr>
<tr>
<td>USE_DOUBLER</td>
<td>Boolean</td>
<td>FALSE, TRUE</td>
<td>FALSE</td>
<td>Use doubler circuitry</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFIO2: I/O Clock Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

BUFIO2_inst : BUFIO2
generic map (  
  DIVIDE => 1, -- DIVCLK divider (1-8)  
  DIVIDE_BYPASS => TRUE, -- Bypass the divider circuitry (TRUE/FALSE)  
  I_INVERT => FALSE, -- Invert clock (TRUE/FALSE)  
  USE_DOUBLER => FALSE -- Use doubler circuitry (TRUE/FALSE)  
)  
port map (  
  DIVCLK => DIVCLK, -- 1-bit output: Divided clock output  
  IOCLK => IOCLK, -- 1-bit output: I/O output clock  
  SERDESSTROBE => SERDESSTROBE, -- 1-bit output: Output SERDES strobe (connect to ISERDES2/OSERDES2)  
  I => I -- 1-bit input: Clock input (connect to IBUFG)  
);  
-- End of BUFIO2_inst instantiation

Verilog Instantiation Template

// BUFIO2: I/O Clock Buffer  
// Spartan-6  

BUFIO2 #(  
  .DIVIDE(1), // DIVCLK divider (1-8)  
  .DIVIDE_BYPASS("TRUE"), // Bypass the divider circuitry (TRUE/FALSE)  
  .I_INVERT("FALSE"), // Invert clock (TRUE/FALSE)  
  .USE_DOUBLER("FALSE") // Use doubler circuitry (TRUE/FALSE)  
)  
BUFIO2_inst (  
  .DIVCLK(DIVCLK), // 1-bit output: Divided clock output  
  .IOCLK(IOCLK), // 1-bit output: I/O output clock  
  .SERDESSTROBE(SERDESSTROBE), // 1-bit output: Output SERDES strobe (connect to ISERDES2/OSERDES2)  
  .I(I) // 1-bit input: Clock input (connect to IBUFG)  
);  
// End of BUFIO2_inst instantiation

For More Information

- See the Spartan-6 FPGA Clocking Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
**BUFIO2_2CLK**

**Primitive: Dual Clock Buffer and Strobe Pulse with Differential Input**

![BUFIO2_2CLK Diagram]

**Introduction**

The BUFIO2_2CLK resource provides high-speed I/O clocking resources from an off-chip source intended to drive the synchronous I/O resources (ISERDES2, OSERDES2) and associated fabric resources via a BUFG with low skew. Please refer to the _Spartan-6 FPGA Clocking Resources User Guide_ (UG382) for details in using this component.

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>Yes</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFIO2_2CLK: Dual Input Differential Clock Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
BUFIO2_2CLK_inst : BUFIO2_2CLK
generic map (  
  DIVIDE => 2 -- DIVCLK divider (1-8)
)
port map (  
  DIVCLK => DIVCLK,  -- 1-bit output: Divided clock output
  IOCLK => IOCLK,  -- 1-bit output: I/O output clock
  SERDESSTROBE => SERDESSTROBE,  -- 1-bit output: Output SERDES strobe (connect to ISERDES2/OSERDES2)
  I => I,  -- 1-bit input: Clock input (connect to IBUFG)
  IB => IB  -- 1-bit input: Secondary clock input
);
-- End of BUFIO2_2CLK_inst instantiation
```

Verilog Instantiation Template

// BUFIO2_2CLK: Dual Input Differential Clock Buffer
// Spartan-6

BUFI02_2CLK #(  
  .DIVIDE(2),     // DIVCLK divider (1-8)
)  
BUFI02_2CLK_inst (  
  .DIVCLK(DIVCLK),    // 1-bit output: Divided clock output
  .IOCLK(IOCLK),      // 1-bit output: I/O output clock
  .SERDESSTROBE(SERDESSTROBE), // 1-bit output: Output SERDES strobe (connect to ISERDES2/OSERDES2)
  .I(I),               // 1-bit input: Clock input (connect to IBUFG)
  .IB(IB)              // 1-bit input: Secondary clock input
);

// End of BUFI02_2CLK_inst instantiation

For More Information

- See the [Spartan-6 FPGA Clocking Resources User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
**Introduction**

This element is a simple buffer that is delay matched to an associated BUFIO2 which is used for the feedback path for proper phase compensation of the feedback when using a DLL or PLL.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Input feedback clock.</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Output feedback clock (Connect to feedback input of DCM/PLL).</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVIDE_BYPASS</td>
<td>Boolean</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Bypass Divider (TRUE/FALSE) Set the same as associated BUFIO2.</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFIO2FB: DCM/PLL Feedback Clock Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

BUFIO2FB_inst : BUFIO2FB
generic map (    
  DIVIDE_BYPASS => TRUE   -- Bypass divider (TRUE/FALSE)
)
port map (          
  O => O, -- 1-bit output: Output feedback clock (connect to feedback input of DCM/PLL)
  I => I -- 1-bit input: Feedback clock input (connect to input port)
);
-- End of BUFIO2FB_inst instantiation
```
Verilog Instantiation Template

```verilog
// BUFIO2FB: DCM/PLL Feedback Clock Buffer
// Spartan-6
BUFIO2FB #(.
  .DIVIDE_BYPASS("TRUE") // Bypass divider (TRUE/FALSE)
) BUFI02FB_inst (.O(O), // 1-bit output: Output feedback clock (connect to feedback input of DCM/PLL)
  .I(I) // 1-bit input: Feedback clock input (connect to input port)
);
// End of BUFI02FB_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Clocking Resources User Guide](http://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](http://www.xilinx.com).
**BUFPLL**

**Primitive: PLL Buffer**

![Diagram of BUFPLL](image)

**Introduction**

High-speed I/O clock buffer sourced from the PLL component.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCLK</td>
<td>Input</td>
<td>1</td>
<td>BUFG Clock Input.</td>
</tr>
<tr>
<td>IOCLK</td>
<td>Output</td>
<td>1</td>
<td>Output I/O Clock.</td>
</tr>
<tr>
<td>LOCK</td>
<td>Output</td>
<td>1</td>
<td>Synchronized LOCK output.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>Input</td>
<td>1</td>
<td>LOCKED Input from PLL.</td>
</tr>
<tr>
<td>PLLIN</td>
<td>Input</td>
<td>1</td>
<td>Clock Input from PLL.</td>
</tr>
<tr>
<td>SERDESSTROBE</td>
<td>Output</td>
<td>1</td>
<td>SERDES strobe (connect to ISERDES/OSERDES).</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVIDE</td>
<td>Integer</td>
<td>1, 2, 3, 4, 5, 6, 7, 8</td>
<td>1</td>
<td>DIVCLK Divider (1-8)</td>
</tr>
<tr>
<td>ENABLE_SYNC</td>
<td>Boolean</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Enable synchronization between PLL and GCLK (TRUE/FALSE).</td>
</tr>
</tbody>
</table>
Chapter 4: About Design Elements

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- BUFPLL: High-speed I/O PLL clock buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
BUFPLL_inst : BUFPLL
generic map (   
  DIVIDE => 1,   -- DIVCLK divider (1-8)  
  ENABLE_SYNC => TRUE   -- Enable synchronization between PLL and GCLK (TRUE/FALSE)  
)
port map (   
  IOCLK => IOCLK,   -- 1-bit output: Output I/O clock  
  LOCK => LOCK,   -- 1-bit output: Synchronized LOCK output  
  SERDESSTROBE => SERDESSTROBE,   -- 1-bit output: Output SERDES strobe (connect to ISERDES2/OSERDES2)  
  GCLK => GCLK,   -- 1-bit input: BUFG clock input  
  LOCKED => LOCKED,   -- 1-bit input: LOCKED input from PLL  
  PLLIN => PLLIN   -- 1-bit input: Clock input from PLL  
);
-- End of BUFPLL_inst instantiation

Verilog Instantiation Template

// BUFPLL: High-speed I/O PLL clock buffer
// Spartan-6
BUFPLL #(   
  .DIVIDE(1),   // DIVCLK divider (1-8)  
  .ENABLE_SYNC("TRUE")   // Enable synchronization between PLL and GCLK (TRUE/FALSE)  
)  
BUFPLL_inst (   
  .IOCLK(IOCLK),   // 1-bit output: Output I/O clock  
  .LOCK(LOCK),   // 1-bit output: Synchronized LOCK output  
  .SERDESSTROBE(SERDESSTROBE),   // 1-bit output: Output SERDES strobe (connect to ISERDES2/OSERDES2)  
  .GCLK(GCLK),   // 1-bit input: BUFG clock input  
  .LOCKED(LOCKED),   // 1-bit input: LOCKED input from PLL  
  .PLLIN(PLLIN)   // 1-bit input: Clock input from PLL  
);
-- End of BUFPLL_inst instantiation

For More Information

- See the Spartan-6 FPGA Clocking Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
BUFPLL_MCB

Primitive: PLL Buffer for the Memory Controller Block

Introduction

The BUFPLL_MCB is a component used by the Memory Interface Generator (MIG) core in conjunction with the MCB block to implement external memory interfaces. The use of this block outside of MIG is not supported.

For More Information

- See the Xilinx Memory Interface Solutions User Guide.
- See the Spartan-6 FPGA Clocking Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
CARRY4

Primitive: Fast Carry Logic with Look Ahead

Introduction

This circuit design represents the fast carry logic for a slice. The carry chain consists of a series of four MUXes and four XORs that connect to the other logic (LUTs) in the slice via dedicated routes to form more complex functions. The fast carry logic is useful for building arithmetic functions like adders, counters, subtractors and add/subs, as well as such other logic functions as wide comparators, address decoders, and some logic gates (specifically, AND and OR).

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>4</td>
<td>Carry chain XOR general data out</td>
</tr>
<tr>
<td>CO</td>
<td>Output</td>
<td>4</td>
<td>Carry-out of each stage of the carry chain</td>
</tr>
<tr>
<td>DI</td>
<td>Input</td>
<td>4</td>
<td>Carry-MUX data input</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>4</td>
<td>Carry-MUX select line</td>
</tr>
<tr>
<td>CYINIT</td>
<td>Input</td>
<td>1</td>
<td>Carry-in initialization input</td>
</tr>
<tr>
<td>CI</td>
<td>Input</td>
<td>1</td>
<td>Carry cascade input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- CARRY4: Fast Carry Logic Component
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

CARRY4_inst : CARRY4
port map (  
  CO => CO, -- 4-bit carry out
  O => O, -- 4-bit carry chain XOR data out
  CI => CI, -- 1-bit carry cascade input
  CYINIT => CYINIT, -- 1-bit carry initialization
  DI => DI, -- 4-bit carry-MUX data in
  S => S -- 4-bit carry-MUX select input
);

-- End of CARRY4_inst instantiation

Verilog Instantiation Template

// CARRY4: Fast Carry Logic Component
// Spartan-6

CARRY4 CARRY4_inst (  
  .CO(CO), // 4-bit carry out
  .O(O), // 4-bit carry chain XOR data out
  .CI(CI), // 1-bit carry cascade input
  .CYINIT(CYINIT), // 1-bit carry initialization
  .DI(DI), // 4-bit carry-MUX data in
  .S(S) // 4-bit carry-MUX select input
);

// End of CARRY4_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

CFGLUT5

Primitive: 5-input Dynamically Reconfigurable Look-Up Table (LUT)

Introduction

This element is a runtime, dynamically reconfigurable, 5-input look-up table (LUT) that enables the changing of the logical function of the LUT during circuit operation. Using the CDI pin, a new INIT value can be synchronously shifted in serially to change the logical function. The O6 output pin produces the logical output function, based on the current INIT value loaded into the LUT and the currently selected I0-I4 input pins. Optionally, you can use the O5 output in combination with the O6 output to create two individual 4-input functions sharing the same inputs or a 5-input function and a 4-input function that uses a subset of the 5-input logic (see tables below). This component occupies one of the four 6-LUT components within a slice.

To cascade this element, connect the CDO pin from each element to the CDI input of the next element. This will allow a single serial chain of data (32-bits per LUT) to reconfigure multiple LUTs.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O6</td>
<td>Output</td>
<td>1</td>
<td>5-LUT output</td>
</tr>
<tr>
<td>O5</td>
<td>Output</td>
<td>1</td>
<td>4-LUT output</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
<tr>
<td>CDO</td>
<td>Output</td>
<td>1</td>
<td>Reconfiguration data cascaded output (optionally connect to the CDI input of a subsequent LUT)</td>
</tr>
<tr>
<td>CDI</td>
<td>Input</td>
<td>1</td>
<td>Reconfiguration data serial input</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Reconfiguration clock</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Active high reconfiguration clock enable</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Yes</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

- Connect the CLK input to the clock source used to supply the reconfiguration data.
- Connect the CDI input to the source of the reconfiguration data.
- Connect the CE pin to the active high logic if you need to enable/disable LUT reconfiguration.
- Connect the I4-I0 pins to the source inputs to the logic equation. The logic function is output on O6 and O5.
- To cascade this element, connect the CDO pin from each element to the CDI input of the next element to allow a single serial chain of data to reconfigure multiple LUTs.

The INIT attribute should be placed on this design element to specify the initial logical function of the LUT. A new INIT can be loaded into the LUT any time during circuit operation by shifting in 32-bits per LUT in the chain, representing the new INIT value. Disregard the O6 and O5 output data until all 32-bits of new INIT data has been clocked into the LUT. The logical function of the LUT changes as new INIT data is shifted into it. Data should be shifted in MSB (INIT[31]) first and LSB (INIT[0]) last.

In order to understand the O6 and O5 logical value based on the current INIT, see the table below:

<table>
<thead>
<tr>
<th>I4 I3 I2 I1 I0</th>
<th>O6 Value</th>
<th>O5 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1</td>
<td>INIT[31]</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>1 1 1 1 0</td>
<td>INIT[30]</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>1 0 0 0 1</td>
<td>INIT[17]</td>
<td>INIT[1]</td>
</tr>
<tr>
<td>1 0 0 0 0</td>
<td>INIT[16]</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>0 1 1 1 1</td>
<td>INIT[15]</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>0 1 1 1 0</td>
<td>INIT[14]</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>0 0 0 0 1</td>
<td>INIT[1]</td>
<td>INIT[1]</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
<td>INIT[0]</td>
<td>INIT[0]</td>
</tr>
</tbody>
</table>

For instance, the INIT value of FFFFFFF000 would represent the following logical equations:
- O6 = I4 or (I3 and I2 and I1 and I0)
- O5 = I3 and I2 and I1 and I0

To use these elements as two, 4-input LUTs with the same inputs but different functions, tie the I4 signal to a logical one. The INIT[31:16] values apply to the logical values of the O6 output and INIT [15:0] apply to the logical values of the O5 output.

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-bit Value</td>
<td>All zeros</td>
<td>Specifies the initial logical expression of this element.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- CFGLUT5: Reconfigurable 5-input LUT
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

CFGLUT5_inst : CFGLUT5
generic map (
    INT => X"00000000"
) port map (  
    CDO => CDO, -- Reconfiguration cascade output  
    O5 => O5, -- 4-LUT output  
    O6 => O6, -- 5-LUT output  
    CDI => CDI, -- Reconfiguration data input  
    CE => CE, -- Reconfiguration enable input  
    CLK => CLK, -- Clock input  
    I0 => I0, -- Logic data input  
    I1 => I1, -- Logic data input  
    I2 => I2, -- Logic data input  
    I3 => I3, -- Logic data input  
    I4 => I4 -- Logic data input
);

-- End of CFGLUT5_inst instantiation
```

Verilog Instantiation Template

```
// CFGLUT5: Reconfigurable 5-input LUT
// Spartan-6

CFGLUT5 #(  
    .INIT(32'h00000000) // Specify initial LUT contents
) CFGLUT5_inst (  
    .CDO(CDO), // Reconfiguration cascade output  
    .O5(O5), // 4-LUT output  
    .O6(O6), // 5-LUT output  
    .CDI(CDI), // Reconfiguration data input  
    .CE(CE), // Reconfiguration enable input  
    .CLK(CLK), // Clock input  
    .I0(I0), // Logic data input  
    .I1(I1), // Logic data input  
    .I2(I2), // Logic data input  
    .I3(I3), // Logic data input  
    .I4(I4) // Logic data input
);

// End of CFGLUT5_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](#).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](#).
**DCM_CLKGEN**

**Primitive: Digital Clock Manager.**

**Introduction**

Digital Clock Manager (DCM) is set to frequency aligned mode and thus is not phase aligned (no phase relationship) to the input clock. By being in frequency aligned mode, it allows additional capabilities including programmable output clock synthesis, jitter reduction, spread spectrum, and free running oscillator modes. Please refer to the *Spartan-6 FPGA Clocking Resources User Guide* (UG382) for details in using this component.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKFX</td>
<td>Output</td>
<td>1</td>
<td>Synthesized clock output, controlled by the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes. Can be either statically set or dynamically programmed through a dedicated 4-wire SPI port (PROGDATA, PROGCLK, PROGDONE, and PROGEN). Always has a 50% duty cycle.</td>
</tr>
<tr>
<td>CLKFXDV</td>
<td>Output</td>
<td>1</td>
<td>Divided CLKFX output clock. Divide value derived from CLKFXDV_DIVIDE attribute. There is no phase alignment between CLKFX and CLKFXDV.</td>
</tr>
<tr>
<td>CLKFX180</td>
<td>Output</td>
<td>1</td>
<td>Synthesized clock output CLKFX, 180 phase shift (appears to be an inverted version of CLKFX). Always has a 50% duty cycle.</td>
</tr>
<tr>
<td>CLkin</td>
<td>Input</td>
<td>1</td>
<td>Clock input to DCM. Always required. The CLkin frequency and jitter must fall within the limits specified in the data sheet. In the case of free-running oscillator mode, running clock needs to be connected until DCM is locked and DCM is frozen, then clock can be removed. In the other modes, a free running clock needs to be provided and remains.</td>
</tr>
<tr>
<td>FREEZEDCM</td>
<td>Input</td>
<td>1</td>
<td>Prevents tap adjustment drift in the event of a lost CLkin input. The DCM is then configured into a free-run mode.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>Output</td>
<td>1</td>
<td>Synchronous output indicates whether the DCM is ready for operation.</td>
</tr>
<tr>
<td>PROGCLK</td>
<td>Input</td>
<td>1</td>
<td>Clock input for M and/or D reconfiguration.</td>
</tr>
<tr>
<td>PROGDATA</td>
<td>Input</td>
<td>1</td>
<td>Serial data input to supply information for the reprogramming of M and/or D values of the DCM. This input must be applied synchronous to the PROGCLK input.</td>
</tr>
</tbody>
</table>
Chapter 4: About Design Elements

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGDONE</td>
<td>Output</td>
<td>1</td>
<td>Active high output to indicate the successful reprogramming of an M or D value.</td>
</tr>
<tr>
<td>PROGEN</td>
<td>Input</td>
<td>1</td>
<td>Active high enable input for the reprogramming of M/D values. This input must be applied synchronously to the PROGCLK input.</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Resets the DCM circuitry. The RST signal is an active High asynchronous reset. Asserting the RST signal asynchronously forces all DCM outputs Low (the LOCKED signal, all status signals, and all output clocks within four source clock cycles). Because the reset is asynchronous, the last cycle of the clock can exhibit an unintended short pulse, severely distorted duty-cycle, and no longer phase adjust with respect to one another while deasserting. The RST pin must be used when reconfiguring the device or changing the input frequency. Deasserting the RST signal synchronously starts the locking process at the next CLKIN cycle. To ensure a proper DCM reset and locking process, the RST signal must be deasserted after the CLKIN signal has been present and stable for at least three clock cycles. In all designs, the DCM must be held in reset until the clock is stable. During configuration, the DCM is automatically held in reset until GSR is released. If the clock is stable when GSR is released, DCM reset after configuration is not necessary.</td>
</tr>
<tr>
<td>STATUS[2:1]</td>
<td>Output</td>
<td>2</td>
<td>Clock Status lines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STATUS[1] indicates that CLKIN has stopped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STATUS[2] indicates that CLKFX or CLKFX180 has stopped.</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Component</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>Recommended</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKFX_DIVIDE</td>
<td>Integer</td>
<td>1 to 256</td>
<td>1</td>
<td>This value in conjunction with the input frequency and CLKFX_MULTIPLY value, determines the resultant output frequency for the CLKFX and CLKFX180 outputs.</td>
</tr>
<tr>
<td>CLKFXDV_DIVIDE</td>
<td>Integer</td>
<td>2, 4, 8, 16, 32</td>
<td>2</td>
<td>Specifies divide value for CLKFXDV.</td>
</tr>
<tr>
<td>CLKFX_MD_MAX</td>
<td>3 significant digit Float</td>
<td>0.000 to 256.000</td>
<td>0.000</td>
<td>When using the DCM_CLKGEN with variable M and D values, this would specify the maximum ratio of M and D used during static timing analysis.</td>
</tr>
<tr>
<td>CLKFX_MULTIPLY</td>
<td>Integer</td>
<td>2 to 256</td>
<td>4</td>
<td>This value in conjunction with the input frequency and CLKFX_DIVIDE value, determine</td>
</tr>
</tbody>
</table>

Spartan-6 Libraries Guide for HDL Designs
www.xilinx.com
UG615 (v 13.3) October 26, 2011
### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- DCM_CLKGEN: Frequency Aligned Digital Clock Manager
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

DCM_CLKGEN_inst : DCM_CLKGEN
generic map (  
  CLKFXDV_DIVIDE => 2, -- CLKFXDV divide value (2, 4, 8, 16, 32)  
  CLKFX_DIVIDE => 1, -- Divide value - D - (1-256)  
  CLKFX_MULTIPLY => 4, -- Multiply value - M - (2-256)  
  CLKIN_PERIOD => 0.0, -- Input clock period specified in nS  
  SPREAD_SPECTRUM => "NONE", -- Spread Spectrum mode "NONE", "CENTER_LOW_SPREAD", "CENTER_HIGH_SPREAD", "VIDEO_LINK_M0", "VIDEO_LINK_M1", or "VIDEO_LINK_M2"  
  STARTUP_WAIT => FALSE -- Delay config DONE until DCM_CLKGEN LOCKED (TRUE/FALSE)  
)
port map (  
  CLKFX => CLKFX, -- 1-bit output: Generated clock output  
  CLKFX180 => CLKFX180, -- 1-bit output: Generated clock output 180 degree out of phase from CLKFX.  
  CLKFXDV => CLKFXDV, -- 1-bit output: Divided clock output
```

### Attribute Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIN_PERIOD</td>
<td>Float</td>
<td>2.000 to 1000.00</td>
<td>None</td>
<td>Specifies the source clock period which is used to help the DCM adjust for the optimum CLKFX/CLKFX180 outputs and also result in faster locking time.</td>
</tr>
<tr>
<td>DFS_BANDWIDTH</td>
<td>String</td>
<td>“OPTIMIZED”, “HIGH”, “LOW”</td>
<td>“OPTIMIZED”</td>
<td>Specifies the frequency adjust bandwidth of the DCM due to process, voltage, and temperature (PVT).</td>
</tr>
<tr>
<td>PROG_MD_BANDWIDTH</td>
<td>String</td>
<td>“OPTIMIZED”, “HIGH”, “LOW”</td>
<td>“OPTIMIZED”</td>
<td>Specifies the frequency adjust bandwidth of the DCM due to change of programming of the M and D values.</td>
</tr>
<tr>
<td>SPREAD_SPECTRUM</td>
<td>String</td>
<td>“NONE”, “CENTER_LOW_SPREAD”, “CENTER_HIGH_SPREAD”, “VIDEO_LINK_M0”, “VIDEO_LINK_M1”, “VIDEO_LINK_M2”</td>
<td>“NONE”</td>
<td>Specifies a supported mode for Spread Spectrum. Must be used in conjunction with the appropriate IP to fully realize the frequency hopping. Used for fixed spread spectrum (CENTER_LOW_SPREAD / CENTER_HIGH_SPREAD) or soft spread spectrum (VIDEO_LINK_M0 / VIDEO_LINK_M1 / VIDEO_LINK_M2). Soft spread spectrum must be used in conjunction with the soft spread spectrum reference design.</td>
</tr>
<tr>
<td>STARTUP_WAIT</td>
<td>Boolean</td>
<td>FALSE, TRUE</td>
<td>FALSE</td>
<td>Delays the configuration DONE signal until DCM LOCKED signal goes high.</td>
</tr>
</tbody>
</table>
LOCKED => LOCKED, -- 1-bit output: Locked output
PROGDONE => PROGDONE, -- 1-bit output: Active high output to indicate the successful re-programming
STATUS => STATUS, -- 2-bit output: DCM_CLKGEN status
CLKIN => CLKIN, -- 1-bit input: Input clock
FREEZEDCM => FREEZEDCM, -- 1-bit input: Prevents frequency adjustments to input clock
PROGCLK => PROGCLK, -- 1-bit input: Clock input for M/D reconfiguration
PROGDATA => PROGDATA, -- 1-bit input: Serial data input for M/D reconfiguration
PROGEN => PROGEN, -- 1-bit input: Active high program enable
RST => RST -- 1-bit input: Reset input pin
);
-- End of DCM_CLKGEN_inst instantiation

Verilog Instantiation Template

// DCM_CLKGEN: Frequency Aligned Digital Clock Manager
// Spartan-6
DCM_CLKGEN (#{
  .CLKFXDV_DIVIDE(2), // CLKFXDV divide value (2, 4, 8, 16, 32)
  .CLKFX_DIVIDE(1), // Divide value - D - (1-256)
  .CLKFX_MDIV_MAX(0.0), // Specify maximum M/D ratio for timing analysis
  .CLKFX_MULTIPLY(4), // Multiply value - M - (2-256)
  .CLKIN_PERIOD(0.0), // Input clock period specified in nS
  .SPREAD_SPECTRUM("NONE"), // Spread Spectrum mode "NONE", "CENTER_LOW_SPREAD", "CENTER_HIGH_SPREAD",
  "VIDEO_LINK_M0", "VIDEO_LINK_M1" or "VIDEO_LINK_M2"
  .STARTUP_WAIT("FALSE") // Delay config DONE until DCM_CLKGEN LOCKED (TRUE/FALSE)
})
DCM_CLKGEN_inst ( // 1-bit output: Generated clock output
  .CLKFX(CLKFX),
  .CLKFX180(CLKFX180), // 1-bit output: Generated clock output 180 degree out of phase from CLKFX.
  .CLKFXDV(CLKFXDV), // 1-bit output: Divided clock output
  .LOCKED(LOCKED), // 1-bit output: Locked output
  .PROGDONE(PROGDONE), -- 1-bit output: Active high output to indicate the successful re-programming
  .STATUS(STATUS), // 2-bit output: DCM_CLKGEN status
  .CLKIN(CLKIN), // 1-bit input: Input clock
  .FREEZEDCM(FREEZEDCM), // 1-bit input: Prevents frequency adjustments to input clock
  .PROGCLK(PROGCLK), -- 1-bit input: Clock input for M/D reconfiguration
  .PROGDATA(PROGDATA), // 1-bit input: Serial data input for M/D reconfiguration
  .PROGEN(PROGEN), // 1-bit input: Active high program enable
  .RST(RST) // 1-bit input: Reset input pin
);
-- End of DCM_CLKGEN_inst instantiation

For More Information

- See the Spartan-6 FPGA Clocking Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
**DCM_SP**

**Primitive: Digital Clock Manager**

This design element is a digital clock manager that provides multiple functions. It can implement a clock delay locked loop (DLL), a digital frequency synthesizer (DFS), and a digital phase shifter (DPS). DCM_SPs are useful for eliminating the clock delay coming on and off the chip, shifting the clock phase to improve data capture, deriving different frequency clocks, as well as other useful clocking functions.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKDV</td>
<td>Output</td>
<td>1</td>
<td>Divided clock output, controlled by the CLKDV_DIVIDE attribute. The CLKDV output has a 50% duty cycle unless the CLKDV_DIVIDE attribute is a non-integer value.</td>
</tr>
<tr>
<td>CLKFB</td>
<td>Input</td>
<td>1</td>
<td>Clock feedback input to DCM. The feedback input is required unless the DFS outputs, CLKF or CLKF180, are used standalone. The source of the CLKFB input must be the CLK0 or CLK2X output from the DCM and the CLK_FEEDBACK must be set to 1X or 2X accordingly. When set to NONE, CLKFB is unused and should be tied low. Ideally, the feedback point includes the delay added by the clock distribution network, either internally or externally.</td>
</tr>
<tr>
<td>CLKF</td>
<td>Output</td>
<td>1</td>
<td>Synthesized clock output, controlled by the CLKF_MULTIPLY and CLKF_DIVIDE attributes. Always has a 50% duty cycle. If no phase relationship is necessary, then no clock feedback is required.</td>
</tr>
<tr>
<td>CLKF180</td>
<td>Output</td>
<td>1</td>
<td>Synthesized clock output CLKF, 180 phase shift (appears to be an inverted version of CLKF). Always has a 50% duty cycle. If no phase relationship is necessary, then no feedback loop is required.</td>
</tr>
<tr>
<td>CLKIN</td>
<td>Input</td>
<td>1</td>
<td>Clock input to DCM. Always required. The CLKIN frequency and jitter must fall within the limits specified in the data sheet.</td>
</tr>
<tr>
<td>CLK0</td>
<td>Output</td>
<td>1</td>
<td>Same frequency as CLKIN, 0 phase shift (i.e., not phase shifted). Always conditioned to a 50% duty cycle on Spartan®-6 FPGAs. CLK_FEEDBACK must be set to 1X or 2X to deskew CLK0.</td>
</tr>
<tr>
<td>CLK2X</td>
<td>Output</td>
<td>1</td>
<td>Double-frequency clock output, 0 phase shift. When available, the CLK2X output always has a 50% duty cycle. Either CLK0 or CLK2X is required as a feedback source for DLL functions.</td>
</tr>
</tbody>
</table>
## Chapter 4: About Design Elements

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK2X180</td>
<td>Output</td>
<td>1</td>
<td>Double-frequency clock output, 180 phase shift. When available, the CLK2X180 output always has a 50% duty cycle.</td>
</tr>
<tr>
<td>CLK90</td>
<td>Output</td>
<td>1</td>
<td>Same frequency as CLKin, 90 phase shift (quarter period). Always conditioned to a 50% duty cycle on Spartan®-6 FPGAs.</td>
</tr>
<tr>
<td>CLK180</td>
<td>Output</td>
<td>1</td>
<td>Same frequency as CLKin, 180 phase shift (half period). Always conditioned to a 50% duty cycle on Spartan®-6 FPGAs.</td>
</tr>
<tr>
<td>CLK270</td>
<td>Output</td>
<td>1</td>
<td>Same frequency as CLKin, 270 phase shift (three-quarters period). Always conditioned to a 50% duty cycle on Spartan®-6 FPGAs.</td>
</tr>
</tbody>
</table>
| LOCKED   | Output  | 1     | All DCM features have locked onto the CLKin frequency. Clock outputs are now valid, assuming CLKin is within specified limits.  
• 0 - DCM is attempting to lock onto CLKin frequency. DCM clock outputs are not valid.  
• 1 - DCM is locked onto CLKin frequency. DCM clock outputs are valid.  
• 1-to-0 - DCM lost lock. Reset DCM. |
| PSCLK    | Input   | 1     | Clock input to variable phase shifter, clocked on rising edge. When using a global clock buffer, only the upper eight BUFGMUXs can drive PSCLK: BUFGMUX_X2Y1, BUFGMUX_X2Y2, BUFGMUX_X2Y3, BUFGMUX_X2Y4, BUFGMUX_X3Y5, BUFGMUX_X3Y6, BUFGMUX_X3Y7 and BUFGMUX_X3Y8. |
| PSDONE   | Output  | 1     | Variable phase shift operation complete.  
• 0 - No phase shift operation is active or phase shift operation is in progress.  
• 1 - Requested phase shift operation is complete. Output High for one PSCLK cycle. Next variable phase shift operation can commence. |
| PSEN     | Input   | 1     | Variable phase-shift enable. Can be inverted within a DCM block. Non-inverted behavior shown below.  
• 0 - Disable variable phase shift. Ignore inputs to phase shifter.  
• 1 - Enable variable phase shift operations on next rising PSCLK clock edge.  
**Note** Tie to 0 when not in use. |
| PSINCDEC | Input   | 1     | Increment/decrement variable phase shift. Can be inverted within a DCM block. Non-inverted behavior shown below.  
• 0 - Decrement phase shift value on next enabled, rising PSCLK clock edge.  
• 1 - Increment phase shift value on next enabled, rising PSCLK clock edge. |
**Chapter 4: About Design Elements**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Asynchronous reset input. Resets the DCM logic to its postconfiguration state. Causes DCM to reacquire and relock to the CLkin input. Invertible within DCM block. Non-inverted behavior shown below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 - No effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 - Reset DCM block. Hold RST pulse High for at least three valid CLkin cycles.</td>
</tr>
<tr>
<td>STATUS[7:0]</td>
<td>Output</td>
<td>8</td>
<td>The status output bus provides DCM status.</td>
</tr>
<tr>
<td>STATUS[0]</td>
<td></td>
<td></td>
<td>• STATUS[0] - Variable phase shift overflow. Control output for variable fine phase shifting. The variable phase shifter has reached a minimum or maximum limit value. The limit value is either +/-255 or a lesser value if the phase shift has reached the end of the delay line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 - The phase shift has not yet reached its limit value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 - The phase shift has reached its limited value.</td>
</tr>
<tr>
<td>STATUS[1]</td>
<td></td>
<td></td>
<td>• STATUS[1] - CLKin Input Stopped Indicator. Available only when the CLKFB feedback input is connected. Held in reset until the LOCKED output is asserted. Requires at least one CLKin cycle to become active. Never asserted if CLKin never toggles.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 - CLKin input is toggling.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 - CLKin input is not toggling even though the locked output can still be High.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 - CLKFX and CLKFX180 outputs are toggling.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 - CLKFX and CLKFX180 outputs are not toggling, even though the LOCKED output can still be High.</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>Recommended</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
# Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed_Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK_FEEDBACK</td>
<td>String</td>
<td>“1X”, “2X”, “NONE”</td>
<td>“1X”</td>
<td>Defines the DCM feedback mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1X: CLK0 as feedback.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 2X: CLK2X as feedback.</td>
</tr>
<tr>
<td>CLKDV_DIVIDE</td>
<td>1 significant digit Float</td>
<td>2.0, 1.5, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0</td>
<td>2.0</td>
<td>Specifies the extent to which the CLKDLL, CLKDLLLE, CLKDLLHF, or DCM_SP clock divider (CLKDV output) is to be frequency divided.</td>
</tr>
<tr>
<td>CLKFX_DIVIDE</td>
<td>Integer</td>
<td>1 to 32</td>
<td>1</td>
<td>Specifies the frequency divider value for the CLKFX output.</td>
</tr>
<tr>
<td>CLKFX_MULTIPLY</td>
<td>Integer</td>
<td>2 to 32</td>
<td>4</td>
<td>Specifies the frequency multiplier value for the CLKFX output.</td>
</tr>
<tr>
<td>CLKIN_DIVIDE_BY_2</td>
<td>Boolean</td>
<td>FALSE, TRUE</td>
<td>FALSE</td>
<td>Enables CLKIN divide by two features.</td>
</tr>
<tr>
<td>CLKIN_PERIOD</td>
<td>Float</td>
<td>2.000 to 1000.000</td>
<td>None</td>
<td>Specifies the input period to the DCM_SP CLKIN input in ns.</td>
</tr>
<tr>
<td>CLKOUT_PHASE_SHIFT</td>
<td>String</td>
<td>“NONE”, “FIXED”, “VARIABLE”</td>
<td>“NONE”</td>
<td>This attribute specifies the phase shift mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• NONE: No phase shift capability. Any set value has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• FIXED: DCM outputs are a fixed phase shift from CLKIN. Value is specified by PHASE_SHIFT attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• VARIABLE: Allows the DCM outputs to be shifted in a positive and negative range relative to CLKIN. Starting value is specified by PHASE_SHIFT.</td>
</tr>
<tr>
<td>DESKEW_ADJUST</td>
<td>String</td>
<td>“SYSTEM_SYNCHRONOUS”, “SOURCE_SYNCHRONOUS”</td>
<td>“SYSTEM_SYNCHRONOUS”</td>
<td>Sets configuration bits affecting the clock delay alignment between the DCM_SP output clocks and an FPGA clock input pin.</td>
</tr>
<tr>
<td>DFS_FREQUENCY_MODE</td>
<td>String</td>
<td>“LOW”, “HIGH”</td>
<td>“LOW”</td>
<td>This is a legacy attribute. The DCM is always in the automatic frequency search mode. Setting High or Low makes no effect.</td>
</tr>
<tr>
<td>DLL_FREQUENCY_MODE</td>
<td>String</td>
<td>“LOW”, “HIGH”</td>
<td>“LOW”</td>
<td>This is a legacy attribute. The DCM is always in the automatic frequency search mode. Setting High or Low makes no effect.</td>
</tr>
<tr>
<td>DUTY_CYCLE_CORRECTION</td>
<td>Boolean</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Unsupported</td>
</tr>
<tr>
<td>FACTORY_JF</td>
<td>Hexadecimal</td>
<td>16’h8080 to 16’hfff</td>
<td>16’h080</td>
<td>Unsupported</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Allowed_Values</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>----------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PHASE_SHIFT</td>
<td>Integer</td>
<td>-255 to 255</td>
<td>0</td>
<td>The PHASE_SHIFT attribute is applicable only if the CLKOUT_PHASE_SHIFT attribute is set to FIXED or VARIABLE. Defines the rising-edge skew between CLkin and all the DCM clock outputs at configuration and consequently phase shifts the DCM clock outputs. The skew or phase shift value is specified as an integer that represents a fraction of the clock period as expressed in the equations in Fine Phase Shifting. Actual allowable values depends on input clock frequency. The actual range is less when TCLKin &gt; FINE_SHIFT_RANGE. The FINE_SHIFT_RANGE specification represents the total delay of all taps in the delay line.</td>
</tr>
<tr>
<td>STARTUP_WAIT</td>
<td>Boolean</td>
<td>FALSE, TRUE</td>
<td>FALSE</td>
<td>Controls whether the FPGA configuration signal DONE waits for the DCM to assert its LOCKED signal before going High.  &lt;br&gt;r  • FALSE - Default. DONE is asserted at the end of configuration without waiting for the DCM to assert LOCKED.  &lt;br&gt; • TRUE - The DONE signal does not transition High until the LOCKED signal transitions High on the associated DCM. STARTUP_WAIT does not prevent LOCKED from transitioning High. The FPGA startup sequence must also be modified to insert a LCK (lock) cycle before the postponed cycle. The DONE cycle or the GWE cycle are typical choices. When more than one DCM is configured, the FPGA waits until all DCMs are LOCKED.</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- DCM_SP: Digital Clock Manager  
-- Spartan-6  
-- Xilinx HDL Libraries Guide, version 13.3

DCM_SP_inst : DCM_SP
generic map (  
   CLKDV_DIVIDE => 2.0, -- CLKDV divide value  
   CLKFX_DIVIDE => 1,  -- Divide value on CLKFX outputs - D = (1-32)  
   CLKFX_MULTIPLY => 4, -- Multiply value on CLKFX outputs - M = (2-32)  
   CLKIN_DIVIDE_BY_2 => FALSE, -- CLKIN divide by two (TRUE/FALSE)  
   CLKIN_PERIOD => 10.0, -- Input clock period specified in ns  
   CLKOUT_PHASE_SHIFT => "NONE", -- Output phase shift (NONE, FIXED, VARIABLE)  
   CLK_FEEDBACK => "1X", -- Feedback source (NONE, 1X, 2X)  
   DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS", -- SYSTEM_SYNCHRONOUS or SOURCE_SYNCHRONOUS  
   DFS_FREQUENCY_MODE => "LOW", -- Unsupported - Do not change value
```

---

**XILINX**

**Chapter 4: About Design Elements**

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- DCM_SP: Digital Clock Manager  
-- Spartan-6  
-- Xilinx HDL Libraries Guide, version 13.3

DCM_SP_inst : DCM_SP
generic map (  
   CLKDV_DIVIDE => 2.0, -- CLKDV divide value  
   CLKFX_DIVIDE => 1,  -- Divide value on CLKFX outputs - D = (1-32)  
   CLKFX_MULTIPLY => 4, -- Multiply value on CLKFX outputs - M = (2-32)  
   CLKIN_DIVIDE_BY_2 => FALSE, -- CLKIN divide by two (TRUE/FALSE)  
   CLKIN_PERIOD => 10.0, -- Input clock period specified in ns  
   CLKOUT_PHASE_SHIFT => "NONE", -- Output phase shift (NONE, FIXED, VARIABLE)  
   CLK_FEEDBACK => "1X", -- Feedback source (NONE, 1X, 2X)  
   DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS", -- SYSTEM_SYNCHRONOUS or SOURCE_SYNCHRONOUS  
   DFS_FREQUENCY_MODE => "LOW", -- Unsupported - Do not change value
```

---

Spartan-6 Libraries Guide for HDL Designs

UG615 (v 13.3) October 26, 2011  
www.xilinx.com
Chapter 4: About Design Elements

DLL_FREQUENCY_MODE => "LOW", -- Unsupported - Do not change value
DSS_MODE => "NONE", -- Unsupported - Do not change value
DUTY_CYCLE_CORRECTION => TRUE, -- Unsupported - Do not change value
FACTORY_JF => X"0080", -- Unsupported - Do not change value
PHASE_SHIFT => 0, -- Amount of fixed phase shift (-255 to 255)
STARTUP_WAIT => FALSE -- Delay config DONE until DCM_SP LOCKED (TRUE/FALSE)

port map ( 
  CLK0 => CLK0, -- 1-bit output: 0 degree clock output
  CLK180 => CLK180, -- 1-bit output: 180 degree clock output
  CLK270 => CLK270, -- 1-bit output: 270 degree clock output
  CLK2X => CLK2X, -- 1-bit output: 2X clock frequency clock output
  CLK2X180 => CLK2X180, -- 1-bit output: 2X clock frequency, 180 degree clock output
  CLK90 => CLK90, -- 1-bit output: 90 degree clock output
  CLKOV => CLKOV, -- 1-bit output: Divided clock output
  CLKFX => CLKFX, -- 1-bit output: Digital Frequency Synthesizer output (DFS)
  CLKFX180 => CLKFX180, -- 1-bit output: 180 degree CLKFX output
  LOCKED => LOCKED, -- 1-bit output: DCM_SP Lock Output
  PSDONE => PSDONE, -- 1-bit output: Phase shift done output
  STATUS => STATUS, -- 8-bit output: DCM_SP status output
  CLKFB => CLKFB, -- 1-bit input: Clock feedback input
  CLKIN => CLKIN, -- 1-bit input: Clock input
  DSSEN => DSSEN, -- 1-bit input: Unsupported, specify to GND.
  PSLCLK => PSLCLK, -- 1-bit input: Phase shift clock input
  PSSEN => PSSEN, -- 1-bit input: Phase shift enable
  PSINCDEC => PSINCDEC, -- 1-bit input: Phase shift increment/decrement input
  RST => RST, -- 1-bit input: Active high reset input
);

-- End of DCM_SP_inst instantiation
Verilog Instantiation Template

// DCM_SP: Digital Clock Manager
// Spartan-6

DCM_SP #( // CLKDV divide value
   .CLKDIVIDE(2.0), // Divide value on CLKFX outputs - D - (1-32)
   .CLKFXDIVIDE(1), // Multiply value on CLKFX outputs - M - (2-32)
   .CLKINDIVIDE_BY_2("FALSE"), // CLKIN divide by two (TRUE/FALSE)
   .CLKINPERIOD(10.0), // Input clock period specified in nS
   .CLKOUT_PHASE_SHIFT("NONE"), // Output phase shift (NONE, FIXED, VARIABLE)
   .DESKREW_ADJUST("SYSTEM SYNCHRONOUS"), // SYSTEM_SYNCHRONOUS or SOURCE_SYNCHRONOUS
   .DFS_FREQUENCY_MODE("LOW"), // Unsupported - Do not change value
   .DLL_FREQUENCY_MODE("LOW"), // Unsupported - Do not change value
   .DS_MODE("NONE"), // Unsupported - Do not change value
   .DUTY_CYCLE_CORRECTION("TRUE"), // Unsupported - Do not change value
   .FACTORY_JF(16'hc080), // Do not change value
   .PHASE_SHIFT(0), // Amount of fixed phase shift (-255 to 255)
   .STARTUP_WAIT("FALSE") // Delay config DONE until DCM_SP LOCKED (TRUE/FALSE)
)

DCM_SP_inst ( // 1-bit output: 0 degree clock output
   .CLKO(CLKO),
   .CLK180(CLK180), // 1-bit output: 180 degree clock output
   .CLK270(CLK270), // 1-bit output: 270 degree clock output
   .CLK2X(CLK2X), // 2X clock frequency clock output
   .CLK2X180(CLK2X180), // 2X clock frequency, 180 degree clock output
   .CLK90(CLK90), // 2X clock frequency clock output
   .CLKDIV(CLKDV), // 1-bit output: Divided clock output
   .CLKFX(CLKFX), // 1-bit output: Digital Frequency Synthesizer output (DFS)
   .CLKFX180(CLKFX180), // 1-bit output: 180 degree CLKFX output
   .LOCKED(LOCKED), // 1-bit output: DCM_SP Lock Output
   .PSDONE(PSDONE), // 1-bit output: Phase shift done output
   .STATUS(STATUS), // 8-bit output: DCM_SP status output
   .CLKFB(CLKFB), // 1-bit input: Clock feedback input
   .CLKIN(CLKIN), // 1-bit input: Clock input
   .DSSEN(DSSEN), // 1-bit input: Unsupported, specify to GND.
   .PSCLR(PSCLR), // 1-bit input: Phase shift clock input
   .PSEN(PSEN), // 1-bit input: Phase shift enable
   .PSINCDEC(PSINCDEC), // 1-bit input: Phase shift increment/decrement input
   .RST(RST) // 1-bit input: Active high reset input
);

// End of DCM_SP_inst instantiation

For More Information

- See the **Spartan-6 FPGA Clocking Resources User Guide**.
- See the **Spartan-6 FPGA Data Sheet: DC and Switching Characteristics**.
DNA_PORT

Primitive: Device DNA Data Access Port

Introduction

This element allows access to a dedicated shift register that can be loaded with the Device DNA data bits (unique ID) for a given device. In addition to shifting out the DNA data bits, this component allows for the inclusion of supplemental bits of your data, or allows for the DNA data to rollover (repeat DNA data after initial data has been shifted out). This component is primarily used in conjunction with other circuitry to build added copy protection for the FPGA bitstream from possible theft. Connect all inputs and outputs to the design to ensure proper operation. To access the Device DNA data, you must first load the shift register by setting the active high READ signal for one clock cycle. After the shift register is loaded, the data can be synchronously shifted out by enabling the active high SHIFT input and capturing the data out the DOUT output port. Additional data can be appended to the end of the 57-bit shift register by connecting the appropriate logic to the DIN port. If DNA data rollover is desired, connect the DOUT port directly to the DIN port to allow for the same data to be shifted out after completing the 57-bit shift operation. If no additional data is necessary, the DIN port can be tied to a logic zero. The attribute SIM_DNA_VALUE can be optionally set to allow for simulation of a possible DNA data sequence. By default, the Device DNA data bits are all zeros in the simulation model.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock input.</td>
</tr>
<tr>
<td>DIN</td>
<td>Input</td>
<td>1</td>
<td>User data input pin.</td>
</tr>
<tr>
<td>DOUT</td>
<td>Output</td>
<td>1</td>
<td>DNA output data.</td>
</tr>
<tr>
<td>READ</td>
<td>Input</td>
<td>1</td>
<td>Active high load DNA, active low read input.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>Input</td>
<td>1</td>
<td>Active high shift enable input.</td>
</tr>
</tbody>
</table>

Design Entry Method

| Instantiation | Recommended | Inference | No |
| CORE Generator™ and wizards | No |
| Macro support | No |

Connect all inputs and outputs to the design to ensure proper operation.
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM_DNA_VALUE</td>
<td>Hexadecimal</td>
<td>57'h000000000000000 to 57'h1ffffffffffffff</td>
<td>57'h00000000</td>
<td>Specifies the Pre-programmed factory ID value.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- DNA_PORT: Device DNA Data Access Port
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
DNA_PORT_inst : DNA_PORT
generic map (    SIM_DNA_VALUE => X"000000000000000" -- Specifies the Pre-programmed factory ID value)
port map (    DOUT => DOUT, -- 1-bit output: DNA output data    CLK => CLK, -- 1-bit input: Clock input    DIN => DIN, -- 1-bit input: User data input pin    READ => READ, -- 1-bit input: Active high load DNA, active low read input    SHIFT => SHIFT -- 1-bit input: Active high shift enable input    );
-- End of DNA_PORT_inst instantiation
```

Verilog Instantiation Template

```verilog
// DNA_PORT: Device DNA Data Access Port
// Spartan-6
DNA_PORT (#(    .SIM_DNA_VALUE(57'h000000000000000) // Specifies the Pre-programmed factory ID value    )
DNA_PORT_inst (
    .DOUT(DOUT), // 1-bit output: DNA output data    .CLK(CLK),  // 1-bit input: Clock input    .DIN(DIN),  // 1-bit input: User data input pin    .READ(READ), // 1-bit input: Active high load DNA, active low read input    .SHIFT(SHIFT) // 1-bit input: Active high shift enable input    );
// End of DNA_PORT_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configuration User Guide](http://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](http://www.xilinx.com).
- See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](http://www.xilinx.com).
## DSP48A1

**Primitive: Multi-Functional, Cascadable, 48-bit Output, Arithmetic Block**

---

### Introduction

This element is a versatile, scalable, hard IP block that allows for the creation of compact, high-speed, arithmetic-intensive operations, such as those seen for many DSP algorithms. The block consists of a configurable, 18-bit, pre-add/sub, followed by an 18x18 signed multiplier, followed by a 48-bit post-add/sub/accum. Several configurable pipeline registers exist within the block, allowing for higher clock speeds with the trade-off of added latency. Opmode pins allow the block operation to change from one clock-cycle to the next, thus allowing a single block to serve several arithmetic functions within a design. Furthermore, multiple DSP48A1 blocks can be cascaded to efficiently form larger multiplication and addition functions.
## Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[17:0]</td>
<td>Input</td>
<td>18</td>
<td>18-bit data input to multiplier or post add/sub depending on the value of OPMODE[1:0].</td>
</tr>
<tr>
<td>B[17:0]</td>
<td>Input</td>
<td>18</td>
<td>18-bit data input to multiplier, pre-add/sub and, perhaps, a post-add/sub depending on the value of OPMODE[3:0].</td>
</tr>
<tr>
<td>BCOUT[17:0]</td>
<td>Output</td>
<td>18</td>
<td>Cascade output for Port B. If used, connect to the B port of downstream cascaded DSP48A1. If not used, leave unconnected.</td>
</tr>
<tr>
<td>C[47:0]</td>
<td>Input</td>
<td>48</td>
<td>48-bit data input to post-add/sub.</td>
</tr>
<tr>
<td>CARRYIN</td>
<td>Input</td>
<td>1</td>
<td>External carry input to the post-add/sub. Connect only to the CARRYOUT pin of another DSP48A1 block.</td>
</tr>
<tr>
<td>CARRYOUT</td>
<td>Output</td>
<td>1</td>
<td>Carry out signal for post-add/sub. Connect only to the CARRYIN pin of another DSP48A1.</td>
</tr>
<tr>
<td>CARRYOUTF</td>
<td>Output</td>
<td>1</td>
<td>Carry out signal for post-add/sub that can be routed into the fabric.</td>
</tr>
<tr>
<td>CEA</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable for the A port registers (A0REG=1 or A1REG=1). Tie to logic one if not used and A0REG=1 or A1REG=1. Tie to logic zero if A0REG=0 and A1REG=0.</td>
</tr>
<tr>
<td>CEB</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable for the B port registers (B0REG=1 or B1REG=1). Tie to logic one if not used and B0REG=1 or B1REG=1. Tie to logic zero if B0REG=0 and B1REG=0.</td>
</tr>
<tr>
<td>CEC</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable for the C port registers (CREG=1). Tie to logic one if not used and CREG=1. Tie to logic zero if CREG=0.</td>
</tr>
<tr>
<td>CECARRYIN</td>
<td>Input</td>
<td>1</td>
<td>Active high, clock enable for the carry-in registers (CARRYINREG=1). Tie to logic one if not used and CARRYINREG=1. Tie to a logic zero if CARRYINREG=0.</td>
</tr>
<tr>
<td>CED</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable for the D port registers (DREG=1). Tie to logic one if not used and DREG=1. Tie to logic zero if DREG=0.</td>
</tr>
<tr>
<td>CEM</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable for the multiplier registers (MREG=1). Tie to logic one if not used and MREG=1. Tie to logic zero if MREG=0.</td>
</tr>
<tr>
<td>CEMODE</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable for the OPMODE input registers (OPMODEREG=1). Tie to logic one if not used and OPMODEREG=1. Tie to logic zero if OPMODEREG=0.</td>
</tr>
<tr>
<td>CEP</td>
<td>Input</td>
<td>1</td>
<td>Active high, clock enable for the output port registers (PREG=1). Tie to logic one if not used and PREG=1. Tie to logic zero if PREG=0.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>DSP48A1 clock</td>
</tr>
<tr>
<td>D[17:0]</td>
<td>Input</td>
<td>18</td>
<td>18-bit data input to pre-add/sub.</td>
</tr>
<tr>
<td>M[35:0]</td>
<td>Output</td>
<td>36</td>
<td>Direct multiplier data output to fabric. Do not use if P is used.</td>
</tr>
<tr>
<td>OPMODE</td>
<td>Input</td>
<td>8</td>
<td>Control input to select the arithmetic operations of the DSP48A1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• OPMODE[1:0] - Specifies the source of the X input to the post-add/sub.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 - Specifies to place all zeroes (disable the post-add/sub).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 - Use the POUT output signal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 - Use the concatenated D, B, A input signals.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 - Use the multiplier product.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• OPMODE[3:2] - Specifies the source of the Y input to the post-add/sub.</td>
</tr>
</tbody>
</table>
Chapter 4: About Design Elements

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P[47:0]</td>
<td>Output</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>PCIN[47:0]</td>
<td>Input</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>PCOUT[47:0]</td>
<td>Output</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>RSTA</td>
<td>Input</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RSTB</td>
<td>Input</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RSTC</td>
<td>Input</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RSTCARRYIN</td>
<td>Input</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RSTD</td>
<td>Input</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RSTM</td>
<td>Input</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- 0 - Specifies to place all zeroes (disable the post-add/sub and propagate the multiplier product to POUT)
- 1 - Use the PCIN.
- 2 - Use the PCOUT port (accumulator).
- 3 - Use the C port.
- OPMODE[4] - Specifies the use of the pre-add/sub
  - 0 - Selects to use the pre-adder adding or subtracting the values on the B and D ports prior to the multiplier.
  - 1 - Bypass the pre-adder, supplying the data on Port B directly to the multiplier.
- OPMODE[6] - Specifies whether the pre-add/sub is an adder or subtracter
  - 0 - Specifies pre-add/sub to perform an addition operation.
  - 1 - Specifies pre-add/sub to perform a subtract operation.
- OPMODE[7] - Specifies whether the post-add/sub is an adder or subtracter
  - 0 - Specifies post-add/sub to perform an addition operation.
  - 1 - Specifies post-add/sub to perform a subtract operation.

R[47:0]       | Output    | 48    |          |

- Specifies to place all zeroes (disable the post-add/sub and propagate the multiplier product to POUT)
- Use the PCIN.
- Use the PCOUT port (accumulator).
- Use the C port.
- OPMODE[4] - Specifies the use of the pre-add/sub
  - Selects to use the pre-adder adding or subtracting the values on the B and D ports prior to the multiplier.
  - Bypass the pre-adder, supplying the data on Port B directly to the multiplier.
- OPMODE[6] - Specifies whether the pre-add/sub is an adder or subtracter
  - Specifies pre-add/sub to perform an addition operation.
  - Specifies pre-add/sub to perform a subtract operation.
- OPMODE[7] - Specifies whether the post-add/sub is an adder or subtracter
  - Specifies post-add/sub to perform an addition operation.
  - Specifies post-add/sub to perform a subtract operation.

Primary data output.
Cascade input for Port P. If used, connect to PCOUT of upstream cascaded DSP48A1. If not used, tie port to all zeros.
Cascade output for Port P. If used, connect to PCIN of downstream cascaded DSP48A1. If not used, leave unconnected.
Active high reset for the A port registers (A0REG=1 or A1REG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.
Active high reset for the B port registers (B0REG=1 or B1REG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.
Active high reset for the C port registers (CREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.
Active high reset for the carry-in register (CARRYINREG =1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.
Active high reset for the D port registers (DREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.
Active high reset for the multiplier registers (MREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.
### Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Selects usage of first stage A input pipeline registers. Set to 1 to use the first stage A pipeline registers.</td>
</tr>
<tr>
<td>A1REG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of second stage A input pipeline registers. Set to 1 to use the second stage A pipeline registers.</td>
</tr>
<tr>
<td>B0REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Selects usage of first stage B input pipeline registers. Set to 1 to use the first stage B pipeline registers.</td>
</tr>
<tr>
<td>B1REG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of second stage B input pipeline registers. Set to 1 to use the second stage B pipeline registers. The second stage B pipeline registers are after the pre-adder circuit.</td>
</tr>
<tr>
<td>CARRYINREG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of the CARRYIN input pipeline registers. Set to 1 to use the CARRYIN pipeline registers.</td>
</tr>
<tr>
<td>CARRYINSEL</td>
<td>String</td>
<td>“CARRYIN”, “OPMODE5”</td>
<td>“OPMODE5”</td>
<td>Selects whether the post add/sub carry-in signal should be sourced from the CARRYIN pin (connected to the CARRYOUT of another DSP48A1) or dynamically controlled from the FPGA fabric by the OPMODE[5] input.</td>
</tr>
<tr>
<td>CARRYOUTREG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of CARRYOUT output pipeline registers. Set to 1 to use the CARRYOUT pipeline registers. The registered outputs will include CARRYOUT and CARRYOUTF.</td>
</tr>
<tr>
<td>CREG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of the C input pipeline registers. Set to 1 to use the C pipeline registers.</td>
</tr>
<tr>
<td>DREG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of D pre-adder input pipeline registers. Set to 1 to use the D pipeline registers.</td>
</tr>
<tr>
<td>MREG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of M multiplier output pipeline registers. Set to 1 to use the M pipeline registers.</td>
</tr>
</tbody>
</table>
### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- DSP48A1: 48-bit Multi-Functional Arithmetic Block
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

DSP48A1_inst : DSP48A1
generic map (  
  A0REG => 0,  -- First stage A input pipeline register (0/1)
  A1REG => 1,  -- Second stage A input pipeline register (0/1)
  B0REG => 0,  -- First stage B input pipeline register (0/1)
  B1REG => 1,  -- Second stage B input pipeline register (0/1)
  CARRYINREG => 1,  -- CARRYIN input pipeline register (0/1)
  CARRYINSEL => "OPMODE5",  -- Specify carry-in source, "CARRYIN" or "OPMODE5"
  CARRYOUTREG => 1,  -- CARRYOUT output pipeline register (0/1)
  CREG => 1,  -- C input pipeline register (0/1)
  DREG => 1,  -- D pre-adder input pipeline register (0/1)
  MREG => 1,  -- M pipeline register (0/1)
  OPMODEREG => 1,  -- Enable=1/disable=0 OPMODE input pipeline registers
  PREG => 1,  -- P output pipeline register (0/1)
  RSTTYPE => "SYNC"  -- Specify reset type, "SYNC" or "ASYNC"
)

port map (  
  -- Cascade Ports: 18-bit (each) output: Ports to cascade from one DSP48 to another
  BCOUT => BCOUT,  -- 18-bit output: B port cascade output
  PCOUT => PCOUT,  -- 48-bit output: P cascade output (if used, connect to PCIN of another DSP48A1)
  -- Data Ports: 1-bit (each) output: Data input and output ports
  CARRYOUT => CARRYOUT,  -- 1-bit output: carry output (if used, connect to CARRYIN pin of another
                         -- DSP48A1)
  CARRYOUTF => CARRYOUTF,  -- 1-bit output: fabric carry output
  M => M,  -- 36-bit output: fabric multiplier data output
  P => P,  -- 48-bit output: data output
  -- Cascade Ports: 48-bit (each) input: Ports to cascade from one DSP48 to another
  PCIN => PCIN,  -- 48-bit input: P cascade input (if used, connect to PCOUT of another DSP48A1)
  -- Control Input Ports: 1-bit (each) input: Clocking and operation mode
  CLK => CLK,  -- 1-bit input: clock input
  OPMODE => OPMODE,  -- 8-bit input: operation mode input
  -- Data Ports: 18-bit (each) input: Data input and output ports
  A => A,  -- 18-bit input: A data input
  B => B,  -- 18-bit input: B data input (connected to fabric or BCOUT of adjacent DSP48A1)
  C => C,  -- 48-bit input: C data input
  CARRYIN => CARRYIN,  -- 1-bit input: carry input signal (if used, connect to CARRYOUT pin of another
                           -- DSP48A1)
  D => D,  -- 18-bit input: B pre-adder data input
```

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPMODEREG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of OPMODE input pipeline registers. Set to 1 to use the OPMODE pipeline registers.</td>
</tr>
<tr>
<td>PREG</td>
<td>Integer</td>
<td>1, 0</td>
<td>1</td>
<td>Selects usage of P output pipeline registers. The registered outputs will include P and PCOUT.</td>
</tr>
<tr>
<td>RSTTYPE</td>
<td>String</td>
<td>“SYNC”, “ASYNC”</td>
<td>“SYNC”</td>
<td>Selects whether all resets for the DSP48A1 should have a synchronous or asynchronous reset capability. Due to improved timing and circuit stability, it is recommended to always have this set to ‘SYNC’ unless an asynchronous reset is absolutely necessary.</td>
</tr>
</tbody>
</table>
-- Reset/Clock Enable Input Ports: 1-bit (each) input: Reset and enable input ports
CEA => CEA,  -- 1-bit input: active high clock enable input for A registers
CEB => CEB,  -- 1-bit input: active high clock enable input for B registers
CEC => CEC,  -- 1-bit input: active high clock enable input for C registers
CECARRYIN => CECARRYIN,  -- 1-bit input: active high clock enable input for CARRYIN registers
CED => CED,  -- 1-bit input: active high clock enable input for D registers
CEM => CEM,  -- 1-bit input: active high clock enable input for multiplier registers
CEOPMODE => CEOPMODE,  -- 1-bit input: active high clock enable input for OPMODE registers
CEP => CEP,  -- 1-bit input: active high clock enable input for P registers
RSTA => RSTA,  -- 1-bit input: reset input for A pipeline registers
RSTB => RSTB,  -- 1-bit input: reset input for B pipeline registers
RSTC => RSTC,  -- 1-bit input: reset input for C pipeline registers
RSTCARRYIN => RSTCARRYIN,  -- 1-bit input: reset input for CARRYIN pipeline registers
RSTD => RSTD,  -- 1-bit input: reset input for D pipeline registers
RSTM => RSTM,  -- 1-bit input: reset input for M pipeline registers
RSTOPMODE => RSTOPMODE,  -- 1-bit input: reset input for OPMODE pipeline registers
RSTP => RSTP  -- 1-bit input: reset input for P pipeline registers
);

-- End of DSP48Al_inst instantiation
Verilog Instantiation Template

DSP48A1 #(  
  .A0REG(0),     // First stage A input pipeline register (0/1)  
  .A1REG(1),     // Second stage A input pipeline register (0/1)  
  .B0REG(0),     // First stage B input pipeline register (0/1)  
  .B1REG(1),     // Second stage B input pipeline register (0/1)  
  .CARRYINREG(1), // CARRYIN input pipeline register (0/1)  
  .CARRYINSEL("OPMODE5"), // Specify carry-in source, "CARRYIN" or "OPMODE5"  
  .CARRYOUTREG(1), // CARRYOUT output pipeline register (0/1)  
  .CREG(1),      // C input pipeline register (0/1)  
  .DREG(1),      // D pre-adder input pipeline register (0/1)  
  .MREG(1),      // M pipeline register (0/1)  
  .OPMODEREG(1), // Enable=1/disable=0 OPMODE input pipeline registers  
  .PREG(1),      // P output pipeline register (0/1)  
  .RSTTYPE("SYNC") // Specify reset type, "SYNC" or "ASYNC"  
)
DSP48A1_inst (  
  // Cascade Ports: 18-bit (each) output: Ports to cascade from one DSP48 to another  
  .BCOUT(BCOUT), // 18-bit output: B port cascade output  
  .PCOUT(PCOUT), // 48-bit output: P cascade output (if used, connect to PCIN of another DSP48A1)  
  // Data Ports: 1-bit (each) output: Data input and output ports  
  .CARRYOUT(CARRYOUT), // 1-bit output: carry output (if used, connect to CARRYIN pin of another //DSP48A1)  
  .CARRYOUTF(CARRYOUTF), // 1-bit output: fabric carry output  
  .M(M), // 16-bit output: fabric multiplier data output  
  .P(P), // 48-bit output: data output  
  // Cascade Ports: 48-bit (each) input: Ports to cascade from one DSP48 to another  
  .PCIN(PCIN), // 48-bit input: P cascade input (if used, connect to PCOUT of another DSP48A1)  
  // Control Input Ports: 1-bit (each) input: Clocking and operation mode  
  .CLK(CLK), // 1-bit input: clock input  
  .OPMODE(OPMODE), // 8-bit input: operation mode input  
  // Data Ports: 18-bit (each) input: Data input and output ports  
  .A(A), // 18-bit input: A data input  
  .B(B), // 18-bit input: B data input (connected to fabric or BCOUT of adjacent DSP48A1)  
  .C(C), // 48-bit input: C data input  
  .CARRYIN(CARRYIN), // 1-bit input: carry input signal (if used, connect to CARRYOUT pin of another //DSP48A1)  
  .D(D), // 18-bit input: B pre-adder data input  
  // Reset/Clock Enable Input Ports: 1-bit (each) input: Reset and enable input ports  
  .CEA(CEA), // 1-bit input: active high clock enable input for A registers  
  .CEB(CEB), // 1-bit input: active high clock enable input for B registers  
  .CECC(CECC), // 1-bit input: active high clock enable input for C registers  
  .CECARRYIN(CECARRYIN), // 1-bit input: active high clock enable input for CARRYIN registers  
  .CED(CED), // 1-bit input: active high clock enable input for D registers  
  .CEC(CEC), // 1-bit input: active high clock enable input for multiplier registers  
  .CEOPMODE(CEOPMODE), // 1-bit input: active high clock enable input for OPMODE registers  
  .CEP(CEP), // 1-bit input: active high clock enable input for P registers  
  .RSTA(RSTA), // 1-bit input: reset input for A pipeline registers  
  .RSTB(RSTB), // 1-bit input: reset input for B pipeline registers  
  .RSTC(RSTC), // 1-bit input: reset input for C pipeline registers  
  .RSTCARRYIN(RSTCARRYIN), // 1-bit input: reset input for CARRYIN pipeline registers  
  .RSTD(RSTD), // 1-bit input: reset input for D pipeline registers  
  .RSTM(RSTM), // 1-bit input: reset input for M pipeline registers  
  .RSTOPMODE(RSTOPMODE), // 1-bit input: reset input for OPMODE pipeline registers  
  .RSTP(RSTP) // 1-bit input: reset input for P pipeline registers  
);  

// End of DSP48A1_inst instantiation

For More Information

- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
- See the Spartan-6 FPGA DSP48A1 Slice User Guide.
FDCE

Primitive: D Flip-Flop with Clock Enable and Asynchronous Clear

Introduction

This design element is a single D-type flip-flop with clock enable and asynchronous clear. When clock enable (CE) is High and asynchronous clear (CLR) is Low, the data on the data input (D) of this design element is transferred to the corresponding data output (Q) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data output (Q) Low. When CE is Low, clock transitions are ignored.

This flip-flop is asynchronously cleared, outputs Low, when power is applied. For FPGA devices, power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the appropriate STARTUP_architecture symbol.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Design Entry Method

| Instantiation | Yes |
| Inference | Recommended |
| CORE Generator™ and wizards | No |
| Macro support | No |

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Binary</td>
<td>0, 1</td>
<td>0</td>
<td>Sets the initial value of Q output after configuration. For Spartan®-6, the INIT value should always match the polarity of the set or reset. For this element, the INIT should be 0. If set to 1, an asynchronous circuit must be created to exhibit this behavior, which Xilinx does not recommend.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- FDCE: Single Data Rate D Flip-Flop with Asynchronous Clear and
-- Clock Enable (posedge clk).
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
FDCE_inst : FDCE
generic map (  
INIT => '0') -- Initial value of register ('0' or '1')
port map (  
Q => Q, -- Data output  
C => C, -- Clock input  
CE => CE, -- Clock enable input  
CLR => CLR, -- Asynchronous clear input  
D => D -- Data input  
);
-- End of FDCE_inst instantiation

Verilog Instantiation Template

// FDCE: Single Data Rate D Flip-Flop with Asynchronous Clear and
// Clock Enable (posedge clk).
// Spartan-6
FDCE_FDCE_inst {    
.Q(Q), // 1-bit Data output  
.C(C), // 1-bit Clock input  
.CE(CE), // 1-bit Clock enable input  
.CLR(CLR), // 1-bit Asynchronous clear input  
.D(D) // 1-bit Data input
};
// End of FDCE_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
FDPE

Primitive: D Flip-Flop with Clock Enable and Asynchronous Preset

![FDPE Diagram]

**Introduction**

This design element is a single D-type flip-flop with data (D), clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and sets the (Q) output High. Data on the (D) input is loaded into the flip-flop when PRE is Low and CE is High on the Low-to-High clock (C) transition. When CE is Low, the clock transitions are ignored.

For FPGA devices, this flip-flop is asynchronously preset, output High, when power is applied. Power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the appropriate STARTUP_architecture symbol.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Design Entry Method**

- **Instantiation**: Yes
- **Inference**: Recommended
- **CORE Generator™ and wizards**: No
- **Macro support**: No

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Binary</td>
<td>0, 1</td>
<td>1</td>
<td>Sets the initial value of Q output after configuration. For Spartan®-6, Xilinx recommends that the INIT value always matches the polarity of the set or reset. For this element, the INIT should be 1. If set to 0, additional asynchronous circuitry will be created to correctly model the behavior.</td>
</tr>
</tbody>
</table>
**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- FDPE: Single Data Rate D Flip-Flop with Asynchronous Preset and
--       Clock Enable (posedge clk).
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
FDPE_inst : FDPE
generic map (  
  INIT => '0') -- Initial value of register ('0' or '1')
port map (  
  Q => Q,   -- Data output
  C => C,   -- Clock input
  CE => CE, -- Clock enable input
  PRE => PRE, -- Asynchronous preset input
  D => D    -- Data input
);
-- End of FDPE_inst instantiation
```

**Verilog Instantiation Template**

```verilog
// FDPE: Single Data Rate D Flip-Flop with Asynchronous Preset and
//       Clock Enable (posedge clk).
// Spartan-6
FDPE_FDPE_inst {  
  .Q(Q),   // 1-bit Data output
  .C(C),   // 1-bit Clock input
  .CE(CE), // 1-bit Clock enable input
  .PRE(PRE), // 1-bit Asynchronous preset input
  .D(D)    // 1-bit Data input
};
// End of FDPE_inst instantiation
```

**For More Information**

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
FDRE

Primitive: D Flip-Flop with Clock Enable and Synchronous Reset

Introduction

This design element is a single D-type flip-flop with data (D), clock enable (CE), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the (Q) output Low on the Low-to-High clock (C) transition. The data on the (D) input is loaded into the flip-flop when R is Low and CE is High during the Low-to-High clock transition.

This flip-flop is asynchronously cleared, outputs Low, when power is applied. For FPGA devices, power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the appropriate STARTUP_architecture symbol.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Binary</td>
<td>0, 1</td>
<td>0</td>
<td>Sets the initial value of Q output after configuration For Spartan®, the INIT value should always match the polarity of the set or reset. For this element, the INIT should be 0. If set to 1, an asynchronous circuit must be created to exhibit this behavior, which Xilinx does not recommend.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- FDRE: Single Data Rate D Flip-Flop with Synchronous Reset and
--       Clock Enable (posedge clk).
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

FDRE_inst : FDRE
  generic map (INIT => '0') -- Initial value of register ('0' or '1')
  port map (Q => Q, -- Data output
            C => C, -- Clock input
            CE => CE, -- Clock enable input
            R => R, -- Synchronous reset input
            D => D -- Data input
                );

-- End of FDRE_inst instantiation

Verilog Instantiation Template

// FDRE: Single Data Rate D Flip-Flop with Synchronous Reset and
//       Clock Enable (posedge clk).
// Spartan-6

FDRE undergrad;

FDRE undergrad {
    .Q(Q),  // 1-bit Data output
    .C(C),  // 1-bit Clock input
    .CE(CE), // 1-bit Clock enable input
    .R(R),  // 1-bit Synchronous reset input
    .D(D)   // 1-bit Data input
        };

// End of FDRE_inst instantiation

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
FDSE

Primitive: D Flip-Flop with Clock Enable and Synchronous Set

Introduction

FDSE is a single D-type flip-flop with data (D), clock enable (CE), and synchronous set (S) inputs and data output (Q). The synchronous set (S) input, when High, overrides the clock enable (CE) input and sets the Q output High during the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low and CE is High during the Low-to-High clock (C) transition.

For FPGA devices, this flip-flop is asynchronously preset, output High, when power is applied. Power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the appropriate STARTUP_architecture symbol.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Design Entry Method

| Instantiation | Yes |
| Inference | Recommended |
| CORE Generator™ and wizards | No |
| Macro support | No |

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Binary</td>
<td>0, 1</td>
<td>1</td>
<td>Sets the initial value of Q output after configuration. For Spartan®-6, Xilinx recommends that the INIT value always matches the polarity of the set or reset. For this element, the INIT should be 1. If set to 0, additional asynchronous circuitry will be created to correctly model the behavior.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- FDSE: Single Data Rate D Flip-Flop with Synchronous Set and
--       Clock Enable (posedge clk).
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

FDSE_inst : FDSE
generic map (INIT => '0') -- Initial value of register ('0' or '1')
port map (Q => Q, -- Data output
          C => C, -- Clock input
          CE => CE, -- Clock enable input
          S => S, -- Synchronous Set input
          D => D, -- Data input
);
-- End of FDSE_inst instantiation
```

Verilog Instantiation Template

```verilog
// FDSE: Single Data Rate D Flip-Flop with Synchronous Set and
//       Clock Enable (posedge clk).
// Spartan-6

FDSE_FDSE_inst (Q(0), -- 1-bit Data output
                C(0), -- 1-bit Clock input
                CE(0), -- 1-bit Clock enable input
                S(0), -- 1-bit Synchronous set input
                D(0) -- 1-bit Data input
);
// End of FDSE_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
## GTPA1_DUAL

**Primitive: Dual Gigabit Transceiver**

<table>
<thead>
<tr>
<th>GTPA1_DUAL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Xilinx®

Chapter 4: About Design Elements

UG615 (v 13.3) October 26, 2011

www.xilinx.com

107
Introduction

This design element represents the Spartan®-6 FPGA RocketIO™ GTP transceiver, a power-efficient and highly configurable transceiver. Refer to Spartan-6 FPGA RocketIO GTP Transceiver User Guide for detailed information regarding this component. The Spartan-6 FPGA RocketIO GTX Transceiver Wizard is the preferred tool to generate a wrapper to instantiate a GTPA1_DUAL primitive. The Wizard can be found in the Xilinx® CORE Generator™ tool.

Design Entry Method

To instantiate this component, use the Spartan-6 FPGA RocketIO GTX Transceiver Wizard or an associated core containing the component. Xilinx does not recommend direct instantiation of this component.

For More Information

- See the Spartan-6 FPGA RocketIO GTP Transceivers User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

IBUF

Primitive: Input Buffer

Introduction

This design element is automatically inserted (inferred) by the synthesis tool to any signal directly connected to a top-level input or in-out port of the design. You should generally let the synthesis tool infer this buffer. However, it can be instantiated into the design if required. In order to do so, connect the input port (I) directly to the associated top-level input or in-out port, and connect the output port (O) to the logic sourced by that port. Modify any necessary generic maps (VHDL) or named parameter value assignment (Verilog) in order to change the default behavior of the component.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Buffer output</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Buffer input</td>
</tr>
</tbody>
</table>

Design Entry Method

| Instantiation | Yes |
| Inference     | Recommended |
| CORE Generator™ and wizards | No |
| Macro support | No |

This element is usually inferred by the synthesis tool for any specified top-level input port to the design, and therefore it is generally not necessary to specify the element in source code. However, if desired, this element may be manually instantiated by copying the instantiation code from below and pasting it into the top-level entity/module of your code. Xilinx recommends that you put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level input port of the design and the O port to the logic in which this input is to source. Specify the desired generic/defparam values in order to configure the proper behavior of the buffer.

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet.</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- IBUF: Single-ended Input Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

IBUF_inst : IBUF
generic map (    
    IBUF_LOW_PWR => TRUE, -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
    IOSTANDARD => "DEFAULT")
port map (    
    O => O, -- Buffer output
    I => I -- Buffer input (connect directly to top-level port)
);

-- End of IBUF_inst instantiation
```

Verilog Instantiation Template

```
// IBUF: Single-ended Input Buffer
// Spartan-6

IBUF #(    
    .IOSTANDARD("DEFAULT") -- Specify the input I/O standard
)IBUF_inst (    
    .O(O), -- Buffer output
    .I(I) -- Buffer input (connect directly to top-level port)
);

// End of IBUF_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
IBUFDS

Primitive: Differential Signaling Input Buffer

Introduction

This design element is an input buffer that supports low-voltage, differential signaling. In IBUFDS, a design level interface signal is represented as two distinct ports (I and IB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>IB</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Diff_p Buffer Input</td>
</tr>
<tr>
<td>IB</td>
<td>Input</td>
<td>1</td>
<td>Diff_n Buffer Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Buffer Output</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level "master" input port of the design, the IB port to the top-level "slave" input port, and the O port to the logic in which this input is to source. Specify the desired generic/defparam values in order to configure the proper behavior of the buffer.
Chapter 4: About Design Elements

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIFF_TERM</td>
<td>Boolean</td>
<td>TRUE or FALSE</td>
<td>FALSE</td>
<td>Enables the built-in differential termination resistor.</td>
</tr>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet.</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- IBUFDS: Differential Input Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

IBUFDS_inst : IBUFDS
generic map (    
  DIFF_TERM => FALSE, -- Differential Termination    
  I buf_LOW_PWR => TRUE, -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards    
  IOSTANDARD => "DEFAULT")
port map (      
  O => O, -- Buffer output      
  I => I, -- Diff_p buffer input (connect directly to top-level port)      
  IB => IB -- Diff_n buffer input (connect directly to top-level port)    
);

-- End of IBUFDS_inst instantiation
```

Verilog Instantiation Template

```verilog
// IBUFDS: Differential Input Buffer
// Spartan-6

IBUFDS #(
  .DIFF_TERM("FALSE"), // Differential Termination
  .IOSTANDARD("DEFAULT") // Specify the input I/O standard
) IBUFDS_inst (    
  .O(O), // Buffer output    
  .I(I), // Diff_p buffer input (connect directly to top-level port)    
  .IB(IB) // Diff_n buffer input (connect directly to top-level port)    
);

// End of IBUFDS_inst instantiation
```

For More Information

- See the Spartan-6 FPGA SelectIO Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
**IBUFDS_DIFF_OUT**

**Primitive: Signaling Input Buffer with Differential Output**

![IBUFDS_DIFF_OUT](image.png)

**Introduction**

This design element is an input buffer that supports differential signaling. In IBUFDS_DIFF_OUT, a design level interface signal is represented as two distinct ports (I and IB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). The IBUFDS_DIFF_OUT differs from the IBUFDS in that it allows internal access to both phases of the differential signal. Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>IB</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

It is suggested to put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level "master" input port of the design, the IB port to the top-level "slave" input port, and the O and OB ports to the logic in which this input is to source. Specify the desired generic/parameter values in order to configure the proper behavior of the buffer.

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet.</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>
**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- IBUFDS_DIFF_OUT: Differential Input Buffer with Differential Output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

IBUFDS_DIFF_OUT_inst : IBUFDS_DIFF_OUT
generic map (
  DIFF_TERM => FALSE, -- Differential Termination
  IOSTANDARD => "DEFAULT") -- Specify the input I/O standard
port map ( 
  O => O, -- Buffer diff_p output
  OB => OB, -- Buffer diff_n output
  I => I, -- Diff_p buffer input (connect directly to top-level port)
  IB => IB -- Diff_n buffer input (connect directly to top-level port)
);

-- End of IBUFDS_DIFF_OUT_inst instantiation
```

**Verilog Instantiation Template**

```verilog
// IBUFDS_DIFF_OUT: Differential Input Buffer with Differential Output
// Spartan-6

IBUFDS_DIFF_OUT #(
  .DIFF_TERM("FALSE"), // Differential Termination, "TRUE"/"FALSE"
  .IOSTANDARD("DEFAULT") // Specify the input I/O standard
) IBUFDS_DIFF_OUT_inst ( 
  .O(O), // Buffer diff_p output
  .OB(OB), // Buffer diff_n output
  .I(I), // Diff_p buffer input (connect directly to top-level port)
  .IB(IB) // Diff_n buffer input (connect directly to top-level port)
);

// End of IBUFDS_DIFF_OUT_inst instantiation
```

**For More Information**

- See the [Spartan-6 FPGA Clocking Resources User Guide](#).
- See the [Spartan-6 FPGA SelectIO Resources User Guide](#).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](#).
Introduction

The IBUFG is a dedicated input to the device which should be used to connect incoming clocks to the FPGA’s global clock routing resources. The IBUFG provides dedicated connections to the DCM, PLL, or BUFG resources, providing the minimum amount of clock delay and jitter to the device. The IBUFG input can only be driven by the global clock (GC) pins.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock Buffer output</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Clock Buffer input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>Recommended</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macro support</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- IBUFG: Single-ended global clock input buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

IBUFG_inst : IBUFG
generic map (  
    IBUF_LOW_PWR => TRUE, -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards 
    IOSTANDARD => "DEFAULT")
port map (   
    O => O, -- Clock buffer output 
    I => I -- Clock buffer input (connect directly to top-level port)
);

-- End of IBUFG_inst instantiation
```
Verilog Instantiation Template

// IBUF: Single-ended global clock input buffer
// Spartan-6

IBUF #(
    .IOSTANDARD("DEFAULT") // Specify the input I/O standard
) IBUF_inst (
    .O(O), // Clock buffer output
    .I(I) // Clock buffer input (connect directly to top-level port)
);

// End of IBUF_inst instantiation

For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
IBUFGDS

Primitive: Differential Signaling Dedicated Input Clock Buffer and Optional Delay

Introduction

This design element is a dedicated differential signaling input buffer for connection to the clock buffer (BUFG) or MMCM. In IBUFGDS, a design-level interface signal is represented as two distinct ports (I and IB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components. Also available is a programmable delay is to assist in the capturing of incoming data to the device.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>IB</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Clock Buffer output</td>
</tr>
<tr>
<td>IB</td>
<td>Input</td>
<td>1</td>
<td>Diff_n Clock Buffer Input</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Diff_p Clock Buffer Input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level "master" input port of the design, the IB port to the top-level "slave" input port and the O port to an MMCM, BUFG or logic in which this input is to source. Some synthesis tools infer the BUFG automatically if necessary, when connecting an IBUFG to the clock resources of the FPGA. Specify the desired generic/defparam values in order to configure the proper behavior of the buffer.
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- IBUFDS: Differential Global Clock Input Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

IBUFDS_inst : IBUFDS
generic map (  
    DIFF_TERM => FALSE, -- Differential Termination  
    IBF_LOW_PWR => TRUE, -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards  
    IOSTANDARD => "DEFAULT")
port map (  
    O => O, -- Clock buffer output
    I => I, -- Diff_p clock buffer input (connect directly to top-level port)
    IB => IB -- Diff_n clock buffer input (connect directly to top-level port)
);

-- End of IBUFDS_inst instantiation

Verilog Instantiation Template

// IBUFDS: Differential Global Clock Input Buffer
// Spartan-6

IBUFDS #(  
    .DIFF_TERM("FALSE"), // Differential Termination  
    .IOSTANDARD("DEFAULT")  // Specify the input I/O standard
) IBUFDS_inst (  
    .O(O), // Clock buffer output
    .I(I), // Diff_p clock buffer input (connect directly to top-level port)
    .IB(IB) // Diff_n clock buffer input (connect directly to top-level port)
);

// End of IBUFDS_inst instantiation

For More Information

- See the Spartan-6 FPGA SelectIO Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
IBUFGDS_DIFF_OUT

Primitive: Differential Signaling Input Buffer with Differential Output

Introduction

This design element is an input buffer that supports differential signaling. In IBUFGDS_DIFF_OUT, a design level interface signal is represented as two distinct ports (I and IB), one deemed the “master” and the other the “slave.” The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). The IBUFGDS_DIFF_OUT differs from the IBUFDS in that it allows internal access to both phases of the differential signal. Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>IB</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Diff_p Buffer Input (connect to top-level port in the design).</td>
</tr>
<tr>
<td>IB</td>
<td>Input</td>
<td>1</td>
<td>Diff_n Buffer Input (connect to top-level port in the design).</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Diff_p Buffer Output.</td>
</tr>
<tr>
<td>OB</td>
<td>Output</td>
<td>1</td>
<td>Diff_n Buffer Output.</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

It is suggested to put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level “master” input port of the design, the IB port to the top-level “slave” input port, and the O and OB ports to the logic in which this input is to source. Specify the desired generic/parameter values in order to configure the proper behavior of the buffer.
## Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
<tr>
<td>DIFF_TERM</td>
<td>Boolean</td>
<td>TRUE, FALSE</td>
<td>FALSE</td>
<td>Specifies the use of the internal differential termination resistance.</td>
</tr>
</tbody>
</table>

## VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- IBUFGDS_DIFF_OUT: Differential Global Clock Buffer with Differential Output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
IBUFGDS_DIFF_OUT_inst : IBUFGDS_DIFF_OUT
generic map (
  DIFF_TERM => FALSE, -- Differential Termination
  IOSTANDARD => "DEFAULT") -- Specify the input I/O standard
port map (
  O => O, -- Buffer diff_p output
  OB => OB, -- Buffer diff_n output
  I => I, -- Diff_p buffer input (connect directly to top-level port)
  IB => IB -- Diff_n buffer input (connect directly to top-level port)
);

-- End of IBUFGDS_DIFF_OUT_inst instantiation
```

## Verilog Instantiation Template

```verilog
// IBUFGDS_DIFF_OUT: Differential Global Clock Buffer with Differential Output
// Spartan-6
IBUFGDS_DIFF_OUT #(
  .DIFF_TERM("FALSE"), // Differential Termination, "TRUE"/"FALSE"
  .IOSTANDARD("DEFAULT") // Specify the input I/O standard
) IBUFGDS_DIFF_OUT_inst (O(O), // Buffer diff_p output
  .OB(OB), // Buffer diff_n output
  .I(I), // Diff_p buffer input (connect directly to top-level port)
  .IB(IB) // Diff_n buffer input (connect directly to top-level port)
);

// End of IBUFGDS_DIFF_OUT_inst instantiation
```

## For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
ICAP_SPARTAN6

Primitive: Internal Configuration Access Port

Introduction

This design element gives you access to the configuration functions of the FPGA from the FPGA fabric. Using this component, commands and data can be written to and read from the configuration logic of the FPGA array. Since the improper use of this function can have a negative effect on the functionality and reliability of the FPGA, you should not use this element unless you are very familiar with its capabilities.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>Output</td>
<td>1</td>
<td>Busy/Ready output.</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Active-Low ICAP Enable input.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock input.</td>
</tr>
<tr>
<td>I[15:0]</td>
<td>Input</td>
<td>16</td>
<td>Configuration data input bus.</td>
</tr>
<tr>
<td>O[15:0]</td>
<td>Output</td>
<td>16</td>
<td>Configuration data output bus.</td>
</tr>
<tr>
<td>WRITE</td>
<td>Input</td>
<td>1</td>
<td>Read/Write control input.</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE_ID</td>
<td>Hexadecimal</td>
<td>32h02000093, 32h0200E093, 32h0201D093, 32h0202E093, 32h0203D093, 32h0201093, 32h0202093, 32h0204093, 32h0208093, 32h0211093, 32h02024093,</td>
<td>32h0200093</td>
<td>Specifies the pre-programmed Device ID value.</td>
</tr>
</tbody>
</table>
### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- ICAP_SPARTAN6: Internal Configuration Access Port
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
ICAP_SPARTAN6_inst : ICAP_SPARTAN6
  generic map (
    DEVICE_ID => X"2000093", -- Specifies the pre-programmed Device ID value
    SIM_CFG_FILE_NAME => "NONE" -- Specifies the Raw Bitstream (RBT) file to be parsed by the simulation model
  )
  port map (
    BUSY => BUSY, -- 1-bit output: Busy/Ready output
    O => O, -- 16-bit output: Configuration data output bus
    CE => CE, -- 1-bit input: Active-Low ICAP Enable input
    CLK => CLK, -- 1-bit input: Clock input
    I => I, -- 16-bit input: Configuration data input bus
    WRITE => WRITE -- 1-bit input: Read/Write control input
  );

-- End of ICAP_SPARTAN6_inst instantiation
```

### Verilog Instantiation Template

```verilog
// ICAP_SPARTAN6: Internal Configuration Access Port
// Spartan-6
ICAP_SPARTAN6 #=>
  .DEVICE_ID(0'h2000093), // Specifies the pre-programmed Device ID value
  .SIM_CFG_FILE_NAME("NONE") // Specifies the Raw Bitstream (RBT) file to be parsed by the simulation model
)
ICAP_SPARTAN6_inst ( // End of ICAP_SPARTAN6_inst instantiation
  .BUSY(BUSY), // 1-bit output: Busy/Ready output
  .O(O), // 16-bit output: Configuration data output bus
  .CE(CE), // 1-bit input: Active-Low ICAP Enable input
  .CLK(CLK), // 1-bit input: Clock input
  .I(I), // 16-bit input: Configuration data input bus
  .WRITE(WRITE) // 1-bit input: Read/Write control input
);

// End of ICAP_SPARTAN6_inst instantiation
```

### For More Information
- See the [Spartan-6 FPGA Configuration User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).

---

**Chapter 4: About Design Elements**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM_CFG_FILE_NAME</td>
<td>String</td>
<td>String representing file name and location</td>
<td>None</td>
<td>Specifies the Raw Bitstream (RBT) file to be parsed by the simulation model.</td>
</tr>
</tbody>
</table>
IDDR2

Primitive: Double Data Rate Input D Flip-Flop with Optional Data Alignment, Clock Enable and Programmable Synchronous or Asynchronous Set/Reset

Introduction

This design element is a dedicated input register designed to receive external dual data rate (DDR) signals into Xilinx® FPGAs. The IDDR2 requires two clocks to be connected to the component, C0 and C1, so that data is captured at the positive edge of both C0 and C1 clocks. The IDDR2 features an active high clock enable port, CE, which be used to suspend the operation of the registers, and both set and reset ports that be configured to be synchronous or asynchronous to the respective clocks. The IDDR2 has an optional alignment feature that allows both output data ports to the component to be aligned to a single clock.

Logic Table

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>×</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Set/Reset can be synchronous via SRTYPE value

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

To change the default behavior of the IDDR2, modify attributes via the generic map (VHDL) or named parameter value assignment (Verilog) as a part of the instantiated component. The IDDR2 can be connected directly to a top-level input port in the design, where an appropriate input buffer can be inferred, or directly to an instantiated IBUF, IOBUF, IBUFDS or IOBUFDS. All inputs and outputs of this component should either be connected or properly tied off.
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR_ALIGNMENT</td>
<td>String</td>
<td>“NONE”, “C0”,</td>
<td>“NONE”</td>
<td>Sets the output alignment more for the DDR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“C1”</td>
<td></td>
<td>• NONE - Makes the data available on the Q0 and Q1 outputs shortly after the corresponding C0 or C1 positive clock edge.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• C0 – Makes the data on both Q0 and Q1 align to the positive edge of the C0 clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• C1 - Makes the data on both Q0 and Q1 align to the positive edge of the C1 clock.</td>
</tr>
<tr>
<td>INIT_Q0</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Sets initial state of the Q0 output to 0 or 1.</td>
</tr>
<tr>
<td>INIT_Q1</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Sets initial state of the Q1 output to 0 or 1.</td>
</tr>
<tr>
<td>SRTYPE</td>
<td>String</td>
<td>“SYNC”, “ASYNC”</td>
<td>“SYNC”</td>
<td>Specifies “SYNC” or “ASYNC” set/reset.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;
-- IDDR2: Input Double Data Rate Input Register with Set, Reset
-- and Clock Enable.
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
IDDR2_inst : IDDR2
generic map(
  DDR_ALIGNMENT => "NONE", -- Sets output alignment to "NONE", "C0", "C1"
  INIT_Q0 => '0', -- Sets initial state of the Q0 output to '0' or '1'
  INIT_Q1 => '0', -- Sets initial state of the Q1 output to '0' or '1'
  SRTYPE => "SYNC") -- Specifies "SYNC" or "ASYNC" set/reset
port map ( 
  Q0 => Q0, -- 1-bit output captured with C0 clock
  Q1 => Q1, -- 1-bit output captured with C1 clock
  C0 => C0, -- 1-bit clock input
  C1 => C1, -- 1-bit clock input
  CE => CE, -- 1-bit clock enable input
  D => D, -- 1-bit data input
  R => R, -- 1-bit reset input
  S => S -- 1-bit set input
);
-- End of IDDR2_inst instantiation

Verilog Instantiation Template

// IDDR2: Input Double Data Rate Input Register with Set, Reset
// and Clock Enable.
// Spartan-6
IDDR2 #(
  .DDR_ALIGNMENT("NONE"), // Sets output alignment to "NONE", "C0" or "C1"
  .INIT_Q0(1'b0), // Sets initial state of the Q0 output to 1'b0 or 1'b1
  .INIT_Q1(1'b0), // Sets initial state of the Q1 output to 1'b0 or 1'b1
  .SRTYPE("SYNC") // Specifies "SYNC" or "ASYNC" set/reset
) IDDR2_inst ( 
  .Q0(Q0), // 1-bit output captured with C0 clock
  .Q1(Q1), // 1-bit output captured with C1 clock
  .C0(C0), // 1-bit clock input
  .C1(C1), // 1-bit clock input
  .D(D), // 1-bit data input
  .R(R), // 1-bit reset input
  .S(S), // 1-bit set input
  .CE(CE) // 1-bit clock enable input
);
.Cl(C1), // 1-bit clock input
.CE(CE), // 1-bit clock enable input
.D(D), // 1-bit DDR data input
.R(R), // 1-bit reset input
.S(S) // 1-bit set input
);

// End of IDD2_inst instantiation

**For More Information**

- See the *Spartan-6 FPGA Configurable Logic Block User Guide*.
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics*. 
IOBUF

Primitive: Bi-Directional Buffer

Introduction
The design element is a bidirectional single-ended I/O Buffer used to connect internal logic to an external bidirectional pin.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Bidirectional</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>I</td>
<td>IO</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Buffer output</td>
</tr>
<tr>
<td>IO</td>
<td>Inout</td>
<td>1</td>
<td>Buffer inout</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Buffer input</td>
</tr>
<tr>
<td>T</td>
<td>Input</td>
<td>1</td>
<td>3-State enable input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
## Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRIVE</td>
<td>Integer</td>
<td>2, 4, 6, 8, 12, 16, 24</td>
<td>12</td>
<td>Selects output drive strength (mA) for the SelectIO™ buffers that use the LVTTL, LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, or LVCMOS33 interface I/O standard.</td>
</tr>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
<tr>
<td>SLEW</td>
<td>String</td>
<td>&quot;SLOW&quot;, &quot;FAST&quot;, &quot;QUIETIO&quot;</td>
<td>&quot;SLOW&quot;</td>
<td>Sets the output rise and fall time. See the Data Sheet for recommendations of the best setting for this attribute.</td>
</tr>
</tbody>
</table>

## VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- IOBUF: Single-ended Bi-directional Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

IOBUF_inst : IOBUF
generic map (  
  DRIVE => 12,
  IOSTANDARD => "DEFAULT",
  SLEW => "SLOW")
port map (  
  O => O,  -- Buffer output
  IO => IO,  -- Buffer inout port (connect directly to top-level port)
  I => I,  -- Buffer input
  T => T  -- 3-state enable input, high=input, low=output
);

-- End of IOBUF_inst instantiation
```

## Verilog Instantiation Template

```
// IOBUF: Single-ended Bi-directional Buffer
// Spartan-6

IOBUF #(  
  .DRIVE(12),  // Specify the output drive strength
  .IOSTANDARD("DEFAULT"),  // Specify the I/O standard
  .SLEW("SLOW")  // Specify the output slew rate
) IOBUF_inst (  
  .O(IO),  // Buffer output
  .IO(IO),  // Buffer inout port (connect directly to top-level port)
  .I(I),  // Buffer input
  .T(T)  // 3-state enable input, high=input, low=output
);

// End of IOBUF_inst instantiation
```

## For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](#).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](#).
IOBUFDS

Primitive: 3-State Differential Signaling I/O Buffer with Active Low Output Enable

Introduction

The design element is a bidirectional buffer that supports low-voltage, differential signaling. For the IOBUFDS, a design level interface signal is represented as two distinct ports (IO and IOB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components. Also available is a programmable delay is to assist in the capturing of incoming data to the device.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Bidirectional</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>T</td>
<td>IO</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Buffer output</td>
</tr>
<tr>
<td>IO</td>
<td>Inout</td>
<td>1</td>
<td>Diff_p inout</td>
</tr>
<tr>
<td>IOB</td>
<td>Inout</td>
<td>1</td>
<td>Diff_n inout</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Buffer input</td>
</tr>
<tr>
<td>T</td>
<td>Input</td>
<td>1</td>
<td>3-state enable input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- IOBUFDS: Differential Bi-directional Buffer
-- Spartan-3/3E/3A
-- Xilinx HDL Libraries Guide, version 13.3
IOBUFDS_inst : IOBUFDS
  generic map
    (IOSTANDARD => "BLVDS_25")
  port map (
    O => O,  -- Buffer output
    IO => IO,  -- Diff_p inout (connect directly to top-level port)
    IOB => IOB,  -- Diff_n inout (connect directly to top-level port)
    I => I,  -- Buffer input
    T => T  -- 3-state enable input, high=input, low=output
  );

-- End of IOBUFDS_inst instantiation
```

Verilog Instantiation Template

```
// IOBUFDS: Differential Bi-directional Buffer
// Spartan-6

IOBUFDS #(  
  .IOSTANDARD("BLVDS_25")  // Specify the I/O standard
) IOBUFDS_inst {
  .O(O),  // Buffer output
  .IO(IO),  // Diff_p inout (connect directly to top-level port)
  .IOB(IOB),  // Diff_n inout (connect directly to top-level port)
  .I(I),  // Buffer input
  .T(T)  // 3-state enable input, high=input, low=output
};

// End of IOBUFDS_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
**IODelay2**

**Primitive: Input and Output Fixed or Variable Delay Element**

**Introduction**

This design element can be used to provide a fixed delay or an adjustable delay to the input path and a fixed delay for the output path of the Spartan®-6 FPGA. This delay can be useful for data alignment of incoming or outgoing data to/from the chip. When used in variable mode, the input path can be adjusted for increasing and decreasing amounts of delay. The output delay path is only available in a fixed delay. The IODELAY can also be used to add additional static or variable delay to an internal path (within the FPGA fabric). However, when IODELAY is used that way, this device is no longer available to the associated I/O for input or output path delays.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>Output</td>
<td>1</td>
<td>Busy after CAL.</td>
</tr>
<tr>
<td>CAL</td>
<td>Input</td>
<td>1</td>
<td>Initiate calibration input.</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Enable increment/decrement.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>IODELAY Clock input.</td>
</tr>
<tr>
<td>DATAOUT</td>
<td>Output</td>
<td>1</td>
<td>Delayed Data output from input port (connect to input datapath logic, can only route to a register in ILOGIC).</td>
</tr>
<tr>
<td>DATAOUT2</td>
<td>Output</td>
<td>1</td>
<td>Delayed Data output from input port (connect to input datapath logic, can route to fabric).</td>
</tr>
<tr>
<td>DOUT</td>
<td>Output</td>
<td>1</td>
<td>Delayed Data Output to IOB.</td>
</tr>
<tr>
<td>IDATAIN</td>
<td>Input</td>
<td>1</td>
<td>Data Signal from IOB.</td>
</tr>
<tr>
<td>INC</td>
<td>Input</td>
<td>1</td>
<td>Increment / Decrement Input.</td>
</tr>
<tr>
<td>IOCLK0</td>
<td>Input</td>
<td>1</td>
<td>Optionally Invertible I/O clock inputs.</td>
</tr>
<tr>
<td>IOCLK1</td>
<td>Input</td>
<td>1</td>
<td>Optionally Invertible I/O clock inputs.</td>
</tr>
</tbody>
</table>
### Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>Yes</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTER_WRAP_AROUND</td>
<td>String</td>
<td>“WRAPAROUND”, “STAY_AT_LIMIT”</td>
<td>“WRAPAROUND”</td>
<td>Sets behavior when tap count exceeds max or min, depending on whether tap setting is being incremented or decremented.</td>
</tr>
<tr>
<td>DATA_RATE</td>
<td>String</td>
<td>“SDR”, “DDR”</td>
<td>“SDR”</td>
<td>Single Data Rate or Double Data Rate operation.</td>
</tr>
</tbody>
</table>
| DELAY_SRC                  | String  | “IO”, “IDATAIN”, “ODATAIN”         | “IO”       | • ODATAIN indicates delay source is the ODATAIN pin from the OSERDES or OLOGIC.  
• IDATAIN indicates the delay source is from the IDATAIN pin; one of the dedicated IOB (P/N) Pads.  
• IO means that the signal source switches between IDATAIN and ODATAIN depending on the sense of the T (tristate) input. |
| IDELAY_MODE                | String  | “NORMAL”, “PCI”                    | “NORMAL”   | Do not specify or modify this attribute.                                    |
## Chapter 4: About Design Elements

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDELAY_TYPE</td>
<td>String</td>
<td>&quot;DEFAULT&quot;, &quot;DIFF_PHASE_DETECTOR&quot;, &quot;FIXED&quot;, &quot;VARIABLE_FROM_HALF_MAX&quot;, &quot;VARIABLE_FROM_ZERO&quot;</td>
<td>&quot;DEFAULT&quot;</td>
<td>Delay Type. VARIABLE refers to the customer calibrated delay mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• DEFAULT utilizes physical chip settings for best approximation of zero hold time programming.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• VARIABLE_FROM_ZERO and VARIABLE_FROM_HALF_MAX refer to the reset behavior.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• DIFF_PHASE_DETECTOR is a special mode where the master and slave IODELAY2s are cascaded.</td>
</tr>
<tr>
<td>IDELAY_VALUE</td>
<td>Integer</td>
<td>0 to 255</td>
<td>0</td>
<td>Delay tap value for IDELAY Mode.</td>
</tr>
<tr>
<td>IDELAY2_VALUE</td>
<td>Integer</td>
<td>0 to 255</td>
<td>0</td>
<td>Delay tap value for IDELAY Mode. Only used when IDELAY_MODE is set to PCI.</td>
</tr>
<tr>
<td>ODELAY_VALUE</td>
<td>Integer</td>
<td>0 to 255</td>
<td>0</td>
<td>Delay tap value for ODELAY Mode.</td>
</tr>
<tr>
<td>SERDES_MODE</td>
<td>String</td>
<td>&quot;NONE&quot;, &quot;MASTER&quot;, &quot;SLAVE&quot;</td>
<td>&quot;NONE&quot;</td>
<td>Specify whether the ISERDES2 is operating in master or slave modes when cascaded width expansion.</td>
</tr>
<tr>
<td>SIM_TAPDELAY_VALUE</td>
<td>Integer</td>
<td>10 to 90</td>
<td>75</td>
<td>A simulation only attribute. Allows setting the nominal tap delay to different settings for simulation.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- IDELAY2: Input and Output Fixed or Variable Delay Element
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

IODELAY2_inst : IODELAY2
generic map (
  COUNTER_WRAPAROUND => "WRAPAROUND", -- "STAY_AT_LIMIT" or "WRAPAROUND"
  DATA_RATE => "SDR", -- "SDR" or "DDR"
  DELAY_SRC => "IO", -- "IO", "ODATAIN" or "IDATAIN"
  IDELAY2_VALUE => 0,
  IDELAY_MODE => "NORMAL", -- "NORMAL" or "PCI"
  IDELAY_TYPE => "DEFAULT", -- "FIXED", "DEFAULT", "VARIABLE_FROM_ZERO", "VARIABLE_FROM_HALF_MAX"
  -- or "DIFF_PHASE_DETECTOR"
  IDELAY_VALUE => 0,
  ODELAY_VALUE => 0,
  SERDES_MODE => "NONE", -- "NONE", "MASTER" or "SLAVE"
  SIM_TAPDELAY_VALUE => 75 -- Per tap delay used for simulation in ps
) port map (BUSY => BUSY, -- 1-bit output: Busy output after CAL
  DATAOUT => DATAOUT, -- 1-bit output: Delayed data output to ISERDES/input register
```

Spartan-6 Libraries Guide for HDL Designs

132  www.xilinx.com

UG615 (v 13.3) October 26, 2011
Verilog Instantiation Template

// IODELAY2: Input and Output Fixed or Variable Delay Element
// Spartan-6

IODELAY2 #(
  .COUNTER_WRAPAROUND("WRAPAROUND"), // "STAY_AT_LIMIT" or "WRAPAROUND"
  .DATA_RATE("SDR"), // "SDR" or "DDR"
  .DELAY_SRC("IO"), // "IO", "ODATAIN" or "IDATAIN"
  .IDELAY2_VALUE(0), // Delay value when IDELAY_MODE="PCI" (0-255)
  .IDELAY_MODE("NORMAL"), // "NORMAL" or "PCI"
  .IDELAY_TYPE("DEFAULT"), // "FIXED", "DEFAULT", "VARIABLE_FROM_ZERO", "VARIABLE_FROM_HALF_MAX" // or "DIFF_PHASE_DETECTOR"
  .IDELAY_VALUE(0), // Amount of taps for fixed input delay (0-255)
  .ODELAY_VALUE(0), // Amount of taps fixed output delay (0-255)
  .SERDES_MODE("NONE"), // "NONE", "MASTER" or "SLAVE"
  .SIM_TAPDELAY_VALUE(75) // Per tap delay used for simulation in ps
)

IODELAY2_inst ( // 1-bit output: Busy output after CAL
  BUSY(BUSY),
  DATAOUT1(DATAOUT), // 1-bit output: Delayed data output to ISERDES/input register
  DATAOUT2(DATAOUT2), // 1-bit output: Delayed data output to general FPGA fabric
  DOUT(DOUT), // 1-bit output: Delayed data output
  TOUT(TOUT), // 1-bit output: Delayed 3-state output
  CAL(CAL), // 1-bit input: Initiate calibration input
  CE(CE), // 1-bit input: Enable INC input
  CLK(CLK), // 1-bit input: Clock input
  IDATAIN(IDATAIN), // 1-bit input: Data input (connect to top-level port or I/O buffer)
  INC(INC), // 1-bit input: Increment / decrement input
  IOCLK0(IOCLK0), // 1-bit input: Input from the I/O clock network
  IOCLK1(IOCLK1), // 1-bit input: Input from the I/O clock network
  ODATAIN(ODATAIN), // 1-bit output: Input data output from output register or OSERDES2.
  RST(RST), // 1-bit input: Reset to zero or 1/2 of total delay period
  .T(T) // 1-bit input: 3-state input signal
); // End of IODELAY2_inst instantiation

For More Information

- See the Spartan-6 FPGA SelectIO Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
IODRP2

Primitive: I/O Control Port

Introduction

Xilinx does not support the use of this element.

For More Information

- See the *Spartan-6 FPGA SelectIO Resources User Guide*.
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics*. 
IODRP2_MCB

Primitive: I/O Control Port for the Memory Controller Block

Introduction

The IODRP2_MCB is a component used by the Memory Interface Generator (MIG) core in conjunction with the MCB block to implement external memory interfaces. The use of this block outside of MIG is not supported.

For More Information

- See the Xilinx Memory Interface Generator (MIG) User Guide
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
ISERDES2

Primitive: Input SERial/DESerializer.

Introduction

Each IOB contains an input deserializer block that can be instantiated in a design by using the ISERDES2 primitive. ISERDES2 allows serial-to-parallel conversion with SerDes ratios of 1:2, 1:3 and 1:4. The SerDes ratio is the ratio between the high speed I/O clock that is capturing data, and the slower internal global clock used for processing the parallel data. For example, with a single-rate I/O clock running at 500 MHz to receive data at 500 Mb/s, the ISERDES2 transfers four bits of data at one quarter of the rate (125 MHz) to the FPGA logic. When using differential inputs, the two ISERDES2 primitives associated with the two IOBs can be cascaded to allow higher SerDes ratios of 1:5, 1:6, 1:7, and 1:8. Each ISERDES2 also contains logic to word-align the parallel data, as required by the designer, by performing a Bitslip operation.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITSLIP</td>
<td>Input</td>
<td>1</td>
<td>Invoke Bitslip when High. Bitslip operation can be used with any DATA_WIDTH, cascaded or not. The amount of Bitslip is fixed by the DATA_WIDTH selection.</td>
</tr>
<tr>
<td>CE0</td>
<td>Input</td>
<td>1</td>
<td>Clock enable input for final (global clock driven) register.</td>
</tr>
<tr>
<td>CFB0</td>
<td>Output</td>
<td>1</td>
<td>Feed-through route to allow a PLL or DCM generated clock to feed back to the PLL or DCM through a BUFIQ2FB.</td>
</tr>
<tr>
<td>CFB1</td>
<td>Output</td>
<td>1</td>
<td>Secondary feed-through route to allow a PLL or DCM generated clock to feed back to the PLL or DCM through a BUFIQ2FB.</td>
</tr>
<tr>
<td>CLKDIV</td>
<td>Input</td>
<td>1</td>
<td>Global clock network input. This is the clock for the FPGA logic domain.</td>
</tr>
<tr>
<td>CLK0</td>
<td>Input</td>
<td>1</td>
<td>I/O clock network input. Optionally invertible. This is the primary clock input used when the clock doubler circuit is not engaged.(see DATA_RATE attribute).</td>
</tr>
<tr>
<td>Port</td>
<td>Type</td>
<td>Width</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CLK1</td>
<td>Input</td>
<td>1</td>
<td>I/O clock network input. Optionally invertible. This secondary clock input is only used when the clock doubler is engaged (see DATA_RATE attribute).</td>
</tr>
<tr>
<td>D</td>
<td>Input</td>
<td>1</td>
<td>Input data. This is the data input after being delayed by the IODELAY2 block.</td>
</tr>
<tr>
<td>DFB</td>
<td>Output</td>
<td>1</td>
<td>Feed-through route to allow an input clock that has been delayed in an IODELAY2 element to be forwarded to a DCM, PLL, or BUFG through a BUFIO2.</td>
</tr>
<tr>
<td>FABRICOUT</td>
<td>Output</td>
<td>1</td>
<td>Asynchronous data for use in the FPGA logic.</td>
</tr>
<tr>
<td>INCDEC</td>
<td>Output</td>
<td>1</td>
<td>Output of phase detector in master mode (dummy in slave mode). Indicates to the FPGA logic whether the received data was sampled early or late.</td>
</tr>
<tr>
<td>IOCE</td>
<td>Input</td>
<td>1</td>
<td>Data strobe signal derived from BUFIO CE. Strobes data capture to be correctly timed with respect to the I/O and global clocks for the SerDes mode selected.</td>
</tr>
<tr>
<td>Q1 - Q4</td>
<td>Output</td>
<td>1</td>
<td>Registered output to fabric.</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Asynchronous reset only.</td>
</tr>
<tr>
<td>SHIFTIN</td>
<td>Input</td>
<td>1</td>
<td>Cascade-in signal for master/slave I/O. Used when master and slave sites are used together for DATA_WIDTHs greater than four. When the block is a master, it transfers data in for use in the phase-detector mode. When the block is a slave, it transfers serial data in to become parallel data.</td>
</tr>
<tr>
<td>SHIFTOUT</td>
<td>Output</td>
<td>1</td>
<td>Cascade-out signal for master/slave I/O. In slave mode, it is used to send sampled data from the slave. In master mode, it sends serial data from the fourth stage of the input shift register to the slave.</td>
</tr>
<tr>
<td>VALID</td>
<td>Output</td>
<td>1</td>
<td>Output of the phase detector in master mode (dummy in slave mode). If the input data contains no edges (no information for the phase detector to work with) the VALID signal transitions Low to indicate that the FPGA logic should ignore the INCDEC signal.</td>
</tr>
</tbody>
</table>

### Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>Recommended</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITSLIP_ENABLE</td>
<td>Boolean</td>
<td>FALSE, TRUE</td>
<td>FALSE</td>
<td>Enables or disables the Bitslip function controlled by the BITSLIP input pin. The number of bits slipped is a function of the DATA_WIDTH selected. When disabled, the Bitslip CE always remains at the default value of one I/O clock before the IOCE clock enable.</td>
</tr>
<tr>
<td>DATA_RATE</td>
<td>String</td>
<td>“SDR”, “DDR”</td>
<td>“SDR”</td>
<td>Data rate setting. The DDR clock can be supplied by separate I/O clocks or by a single I/O clock. If two clocks are supplied, they must be approximately 180° out of phase.</td>
</tr>
<tr>
<td>DATA_WIDTH</td>
<td>Integer</td>
<td>1, 2, 3, 4, 5, 6, 7, 8</td>
<td>1</td>
<td>Data width. Defines the parallel data output width of the serial-to-parallel converter. Values greater than four are only valid when two ISERDES2 blocks are cascaded. In this case, the same value should be applied to both the master and slave blocks.</td>
</tr>
<tr>
<td>INTERFACE_TYPE</td>
<td>String</td>
<td>“NETWORKING”, “NETWORKING_PIPELINED”, “RETIMED”</td>
<td>“NETWORKING”</td>
<td>Selects mode of operation and determines which set of parallel data is available to the FPGA logic.</td>
</tr>
<tr>
<td>SERDES_MODE</td>
<td>String</td>
<td>“NONE”, “MASTER”, “SLAVE”</td>
<td>“NONE”</td>
<td>Indicates if the ISERDES is being used alone, or as a master or slave, when two ISERDES2 blocks are cascaded.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- ISERDES2: Input SERial/DESerializer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

ISERDES2_inst : ISERDES2
generic map (   BITSLIP_ENABLE => FALSE, -- Enable Bitslip Functionality (TRUE/FALSE)
                DATA_RATE => "SDR", -- Data-rate ("SDR" or "DDR")
                DATA_WIDTH => 1, -- Parallel data width selection (2-8)
                INTERFACE_TYPE => "NETWORKING", -- "NETWORKING", "NETWORKING_PIPELINED" or "RETIMED"
                SERDES_MODE => "NONE" -- "NONE", "MASTER" or "SLAVE"
            )
port map (   CFB0 => CFB0, -- 1-bit output: Clock feed-through route output
             CFB1 => CFB1, -- 1-bit output: Clock feed-through route output
             DBF => DBF, -- 1-bit output: Feed-through clock output
             FABRICOUT => FABRICOUT, -- 1-bit output: Unsyncronized data output
             INCDEC => INCDEC, -- 1-bit output: Phase detector output
             -- Q1 - Q4: 1-bit (each) output: Registered outputs to FPGA logic
             Q1 => Q1,
             Q2 => Q2,
             Q3 => Q3,
             Q4 => Q4,
             SHIFTOUT => SHIFTOUT, -- 1-bit output: Cascade output signal for master/slave I/O
             VALID => VALID, -- 1-bit output: Output status of the phase detector
             BITSLIP => BITSLIP, -- 1-bit input: Bitslip enable input
            )
```
CE0 -> CE0, -- 1-bit input: Clock enable input
CLK0 -> CLK0, -- 1-bit input: I/O clock network input
CLK1 -> CLK1, -- 1-bit input: Secondary I/O clock network input
CLKDIV -> CLKDIV, -- 1-bit input: FPGA logic domain clock input
D -> D, -- 1-bit input: Input data
IOCE -> IOCE, -- 1-bit input: Data strobe input
RST -> RST, -- 1-bit input: Asynchronous reset input
SHIFTIN -> SHIFTIN -- 1-bit input: Cascade input signal for master/slave I/O

-- End of ISERDES2_inst instantiation

Verilog Instantiation Template

// ISERDES2: Input SERial/DESerializer
// Spartan-6
ISERDES2 #(  
  .BITSLIP_ENABLE("FALSE"), // Enable Bitslip Functionality (TRUE/FALSE)  
  .DATA_RATE("SDR"), // Data-rate ("SDR" or "DDR")  
  .DATA_WIDTH(1), // Parallel data width selection (2-8)  
  .INTERFACE_TYPE("NETWORKING"), // "NETWORKING", "NETWORKING_PIPELINED" or "RETIMED"  
  .SERDES_MODE("NONE") // "NONE", "MASTER" or "SLAVE"
) ISERDES2_inst (  
  .CFB0(CFB0), // 1-bit output: Clock feed-through route output  
  .CFB1(CFB1), // 1-bit output: Clock feed-through route output  
  .DFB(DFB), // 1-bit output: Feed-through clock output  
  .FABRICOUT(FABRICOUT), // 1-bit output: Unsynchronized data output  
  .INCDEC(INCDEC), // 1-bit output: Phase detector output  
  // Q1 - Q4: 1-bit (each) output: Registered outputs to FPGA logic  
  .Q1(Q1),  
  .Q2(Q2),  
  .Q3(Q3),  
  .Q4(Q4),  
  .SHIFTOUT(SHIFTOUT), // 1-bit output: Cascade output signal for master/slave I/O  
  .BITSLIP(BITSLIP), // 1-bit input: Bitslip enable input  
  .CE0(CE0), // 1-bit input: Clock enable input  
  .CLK0(CLK0), // 1-bit input: I/O clock network input  
  .CLK1(CLK1), // 1-bit input: Secondary I/O clock network input  
  .CLKDIV(CLKDIV), // 1-bit input: FPGA logic domain clock input  
  .D(D), // 1-bit input: Input data  
  .IOCE(IOCE), // 1-bit input: Data strobe input  
  .RST(RST), // 1-bit input: Asynchronous reset input  
  .SHIFTIN(SHIFTIN) // 1-bit input: Cascade input signal for master/slave I/O
);

// End of ISERDES2_inst instantiation

For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](https://www.xilinx.com).  
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
JTAG_SIM_SPARTAN6

Simulation: JTAG TAP Controller Simulation Model

Introduction

This simulation component allows the functional simulation of the JTAG TAP controller interface, functions and commands to assist with board-level understanding and debug of the JTAG and Boundary-scan behaviors as well as the behaviors connected to the USER commands and the BSCAN_SPARTAN6 components. This model does not map to a specific primitive in the FPGA software and cannot be directly instantiated in the design however can be used in conjunction with the source design if specified either in a simulation-only file like a testbench or by some means guarded from synthesis so that it is not synthesized into the design netlist. This model may be used for either functional (RTL) simulation or timing simulation.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDO</td>
<td>Output</td>
<td>1</td>
<td>Test Data Out - This pin is the serial output for all JTAG instruction and data registers. The state of the TAP controller and the current instruction determine the register (instruction or data) that feeds TDO for a specific operation. TDO changes state on the falling edge of TCK and is only active during the shifting of instructions or data through the device. TDO is an active driver output.</td>
</tr>
<tr>
<td>TCK</td>
<td>Input</td>
<td>1</td>
<td>Test Clock - This pin is the JTAG Test Clock. TCK sequences the TAP controller and the JTAG registers.</td>
</tr>
<tr>
<td>TDI</td>
<td>Input</td>
<td>1</td>
<td>Test Data - This pin is the serial input to all JTAG instruction and data registers. The state of the TAP controller and the current instruction determine the register that is fed by the TDI pin for a specific operation. TDI has an internal resistive pull-up to provide a logic High to the system if the pin is not driven. TDI is applied into the JTAG registers on the rising edge of TCK.</td>
</tr>
<tr>
<td>TMS</td>
<td>Input</td>
<td>1</td>
<td>Test Mode Select - This pin determines the sequence of states through the TAP controller on the rising edge of TCK. TMS has an internal resistive pull-up to provide a logic High if the pin is not driven.</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>In testbench or simulation-only file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Xilinx suggests that you instantiate this in the testbench file and not an implementation file or file used during synthesis of the design.

More information on simulating and using this component can be found in the Xilinx Synthesis and Simulation Design Guide. Please refer to that guide for further detail on using this component.
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PART_NAME</td>
<td>String</td>
<td>&quot;LX4&quot;, &quot;LX9&quot;, &quot;LX16&quot;, &quot;LX25&quot;, &quot;LX25T&quot;, &quot;LX45&quot;, &quot;LX45T&quot;, &quot;LX75&quot;, &quot;LX75T&quot;, &quot;LX100&quot;, &quot;LX100T&quot;, &quot;LX150&quot;, &quot;LX150T&quot;&quot;</td>
<td>&quot;LX4&quot;</td>
<td>Specify the target device in order to properly set IDCODE and other device specific attributes.</td>
</tr>
</tbody>
</table>

 VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- JTAG_SIM_SPARTAN6: JTAG Interface Simulation Model
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

JTAG_SIM_SPARTAN6_inst : JTAG_SIM_SPARTAN6
generic map (  
    PART_NAME => "LX4") -- Specify target S6 device. Possible values are:
                              -- "LX4", "LX150", "LX150T", "LX16", "LX45", "LX45T"
port map (  
    TDO => TDO,        -- JTAG data output (1-bit)
    TCK => TCK,        -- Clock input (1-bit)
    TDI => TDI,        -- JTAG data input (1-bit)
    TMS => TMS         -- JTAG command input (1-bit)
);

-- End of JTAG_SIM_SPARTAN6_inst instantiation
```

 Verilog Instantiation Template

```verilog
// JTAG_SIM_SPARTAN6: JTAG Interface Simulation Model
// Spartan-6

JTAG_SIM_SPARTAN6 #(  
    .PART_NAME("LX4")) // Specify target S6 device. Possible values are:
                        // "LX4", "LX150", "LX150T", "LX16", "LX45", "LX45T"
) JTAG_SIM_SPARTAN6_inst (  
    .TDO(TDO), // 1-bit JTAG data output
    .TCK(TCK), // 1-bit Clock input
    .TDI(TDI), // 1-bit JTAG data input
    .TMS(TMS)  // 1-bit JTAG command input
);

// End of JTAG_SIM_SPARTAN6_inst instantiation
```

For More Information

- See the *Synthesis and Simulation Design Guide*.
- See the *Spartan-6 FPGA Configuration User Guide*.
**KEEPER**

**Primitive: KEEPERR Symbol**

![KEEPER Symbol]

**Introduction**

The design element is a weak keeper element that retains the value of the net connected to its bidirectional O pin. For example, if a logic 1 is being driven onto the net, KEEPERR drives a weak/resistive 1 onto the net. If the net driver is then 3-stated, KEEPERR continues to drive a weak/resistive 1 onto the net.

**Port Descriptions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1-Bit</td>
<td>Keeper output</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- KEEPER: I/O Buffer Weak Keeper
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

KEEPERR_inst : KEEPER
    port map (     
        O => O     -- Keeper output (connect directly to top-level port) 
    );

-- End of KEEPER_inst instantiation
```

---

**Chapter 4: About Design Elements**

**UG615 (v 13.3) October 26, 2011**
Verilog Instantiation Template

// KEEPER: I/O Buffer Weak Keeper
// Spartan-6

KEEPER KEEPER_inst (.O(O) // Keeper output (connect directly to top-level port)
);

// End of KEEPER_inst instantiation

For More Information

- See the Spartan-6 FPGA SelectIO Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
LDCE

Primitive: Transparent Data Latch with Asynchronous Clear and Gate Enable

Introduction

This design element is a transparent data latch with asynchronous clear and gate enable. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High and CLR is Low. If (GE) is Low, data on (D) cannot be latched. The data on the (D) input during the High-to-Low gate transition is stored in the latch. The data on the (Q) output remains unchanged as long as (G) or (GE) remains low.

This latch is asynchronously cleared, outputs Low, when power is applied. For FPGA devices, power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the appropriate STARTUP_architecture symbol.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>GE</th>
<th>G</th>
<th>D</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>GE</td>
<td>G</td>
<td>D</td>
<td>Q</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>No Change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>No Change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↓</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Binary</td>
<td>0, 1</td>
<td>0</td>
<td>Sets the initial value of Q output after configuration.</td>
</tr>
</tbody>
</table>
**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LDCE: Transparent latch with Asynchronous Reset and
-- Gate Enable.
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LDCE_inst : LDCE
generic map
    (INIT => '0') -- Initial value of latch ('0' or '1')
port map
    (Q => Q, -- Data output
     CLR => CLR, -- Asynchronous clear/reset input
     D => D, -- Data input
     G => G, -- Gate input
     GE => GE -- Gate enable input
);
-- End of LDCE_inst instantiation
```

**Verilog Instantiation Template**

```verilog
// LDCE: Transparent latch with Asynchronous Reset and Gate Enable.
// Spartan-6
LDCE LDCE_inst {
    .Q(Q), // Data output
    .CLR(CLR), // Asynchronous clear/reset input
    .D(D), // Data input
    .G(G), // Gate input
    .GE(GE) // Gate enable input
};
// End of LDCE_inst instantiation
```

**For More Information**

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](http://www.xilinx.com).
Chapter 4: About Design Elements

LDPE

Primitive: Transparent Data Latch with Asynchronous Preset and Gate Enable

Introduction

This design element is a transparent data latch with asynchronous preset and gate enable. When the asynchronous preset (PRE) is High, it overrides the other input and presets the data (Q) output High. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High. The data on the (D) input during the High-to-Low gate transition is stored in the latch. The data on the (Q) output remains unchanged as long as (G) or (GE) remains Low.

The latch is asynchronously preset, output High, when power is applied. For FPGA devices, power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the appropriate STARTUP_architecture symbol.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>GE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Binary</td>
<td>0, 1</td>
<td>1</td>
<td>Specifies the initial value upon power-up or the assertion of GSR for the (Q) port.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- LDPE: Transparent latch with Asynchronous Set and
--     Gate Enable.
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

LDPE_inst : LDPE
generic map (
    INIT => '0') -- Initial value of latch ('0' or '1')
port map (
    Q => Q, -- Data output
    CLR => CLR, -- Asynchronous preset/set input
    D => D, -- Data input
    G => G, -- Gate input
    GE => GE -- Gate enable input
);

-- End of LDPE_inst instantiation

Verilog Instantiation Template

// LDPE: Transparent latch with Asynchronous Preset and Gate Enable.
// Spartan-6

LDPE.LDPE_inst (.Q(Q), // Data output
    .PRE(PRE), // Asynchronous preset/set input
    .D(D), // Data input
    .G(G), // Gate input
    .GE(GE) // Gate enable input
);

// End of LDPE_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
LUT1

Macro: 1-Bit Look-Up Table with General Output

Introduction

This design element is a 1-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Two LUTs are available in each CLB slice; four LUTs are available in each CLB. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>O</td>
</tr>
<tr>
<td>0</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>1</td>
<td>INIT[1]</td>
</tr>
</tbody>
</table>

INIT = Binary number assigned to the INIT attribute

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 2-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT1: 1-input Look-Up Table with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT1_inst : LUT1
generic map (
    INIT => "00"
) port map (
    O => O, -- LUT general output
    I0 => I0 -- LUT input
);
-- End of LUT1_inst instantiation
```

Verilog Instantiation Template

```verilog
// LUT1: 1-input Look-Up Table with general output
// Spartan-6
LUT1 #(
    .INIT(2'b00) // Specify LUT Contents
) LUT1_inst (
    .O(O), // LUT general output
    .I0(I0) // LUT input
);
// End of LUT1_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](http://www.xilinx.com).
Chapter 4: About Design Elements

LUT1_D

Macro: 1-Bit Look-Up Table with Dual Output

Introduction

This design element is a 1-bit look-up table (LUT) with two functionally identical outputs, O and LO. It provides a look-up table version of a buffer or inverter.

The O output is a general interconnect. The LO output is used to connect to another output within the same CLB slice and to the fast connect buffer. A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>O</td>
</tr>
<tr>
<td>0</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>1</td>
<td>INIT[1]</td>
</tr>
</tbody>
</table>

INIT = Binary number assigned to the INIT attribute

Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 2-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT1_D: 1-input Look-Up Table with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT1_D_inst : LUT1_D
generic map (  
  INIT => "00")
port map (  
  L0 => L0, -- LUT local output  
  O => O,  -- LUT general output  
  I0 => I0  -- LUT input
);
-- End of LUT1_D_inst instantiation
```

Verilog Instantiation Template

```verilog
// LUT1_D: 1-input Look-Up Table with general and local outputs
// Spartan-6
LUT1_D #(  
  .INIT(2'b00) // Specify LUT Contents)
LUT1_D_inst (  
  .LO(LO), // LUT local output  
  .O(O),  // LUT general output  
  .I0(I0) // LUT input
);
// End of LUT1_D_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](http://www.xilinx.com).
LUT1_L
Macro: 1-Bit Look-Up Table with Local Output

Introduction
This design element is a 1-bit look-up table (LUT) with a local output (LO) that is used to connect to another output within the same CLB slice and to the fast connect buffer. It provides a look-up table version of a buffer or inverter.

A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method -A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method -Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>LO</td>
</tr>
<tr>
<td>0</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>1</td>
<td>INIT[1]</td>
</tr>
</tbody>
</table>

INIT = Binary number assigned to the INIT attribute

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 2-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT1_L: 1-input Look-Up Table with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT1_L_inst : LUT1_L
generic map (  
  INIT => "00")
port map (  
  LO => LO, -- LUT local output
  I0 => I0 -- LUT input
);
-- End of LUT1_L_inst instantiation
```

Verilog Instantiation Template

```verilog
// LUT1_L: 1-input Look-Up Table with local output
// Spartan-6
LUT1_L #(  
  .INIT(2'b00) // Specify LUT Contents
) LUT1_L_inst (  
  .LO(LO), // LUT local output
  .I0(I0) // LUT input
);
// End of LUT1_L_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](http://www.xilinx.com).
LUT2

Macro: 2-Bit Look-Up Table with General Output

![LUT2 Diagram]

**Introduction**

This design element is a 2-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Two LUTs are available in each CLB slice; four LUTs are available in each CLB. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

**The Logic Table Method** - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

**The Equation Method** - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 4-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT2: 2-input Look-Up Table with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

LUT2_inst : LUT2
generic map (
  INIT => X"0"
) port map (  
  O => O, -- LUT general output
  I0 => I0, -- LUT input
  I1 => I1 -- LUT input
);

-- End of LUT2_inst instantiation
```

Verilog Instantiation Template

```
// LUT2: 2-input Look-Up Table with general output
// Spartan-6

LUT2 #(  
  .INIT(4'h0) // Specify LUT Contents
) LUT2_inst (  
  .O(O), // LUT general output
  .I0(I0), // LUT input
  .I1(I1) // LUT input
);

// End of LUT2_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](https://www.xilinx.com).
Chapter 4: About Design Elements

LUT2_D
Macro: 2-Bit Look-Up Table with Dual Output

Introduction
This design element is a 2-bit look-up table (LUT) with two functionally identical outputs, O and LO. The O output is a general interconnect. The LO output is used to connect to another output within the same CLB slice and to the fast connect buffer. A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The LogicTable Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 4-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT2_D: 2-input Look-Up Table with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT2_D_inst : LUT2_D
generic map (INIT => X"0")
port map (LO => LO, -- LUT local output
0 => 0, -- LUT general output
I0 => I0, -- LUT input
I1 => I1 -- LUT input);
-- End of LUT2_D_inst instantiation
```

Verilog Instantiation Template

```verilog
// LUT2_D: 2-input Look-Up Table with general and local outputs
// Spartan-6
LUT2_D #(.
    .INIT(4'0'h0) // Specify LUT Contents
) LUT2_D_inst (
    .LO(LO), // LUT local output
    .O(0), // LUT general output
    .I0(I0), // LUT input
    .I1(I1) // LUT input
);
// End of LUT2_D_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](http://www.xilinx.com).
Chapter 4: About Design Elements

LUT2_L
Macro: 2-Bit Look-Up Table with Local Output

Introduction

This design element is a 2-bit look-up table (LUT) with a local output (LO) that is used to connect to another output within the same CLB slice and to the fast connect buffer. It provides a look-up table version of a buffer or inverter.

A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 4-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT2_L: 2-input Look-Up Table with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT2_L_inst : LUT2_L
generic map (  
  INIT => X"0"
)  
port map (  
  LO => LO, -- LUT local output  
  I0 => I0, -- LUT input  
  I1 => I1 -- LUT input  
);  
-- End of LUT2_L_inst instantiation

Verilog Instantiation Template

// LUT2_L: 2-input Look-Up Table with local output
// Spartan-6

LUT2_L #(  
  .INIT(4'h0) // Specify LUT Contents
) LUT2_L_inst (  
  .LO(LO), // LUT local output  
  .I0(I0), // LUT input  
  .I1(I1) // LUT input
);  
// End of LUT2_L_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
LUT3

Macro: 3-Bit Look-Up Table with General Output

Introduction

This design element is a 3-bit look-up table (LUT) with general output (O). A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Two LUTs are available in each CLB slice; four LUTs are available in each CLB. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I2</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute
Chapter 4: About Design Elements

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Yes</th>
<th>Recommended</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inference</td>
<td></td>
<td>Recommended</td>
<td></td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 8-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT3: 3-input Look-Up Table with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

LUT3_inst: LUT3
generic map (  
  INIT => X"00"
) port map (  
  O => O,  -- LUT general output  
  I0 => I0,  -- LUT input  
  I1 => I1,  -- LUT input  
  I2 => I2  -- LUT input  
);
-- End of LUT3_inst instantiation
```

Verilog Instantiation Template

// LUT3: 3-input Look-Up Table with general output
// Spartan-6

```verilog
LUT3 #(  
  .INIT(8'h00) // Specify LUT Contents  
) LUT3_inst (  
  .O(O),  // LUT general output  
  .I0(I0), // LUT input  
  .I1(I1), // LUT input  
  .I2(I2)  // LUT input  
);
// End of LUT3_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](https://www.xilinx.com).
LUT3_D
Macro: 3-Bit Look-Up Table with Dual Output

Introduction
This design element is a 3-bit look-up table (LUT) with two functionally identical outputs, O and LO.

The O output is a general interconnect. The LO output is used to connect to another output within the same CLB slice and to the fast connect buffer. A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2</td>
<td>I1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 8-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT3_D: 3-input Look-Up Table with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT3_D_inst : LUT3_D
generic map (INIT => X"00")
port map (LO => LO, -- LUT local output
          O => O,   -- LUT general output
          I0 => I0, -- LUT input
          I1 => I1, -- LUT input
          I2 => I2, -- LUT input);
-- End of LUT3_D_inst instantiation
```

Verilog Instantiation Template

```
// LUT3_D: 3-input Look-Up Table with general and local outputs
// Spartan-6
LUT3_D #(.INIT(8'h00)) // Specify LUT Contents ) LUT3_D_inst (
  .LO(LO), // LUT local output
  .O(O),   // LUT general output
  .I0(I0), // LUT input
  .I1(I1), // LUT input
  .I2(I2)  // LUT input
);
// End of LUT3_D_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](https://www.xilinx.com).
Chapter 4: About Design Elements

LUT3_L

Macro: 3-Bit Look-Up Table with Local Output

Introduction

This design element is a 3-bit look-up table (LUT) with a local output (LO) that is used to connect to another output within the same CLB slice and to the fast connect buffer. It provides a look-up table version of a buffer or inverter.

A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute
Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 8-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT3_L: 3-input Look-Up Table with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT3_L_inst : LUT3_L
generic map (    
  INIT => X"00")
port map (    
  LO  => LO,  -- LUT local output
  I0  => I0,  -- LUT input
  I1  => I1,  -- LUT input
  I2  => I2   -- LUT input
);
-- End of LUT3_L_inst instantiation
```

Verilog Instantiation Template

```
// LUT3_L: 3-input Look-Up Table with local output
// Spartan-6
LUT3_L #(    
  .INIT(8’h00)   // Specify LUT Contents
) LUT3_L_inst (    
  .LO(LO),  // LUT local output
  .I0(I0),   // LUT input
  .I1(I1),   // LUT input
  .I2(I2)    // LUT input
);
// End of LUT3_L_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](http://www.xilinx.com).
LUT4
Macro: 4-Bit Look-Up-Table with General Output

Introduction

This design element is a 4-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Two LUTs are available in each CLB slice; four LUTs are available in each CLB. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>I2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Inputs | Outputs
---|---
I3 | I2 | I1 | I0 | O
1 | 1 | 0 | 0 | INIT[12]
1 | 1 | 0 | 1 | INIT[13]
1 | 1 | 1 | 0 | INIT[14]
1 | 1 | 1 | 1 | INIT[15]

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 16-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT4: 4-input Look-Up Table with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT4_inst : LUT4
generic map (  
  INIT => X"0000")
port map (  
  O => O, -- LUT general output  
  I0 => I0, -- LUT input  
  I1 => I1, -- LUT input  
  I2 => I2, -- LUT input  
  I3 => I3 -- LUT input  
);
-- End of LUT4_inst instantiation
```
Verilog Instantiation Template

// LUT4: 4-input Look-Up Table with general output
// Spartan-6

LUT4 #(  
  .INIT(16'h0000) // Specify LUT Contents  
) LUT4_inst (  
  .O(O), // LUT general output  
  .I0(I0), // LUT input  
  .I1(I1), // LUT input  
  .I2(I2), // LUT input  
  .I3(I3) // LUT input  
);

// End of LUT4_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
LUT4_D

Macro: 4-Bit Look-Up Table with Dual Output

Introduction

This design element is a 4-bit look-up table (LUT) with two functionally identical outputs, O and LO.

The O output is a general interconnect. The LO output is used to connect to another output within the same CLB slice and to the fast connect buffer. A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>I2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Chapter 4: About Design Elements

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>I2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

Design Entry Method

| | 
|-----------------|-----------------|
| Instantiation   | Yes             |
| Inference       | Recommended     |
| CORE Generator™ and wizards | No             |
| Macro support   | No              |

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 16-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT4_D: 4-input Look-Up Table with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT4_D_inst : LUT4_D
generic map ( INIT => X"0000"
port map ( L0 => L0, -- LUT local output
O => O, -- LUT general output
I0 => I0, -- LUT input
I1 => I1, -- LUT input
I2 => I2, -- LUT input
I3 => I3 -- LUT input
);
-- End of LUT4_D_inst instantiation
```
Verilog Instantiation Template

// LUT4_D: 4-input Look-Up Table with general and local outputs
// Spartan-6

LUT4_D #(
    .INIT(16'h0000) // Specify LUT Contents
) LUT4_D_inst (  
    .LO(LO), // LUT local output
    .O(O), // LUT general output
    .I0(I0), // LUT input
    .I1(I1), // LUT input
    .I2(I2), // LUT input
    .I3(I3) // LUT input
);

// End of LUT4_D_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
Chapter 4: About Design Elements

LUT4_L
Macro: 4-Bit Look-Up Table with Local Output

Introduction

This design element is a 4-bit look-up table (LUT) with a local output (LO) that is used to connect to another output within the same CLB slice and to the fast connect buffer. It provides a look-up table version of a buffer or inverter.

A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>I2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Spartan-6 Libraries Guide for HDL Designs

www.xilinx.com

UG615 (v 13.3) October 26, 2011
### Inputs

<table>
<thead>
<tr>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[13]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[15]</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

### Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 16-Bit Value</td>
<td>All zeros</td>
<td>Initializes look-up tables.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT4_L: 4-input Look-Up Table with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT4_L_inst : LUT4_L
  generic map (    
    INIT => X"0000")
  port map (    
    LO => LO, -- LUT local output
    I0 => I0, -- LUT input
    I1 => I1, -- LUT input
    I2 => I2, -- LUT input
    I3 => I3 -- LUT input
  );
-- End of LUT4_L_inst instantiation
```
Verilog Instantiation Template

// LUT4_L: 4-input Look-Up Table with local output
// Spartan-6

LUT4_L #(
  .INIT(16'h0000) // Specify LUT Contents
) LUT4_L_inst (  
  .LO(LO), // LUT local output
  .I0(I0), // LUT input
  .I1(I1), // LUT input
  .I2(I2), // LUT input
  .I3(I3) // LUT input
);
// End of LUT4_L_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
LUT5

Primitive: 5-Input Lookup Table with General Output

Introduction

This design element is a 5-input, 1-output look-up table (LUT) that can either act as an asynchronous 32-bit ROM (with 5-bit addressing) or implement any 5-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. One LUT5 is packed into a LUT6 within a slice, or two LUT5s can be packed into a single LUT6 with some restrictions. The functionality of the LUT5, LUT5_L and LUT5_D is the same. However, the LUT5_L and LUT5_D allow the additional specification to connect the LUT5 output signal to an internal slice or CLB connection using the LO output. The LUT5_L specifies that the only connections from the LUT5 will be within a slice or CLB, while the LUT5_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT5 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 32-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to the corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 32’h80000000 (X”80000000” for VHDL) makes the output zero unless all of the inputs are one (a 5-input AND gate). A Verilog INIT value of 32’hfffffff (X”FFFFFFFF” for VHDL) makes the output one unless all zeros are on the inputs (a 5-input OR gate).

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Chapter 4: About Design Elements
### Inputs

<table>
<thead>
<tr>
<th></th>
<th>I4</th>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>LO</th>
<th>INIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[6]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[7]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[8]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[9]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[10]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[11]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[12]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[13]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[16]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[17]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[18]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[19]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[20]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[21]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[22]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[23]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[24]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[25]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[26]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[27]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[28]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[29]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[30]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[31]</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

### Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>5-LUT output</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>Specifies the logic value for the look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT5: 5-input Look-Up Table with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT5_inst : LUT5
generic map (  
  INIT => X"00000000") -- Specify LUT Contents
port map (  
  O => O, -- LUT general output  
  I0 => I0, -- LUT input  
  I1 => I1, -- LUT input  
  I2 => I2, -- LUT input  
  I3 => I3, -- LUT input  
  I4 => I4 -- LUT input
);

-- End of LUT5_inst instantiation
```

Verilog Instantiation Template

```verilog
// LUT5: 5-input Look-Up Table with general output
// Spartan-6
LUT5 #(  
  .INIT(32'h00000000) // Specify LUT Contents
) LUT5_inst (  
  .O(O), // LUT general output  
  .I0(I0), // LUT input  
  .I1(I1), // LUT input  
  .I2(I2), // LUT input  
  .I3(I3), // LUT input  
  .I4(I4) // LUT input
);

// End of LUT5_inst instantiation
```
For More Information

- See the *Spartan-6 FPGA Configurable Logic Block User Guide*.
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics*. 
LUT5_D

Primitive: 5-Input Lookup Table with General and Local Outputs

Introduction

This design element is a 5-input, 1-output look-up table (LUT) that can either act as an asynchronous 32-bit ROM (with 5-bit addressing) or implement any 5-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. One LUT5 will be packed into a LUT6 within a slice, or two LUT5s can be packed into a single LUT6 with some restrictions. The functionality of the LUT5, LUT5_L and LUT5_D is the same. However, the LUT5_L and LUT5_D allow the additional specification to connect the LUT5 output signal to an internal slice or CLB connection using the LO output. The LUT5_L specifies that the only connections from the LUT5 will be within a slice or CLB, while the LUT5_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT5 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 32-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to the corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 32'h80000000 (X"80000000" for VHDL) will make the output zero unless all of the inputs are one (a 5-input AND gate). A Verilog INIT value of 32'hffffffff (X"FFFFFFFE" for VHDL) will make the output one unless all zeros are on the inputs (a 5-input OR gate).

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method -A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method -Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and more self-documenting that the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I4</td>
<td>O</td>
</tr>
<tr>
<td>I3</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>I2</td>
<td>INIT[1]</td>
</tr>
<tr>
<td>I1</td>
<td>INIT[2]</td>
</tr>
<tr>
<td>I0</td>
<td>INIT[3]</td>
</tr>
<tr>
<td></td>
<td>INIT[4]</td>
</tr>
<tr>
<td></td>
<td>INIT[5]</td>
</tr>
</tbody>
</table>
Inputs

<table>
<thead>
<tr>
<th>I4</th>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>O</th>
<th>LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[6]</td>
<td>INIT[6]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[7]</td>
<td>INIT[7]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[8]</td>
<td>INIT[8]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[9]</td>
<td>INIT[9]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[10]</td>
<td>INIT[10]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[12]</td>
<td>INIT[12]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[13]</td>
<td>INIT[13]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[14]</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[15]</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[16]</td>
<td>INIT[16]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[17]</td>
<td>INIT[17]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[18]</td>
<td>INIT[18]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[19]</td>
<td>INIT[19]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[20]</td>
<td>INIT[20]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[21]</td>
<td>INIT[21]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[22]</td>
<td>INIT[22]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[23]</td>
<td>INIT[23]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[24]</td>
<td>INIT[24]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[25]</td>
<td>INIT[25]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[26]</td>
<td>INIT[26]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[27]</td>
<td>INIT[27]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[28]</td>
<td>INIT[28]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[29]</td>
<td>INIT[29]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[30]</td>
<td>INIT[30]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[31]</td>
<td>INIT[31]</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>5-LUT output</td>
</tr>
<tr>
<td>L0</td>
<td>Output</td>
<td>1</td>
<td>5-LUT output for internal CLB connection</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>Specifies the logic value for the look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT5_D: 5-input Look-Up Table with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

LUT5_D_inst : LUT5_D
generic map (  
  INIT => X"00000000") -- Specify LUT contents
port map (  
  LO => LO, -- LUT local output
  O => O, -- LUT general output
  I0 => I0, -- LUT input
  I1 => I1, -- LUT input
  I2 => I2, -- LUT input
  I3 => I3, -- LUT input
  I4 => I4 -- LUT input
);
-- End of LUT5_D_inst instantiation
```

Verilog Instantiation Template

// LUT5_D: 5-input Look-Up Table with general and local outputs
// Spartan-6

```verilog
LUT5_D #(  
  .INIT(32'h00000000) // Specify LUT Contents
) LUT5_D_inst (  
  .LO(LO), // LUT local output
  .O(O),   // LUT general output
  .I0(I0), // LUT input
  .I1(I1), // LUT input
  .I2(I2), // LUT input
  .I3(I3), // LUT input
  .I4(I4)  // LUT input
);
// End of LUT5_D_inst instantiation
```
For More Information

- See the *Spartan-6 FPGA Configurable Logic Block User Guide*.
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics*. 
**LUT5_L**

**Primitve: 5-Input Lookup Table with Local Output**

```
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I4</td>
<td>LO</td>
</tr>
<tr>
<td>I3</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>I2</td>
<td>INIT[1]</td>
</tr>
<tr>
<td>I1</td>
<td>INIT[2]</td>
</tr>
<tr>
<td>I0</td>
<td>INIT[3]</td>
</tr>
</tbody>
</table>
```

**Introduction**

This design element is a 5-input, 1-output look-up table (LUT) that can either act as an asynchronous 32-bit ROM (with 5-bit addressing) or implement any 5-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. One LUT5 will be packed into a LUT6 within a slice, or two LUT5s can be packed into a single LUT6 with some restrictions. The functionality of the LUT5, LUT5_L and LUT5_D is the same. However, the LUT5_L and LUT5_D allow the additional specification to connect the LUT5 output signal to an internal slice or CLB connection using the LO output. The LUT5_L specifies that the only connections from the LUT5 is within a slice or CLB, while the LUT5_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT5 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 32-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to the corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 32'h80000000 (X"80000000" for VHDL) makes the output zero unless all of the inputs are one (a 5-input AND gate). A Verilog INIT value of 32'hfffffe (X"FFFFFFFF" for VHDL) makes the output one unless all zeros are on the inputs (a 5-input OR gate).

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

**The Logic Table Method** - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary truth table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

**The Equation Method** - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed logic value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.
### Chapter 4: About Design Elements

#### Inputs

<table>
<thead>
<tr>
<th>I4</th>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>LO</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[7]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[8]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[9]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[10]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[11]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[12]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[13]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[16]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[17]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[18]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[19]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[20]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[21]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[22]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[23]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[24]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[25]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[26]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[27]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[28]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[29]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[30]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[31]</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

### Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>Output</td>
<td>1</td>
<td>6/5-LUT output for internal CLB connection</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
</tbody>
</table>
**Design Entry Method**

<table>
<thead>
<tr>
<th>Design Entry Method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>Specifies the logic value for the look-up tables.</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT5_L: 5-input Look-Up Table with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT5_L_inst : LUT5_L
generic map (  
  INIT => X"00000000") -- Specify LUT Contents
port map (  
  LO => LO, -- LUT local output  
  I0 => I0, -- LUT input  
  I1 => I1, -- LUT input  
  I2 => I2, -- LUT input  
  I3 => I3, -- LUT input  
  I4 => I4 -- LUT input  
);
-- End of LUT5_L_inst instantiation
```

**Verilog Instantiation Template**

```
// LUT5_L: 5-input Look-Up Table with local output
// Spartan-6
LUT5_L #(  
  .INIT(32'h00000000) // Specify LUT Contents
) LUT5_L_inst (  
  .LO(LO), // LUT local output  
  .I0(I0), // LUT input  
  .I1(I1), // LUT input  
  .I2(I2), // LUT input  
  .I3(I3), // LUT input  
  .I4(I4) // LUT input  
);
// End of LUT5_L_inst instantiation
```
For More Information

- See the *Spartan-6 FPGA Configurable Logic Block User Guide*.
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics*. 
LUT6

Primitive: 6-Input Lookup Table with General Output

![6-Input Look-Up Table](image)

Introduction

This design element is a 6-input, 1-output look-up table (LUT) that can either act as an asynchronous 64-bit ROM (with 6-bit addressing) or implement any 6-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6 is mapped to one of the four look-up tables in the slice. The functionality of the LUT6, LUT6_L and LUT6_D is the same. However, the LUT6_L and LUT6_D allow the additional specification to connect the LUT6 output signal to an internal slice, or CLB connection, using the LO output. The LUT6_L specifies that the only connections from the LUT6 will be within a slice, or CLB, while the LUT6_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT6 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 64-bit Hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64'h8000000000000000 (X"8000000000000000" for VHDL) makes the output zero unless all of the inputs are one (a 6-input AND gate). A Verilog INIT value of 64'hfffffffffffffffe (X"FFFFFFFFFFFFFFFF" for VHDL) makes the output one unless all zeros are on the inputs (a 6-input OR gate).

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

```
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I5</td>
<td>O</td>
</tr>
<tr>
<td>I4</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>I3</td>
<td>INIT[1]</td>
</tr>
<tr>
<td>I2</td>
<td>INIT[2]</td>
</tr>
<tr>
<td>I1</td>
<td></td>
</tr>
<tr>
<td>I0</td>
<td></td>
</tr>
</tbody>
</table>
```

Spartan-6 Libraries Guide for HDL Designs
UG615 (v 13.3) October 26, 2011  www.xilinx.com  187
## Inputs

<table>
<thead>
<tr>
<th>I5</th>
<th>I4</th>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[3]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[4]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[5]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[6]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[7]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[8]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[9]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[10]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[12]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[13]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[16]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[17]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[18]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[19]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[20]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[21]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[22]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[23]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[24]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[25]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[26]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[28]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[29]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[30]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[31]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[32]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[33]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[34]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[35]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[37]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[38]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[39]</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>I5</th>
<th>I4</th>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[3]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[4]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[5]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[6]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[7]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[8]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[9]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[10]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[12]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[13]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[16]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[17]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[18]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[19]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[20]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[21]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[22]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[23]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[24]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[25]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[26]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[28]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[29]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[30]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[31]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[32]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[33]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[34]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[35]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[37]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[38]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[39]</td>
</tr>
</tbody>
</table>
### Inputs

<table>
<thead>
<tr>
<th>I5</th>
<th>I4</th>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[40]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[41]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[42]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[43]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[44]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[45]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[46]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[47]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[48]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[49]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[50]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[51]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[52]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[53]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[54]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[55]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[56]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[57]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[58]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[59]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[60]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[61]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[62]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[63]</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute

### Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>6/5-LUT output</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4, I5</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
</tbody>
</table>

### Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the logic value for the look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT6: 6-input Look-Up Table with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

LUT6_inst : LUT6
generic map (    
  INIT => X"0000000000000000") -- Specify LUT Contents
port map (      
  O => O, -- LUT general output
  I0 => I0, -- LUT input
  I1 => I1, -- LUT input
  I2 => I2, -- LUT input
  I3 => I3, -- LUT input
  I4 => I4, -- LUT input
  I5 => I5 -- LUT input
  );
-- End of LUT6_inst instantiation
```

Verilog Instantiation Template

```
// LUT6: 6-input Look-Up Table with general output
// Spartan-6

LUT6 #(
  .INIT(64'h0000000000000000) // Specify LUT Contents
) LUT6_inst (  
  .O(O), // LUT general output
  .I0(I0), // LUT input
  .I1(I1), // LUT input
  .I2(I2), // LUT input
  .I3(I3), // LUT input
  .I4(I4), // LUT input
  .I5(I5) // LUT input
  );
// End of LUT6_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
**LUT6_2**

**Primitive: Six-input, 2-output, Look-Up Table**

![Image of LUT6_2](image)

**Introduction**

This design element is a 6-input, 2-output look-up table (LUT) that can either act as a dual asynchronous 32-bit ROM (with 5-bit addressing), implement any two 5-input logic functions with shared inputs, or implement a 6-input logic function and a 5-input logic function with shared inputs and shared logic values. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6_2 will be mapped to one of the four look-up tables in the slice.

An INIT attribute consisting of a 64-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64'hfffffffffffffffe (X'FFFFFFFFFFFFFFFFFFF' for VHDL) makes the O6 output 1 unless all zeros are on the inputs and the O5 output a 1, or unless I[4:0] are all zeroes (a 5-input and 6-input OR gate). The lower half (bits 31:0) of the INIT values apply to the logic function of the O5 output.

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

**The Logic Table Method** - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

**The Equation Method** - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>I5</td>
<td>I4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Chapter 4: About Design Elements

### Inputs

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>INIT[5]</th>
<th>INIT[5]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[6]</td>
<td>INIT[6]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[7]</td>
<td>INIT[7]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[8]</td>
<td>INIT[8]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[9]</td>
<td>INIT[9]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[10]</td>
<td>INIT[10]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[12]</td>
<td>INIT[12]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[13]</td>
<td>INIT[13]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[14]</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[15]</td>
<td>INIT[15]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[16]</td>
<td>INIT[16]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[17]</td>
<td>INIT[17]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[18]</td>
<td>INIT[18]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[19]</td>
<td>INIT[19]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[20]</td>
<td>INIT[20]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[21]</td>
<td>INIT[21]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[22]</td>
<td>INIT[22]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[23]</td>
<td>INIT[23]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[24]</td>
<td>INIT[24]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[25]</td>
<td>INIT[25]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[26]</td>
<td>INIT[26]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[27]</td>
<td>INIT[27]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[28]</td>
<td>INIT[28]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[29]</td>
<td>INIT[29]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[30]</td>
<td>INIT[30]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[31]</td>
<td>INIT[31]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[0]</td>
<td>INIT[32]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[1]</td>
<td>INIT[33]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[2]</td>
<td>INIT[34]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[3]</td>
<td>INIT[35]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT[4]</td>
<td>INIT[36]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT[5]</td>
<td>INIT[37]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT[6]</td>
<td>INIT[38]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT[7]</td>
<td>INIT[39]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[8]</td>
<td>INIT[40]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[9]</td>
<td>INIT[41]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[10]</td>
<td>INIT[42]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT[11]</td>
<td>INIT[43]</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT[8]</td>
<td>INIT[8]</td>
</tr>
<tr>
<td>INIT[12]</td>
<td>INIT[12]</td>
</tr>
<tr>
<td>INIT[14]</td>
<td>INIT[14]</td>
</tr>
<tr>
<td>INIT[16]</td>
<td>INIT[16]</td>
</tr>
<tr>
<td>INIT[17]</td>
<td>INIT[17]</td>
</tr>
<tr>
<td>INIT[18]</td>
<td>INIT[18]</td>
</tr>
<tr>
<td>INIT[19]</td>
<td>INIT[19]</td>
</tr>
<tr>
<td>INIT[21]</td>
<td>INIT[21]</td>
</tr>
<tr>
<td>INIT[22]</td>
<td>INIT[22]</td>
</tr>
<tr>
<td>INIT[23]</td>
<td>INIT[23]</td>
</tr>
<tr>
<td>INIT[26]</td>
<td>INIT[26]</td>
</tr>
<tr>
<td>INIT[27]</td>
<td>INIT[27]</td>
</tr>
<tr>
<td>INIT[28]</td>
<td>INIT[28]</td>
</tr>
<tr>
<td>INIT[29]</td>
<td>INIT[29]</td>
</tr>
<tr>
<td>INIT[31]</td>
<td>INIT[31]</td>
</tr>
<tr>
<td>INIT[0]</td>
<td>INIT[32]</td>
</tr>
<tr>
<td>INIT[1]</td>
<td>INIT[33]</td>
</tr>
<tr>
<td>INIT[2]</td>
<td>INIT[34]</td>
</tr>
<tr>
<td>INIT[5]</td>
<td>INIT[37]</td>
</tr>
<tr>
<td>INIT[6]</td>
<td>INIT[38]</td>
</tr>
<tr>
<td>INIT[8]</td>
<td>INIT[40]</td>
</tr>
<tr>
<td>INIT[9]</td>
<td>INIT[41]</td>
</tr>
<tr>
<td>INIT[10]</td>
<td>INIT[42]</td>
</tr>
</tbody>
</table>

---

Spartan-6 Libraries Guide for HDL Designs

192  
www.xilinx.com  
UG615 (v 13.3) October 26, 2011
Chapter 4: About Design Elements

Inputs

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O6</td>
<td>Output</td>
<td>1</td>
<td>6/5-LUT output</td>
</tr>
<tr>
<td>O5</td>
<td>Output</td>
<td>1</td>
<td>5-LUT output</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4, I5</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the LUT5/6 output function.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT6_2: 6-input 2 output Look-Up Table
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT6_2_inst : LUT6_2
generic map
  (INIT => X"0000000000000000") -- Specify LUT Contents
port map
  (O6 => O6, -- 6/5-LUT output (1-bit)
   O5 => O5, -- 5-LUT output (1-bit)
   I0 => I0, -- LUT input (1-bit)
   I1 => I1, -- LUT input (1-bit)
   I2 => I2, -- LUT input (1-bit)
   I3 => I3, -- LUT input (1-bit)
   I4 => I4, -- LUT input (1-bit)
   I5 => I5 -- LUT input (1-bit)
);
-- End of LUT6_2_inst instantiation
```

Verilog Instantiation Template

```
// LUT6_2: 6-input, 2 output Look-Up Table
// Spartan-6
LUT6_2 #(
   .INIT(64'h0000000000000000) // Specify LUT Contents
) LUT6_2_inst (
   .O6(O6), // 1-bit LUT6 output
   .O5(O5), // 1-bit lower LUT5 output
   .I0(I0), // 1-bit LUT input
   .I1(I1), // 1-bit LUT input
   .I2(I2), // 1-bit LUT input
   .I3(I3), // 1-bit LUT input
   .I4(I4), // 1-bit LUT input
   .I5(I5) // 1-bit LUT input (fast MUX select only available to O6 output)
);
// End of LUT6_2_inst instantiation
```

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
LUT6_D

Primitive: 6-Input Lookup Table with General and Local Outputs

Introduction

This design element is a six-input, one-output look-up table (LUT) that can either act as an asynchronous 64-bit ROM (with 6-bit addressing) or implement any 6-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6 is mapped to one of the four look-up tables in the slice. The functionality of the LUT6, LUT6_L and LUT6_D is the same. However, the LUT6_L and LUT6_D allow the additional specification to connect the LUT6 output signal to an internal slice, or CLB connection, using the LO output. The LUT6_L specifies that the only connections from the LUT6 will be within a slice, or CLB, while the LUT6_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT6 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 64-bit Hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64’h8000000000000000 (X"8000000000000000" for VHDL) makes the output zero unless all of the inputs are one (a 6-input AND gate). A Verilog INIT value of 64’hfffffffffffffe (X"FFFFFFFFFFFFFFFFFE" for VHDL) makes the output one unless all zeros are on the inputs (a 6-input OR gate).

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method - A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method - Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and more is self-documenting that the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I5</td>
<td>O</td>
</tr>
<tr>
<td>I4</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>I3</td>
<td>INIT[1]</td>
</tr>
<tr>
<td>I2</td>
<td>INIT[2]</td>
</tr>
<tr>
<td>I1</td>
<td>LO</td>
</tr>
<tr>
<td>I0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I5</th>
<th>I4</th>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>O</th>
<th>LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT[0]</td>
<td>INIT[0]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT[1]</td>
<td>INIT[1]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT[2]</td>
<td>INIT[2]</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I5</td>
<td>INIT[3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I4</td>
<td>INIT[3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I3</td>
<td>INIT[4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2</td>
<td>INIT[4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I1</td>
<td>INIT[5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I0</td>
<td>INIT[5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>INIT[6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LO</td>
<td>INIT[6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chapter 4: About Design Elements

Sparan-6 Libraries Guide for HDL Designs

UG615 (v 13.3) October 26, 2011
### Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O6</td>
<td>Output</td>
<td>1</td>
<td>6/5-LUT output</td>
</tr>
<tr>
<td>O5</td>
<td>Output</td>
<td>1</td>
<td>5-LUT output</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4, I5</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
</tbody>
</table>

### Design Entry Method

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the logic value for the look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT6_D: 6-input Look-Up Table with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT6_D_inst : LUT6_D
generic map (
  INIT => X"0000000000000000") -- Specify LUT contents
port map (  
  LO => LO, -- LUT local output
  O => O,   -- LUT general output
  I0 => I0, -- LUT input
  I1 => I1, -- LUT input
  I2 => I2, -- LUT input
  I3 => I3, -- LUT input
  I4 => I4, -- LUT input
  I5 => I5 -- LUT input
);
-- End of LUT6_D_inst instantiation
```

Verilog Instantiation Template

```
// LUT6_D: 6-input Look-Up Table with general and local outputs
// Spartan-6

LUT6_D #(  
  .INIT(64'h0000000000000000) // Specify LUT Contents
) LUT6_D_inst (  
  .LO(LO), // LUT local output
  .O(O),   // LUT general output
  .I0(I0), // LUT input
  .I1(I1), // LUT input
  .I2(I2), // LUT input
  .I3(I3), // LUT input
  .I4(I4), // LUT input
  .I5(I5)  // LUT input
);

// End of LUT6_D_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](http://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](http://www.xilinx.com).
LUT6_L

Primitive: 6-Input Lookup Table with Local Output

Introduction

This design element is a 6-input, 1-output look-up table (LUT) that can either act as an asynchronous 64-bit ROM (with 6-bit addressing) or implement any 6-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6 is mapped to one of the four look-up tables in the slice. The functionality of the LUT6, LUT6_L and LUT6_D is the same. However, the LUT6_L and LUT6_D allow the additional specification to connect the LUT6 output signal to an internal slice, or CLB connection, using the LO output. The LUT6_L specifies that the only connections from the LUT6 are within a slice, or CLB, while the LUT6_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT6 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 64-bit hexadecimal value must be specified to indicate the LUT's logical function. The INIT value is calculated by assigning a 1 to the corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64'h8000000000000000 (X"8000000000000000" for VHDL) will make the output zero unless all of the inputs are one (a 6-input AND gate). A Verilog INIT value of 64'hfffffffffffffe (X"FFFFFFFfffffff" for VHDL) will make the output one unless all zeros are on the inputs (a 6-input OR gate).

The INIT parameter for the FPGA LUT primitive is what gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined:

The Logic Table Method

-A common method to determine the desired INIT value for a LUT is using a logic table. To do so, simply create a binary truth table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

The Equation Method

-Another method to determine the LUT value is to define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting that the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I5</td>
<td>I4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Chapter 4: About Design Elements

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_5 )</td>
<td>( I_4 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### Port Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO</td>
<td>Output</td>
<td>1</td>
<td>6/5-LUT output or internal CLB connection</td>
</tr>
<tr>
<td>I0, I1, I2, I3, I4, I5</td>
<td>Input</td>
<td>1</td>
<td>LUT inputs</td>
</tr>
</tbody>
</table>

### Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the logic value for the look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT6_L: 6-input Look-Up Table with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
LUT6_LInst : LUT6_L
generic map (  
  INIT => X"0000000000000000") -- Specify LUT Contents
port map (  
  LO => LO, -- LUT local output  
  I0 => I0, -- LUT input  
  I1 => I1, -- LUT input  
  I2 => I2, -- LUT input  
  I3 => I3, -- LUT input  
  I4 => I4, -- LUT input  
  I5 => I5 -- LUT input  
);
-- End of LUT6_LInst instantiation
```

Verilog Instantiation Template

```verilog
// LUT6_L: 6-input Look-Up Table with local output
// Spartan-6
LUT6_L #(
  .INIT(64'h0000000000000000) // Specify LUT Contents
) LUT6_LInst (  
  .LO(LO), // LUT local output  
  .I0(I0), // LUT input  
  .I1(I1), // LUT input  
  .I2(I2), // LUT input  
  .I3(I3), // LUT input  
  .I4(I4), // LUT input  
  .I5(I5) // LUT input  
);
// End of LUT6_LInst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
MCB

Primitive: Memory Control Block

Introduction

This element is a hardened memory controller supporting several popular memory interfaces. The MCB is only supported through the use of the Memory Interface Generator (MIG) tool. Details on the use and capability of the MCB can be found in the Spartan-6 FPGA Memory Controller User Guide, UG 388.

For More Information

- See the Spartan-6 FPGA Memory Controller User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
- See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
### MUXF7

**Primitive:** 2-to-1 Look-Up Table Multiplexer with General Output

#### Introduction

This design element provides a multiplexer function for use in creating a function-of-7 look-up table or an 8-to-1 multiplexer in combination with the associated look-up tables. Local outputs (LO) of MUXF6 are connected to the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The variants, “MUXF7_D” and “MUXF7_L”, provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

#### Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to general routing</td>
</tr>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF6 LO out)</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF6 LO out)</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Input select to MUX</td>
</tr>
</tbody>
</table>

#### Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- MUXF7: CLB MUX to tie two MUXF6's together with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
MUXF7_inst : MUXF7
port map (
  O => O, -- Output of MUX to general routing
  I0 => I0, -- Input (tie to MUXF6 LO out or LUT6 O6 pin)
  I1 => I1, -- Input (tie to MUXF6 LO out or LUT6 O6 pin)
  S => S -- Input select to MUX
);

-- End of MUXF7_inst instantiation

Verilog Instantiation Template

// MUXF7: CLB MUX to tie two LUT6's or MUXF6's together with general output
// Spartan-6
MUXF7 MUXF7_inst {
  .O(O), // Output of MUX to general routing
  .I0(I0), // Input (tie to LUT6 O6 pin)
  .I1(I1), // Input (tie to LUT6 O6 pin)
  .S(S) // Input select to MUX
};

// End of MUXF7_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
**MUXF7_D**

Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output

**Introduction**

This design element provides a multiplexer function for use in creating a function-of-7 look-up table or a 16-to-1 multiplexer in combination with the associated look-up tables. Local outputs (LO) of MUXF6 are connected to the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO output connects to other inputs in the same CLB slice.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to general routing</td>
</tr>
<tr>
<td>LO</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to local routing</td>
</tr>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF6 LO out)</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF6 LO out)</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Input select to MUX</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- MUXF7_D: CLB MUX to tie two MUXF6's together with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
MUXF7_D_inst : MUXF7_D
port map ( 
  LO => LO, -- Output of MUX to local routing
  O => O, -- Output of MUX to general routing
  I0 => I0, -- Input (tie to MUXF6 LO out or LUT6 O6 pin)
  I1 => I1, -- Input (tie to MUXF6 LO out or LUT6 O6 pin)
  S => S -- Input select to MUX
);

-- End of MUXF7_D_inst instantiation

Verilog Instantiation Template

// MUXF7_D: CLB MUX to tie two LUT6's or MUXF6's together with general and local outputs
// Spartan-6
MUXF7_D_MUXF7_D_inst ( 
  .LO(LO), // Output of MUX to local routing
  .O(O), // Output of MUX to general routing
  .I0(I0), // Input (tie to LUT6 O6 pin)
  .I1(I1), // Input (tie to LUT6 O6 pin)
  .S(S) // Input select to MUX
);

// End of MUXF7_D_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
**MUXF7_L**

**Primitive: 2-to-1 look-up table Multiplexer with Local Output**

**Introduction**

This design element provides a multiplexer function for use in creating a function-of-7 look-up table or a 16-to-1 multiplexer in combination with the associated look-up tables. Local outputs (LO) of MUXF6 are connected to the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The LO output connects to other inputs in the same CLB slice.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to local routing</td>
</tr>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Input</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Input</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Input select to MUX</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- MUXF7_L: CLB MUX to tie two MUXF6's together with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
MUXF7_L_inst : MUXF7_L
port map (LO => LO, -- Output of MUX to local routing
         I0 => I0, -- Input (tie to MUXF6 LO out or LUT6 O6 pin)
         I1 => I1, -- Input (tie to MUXF6 LO out or LUT6 O6 pin)
         S => S,  -- Input select to MUX
        );
-- End of MUXF7_L_inst instantiation
```

Verilog Instantiation Template

```verilog
// MUXF7_L: CLB MUX to tie two LUT6's or MUXF6's together with local output
// Spartan-6
MUXF7_L MUXF7_L_inst {
         .LO(LO), // Output of MUX to local routing
         .I0(I0), // Input (tie to LUT6 O6 pin)
         .I1(I1), // Input (tie to LUT6 O6 pin)
         .S(S),  // Input select to MUX
        );
// End of MUXF7_L_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
Chapter 4: About Design Elements

MUXF8

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output

Introduction

This design element provides a multiplexer function in eight slices for creating a function-of-8 look-up table or a 16-to-1 multiplexer in combination with the associated look-up tables, MUXF5s, MUXF6s, and MUXF7s. Local outputs (LO) of MUXF7 are connected to the I0 and I1 inputs of the MUXF8. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to general routing</td>
</tr>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF7 LO out)</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF7 LO out)</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Input select to MUX</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- MUXF8: CLB MUX to tie two MUXF7's together with general output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
MUXF8_inst : MUXF8
port map (  
  O => O,  -- Output of MUX to general routing
  I0 => I0,  -- Input (tie to MUXF7 LO out)
  I1 => I1,  -- Input (tie to MUXF7 LO out)
  S => S    -- Input select to MUX
);

-- End of MUXF8_inst instantiation

Verilog Instantiation Template

// MUXF8: CLB MUX to tie two MUXF7's together with general output
// Spartan-6
MUXF8 MUXF8_inst (  
  .O(O),  // Output of MUX to general routing
  .I0(I0),  // Input (tie to MUXF7 LO out)
  .I1(I1),  // Input (tie to MUXF7 LO out)
  .S(S)  // Input select to MUX
);

// End of MUXF8_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

MUXF8_D

Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output

Introduction

This design element provides a multiplexer function in eight slices for creating a function-of-8 look-up table or a 32-to-1 multiplexer in combination with the associated four look-up tables and two MUXF8s. Local outputs (LO) of MUXF7 are connected to the I0 and I1 inputs of the MUXF8. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO output connects to other inputs in the same CLB slice.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to general routing</td>
</tr>
<tr>
<td>LO</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to local routing</td>
</tr>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF7 LO out)</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF7 LO out)</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Input select to MUX</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Chapter 4: About Design Elements

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- MUXF8_D: CLB MUX to tie two MUXF7's together with general and local outputs
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
MUXF8_D_inst : MUXF8_D
port map (
    LO => LO, -- Output of MUX to local routing
    O => O, -- Output of MUX to general routing
    I0 => I0, -- Input (tie to MUXF7 LO out)
    I1 => I1, -- Input (tie to MUXF7 LO out)
    S => S -- Input select to MUX
);
-- End of MUXF8_D_inst instantiation
```

Verilog Instantiation Template

```verilog
// MUXF8_D: CLB MUX to tie two MUXF7's together with general and local outputs
// Spartan-6
MUXF8_D_MUXF8_D_inst (
    .LO(LO), // Output of MUX to local routing
    .O(O), // Output of MUX to general routing
    .I0(I0), // Input (tie to MUXF7 LO out)
    .I1(I1), // Input (tie to MUXF7 LO out)
    .S(S) // Input select to MUX
);
// End of MUXF8_D_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
MUXF8_L

Primitive: 2-to-1 Look-Up Table Multiplexer with Local Output

Introduction

This design element provides a multiplexer function in eight slices for creating a function-of-8 look-up table or a 32-to-1 multiplexer in combination with the associated four look-up tables and two MUXF8s. Local outputs (LO) of MUXF7 are connected to the I0 and I1 inputs of the MUXF8. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The LO output connects to other inputs in the same CLB slice.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>I0</th>
<th>I1</th>
<th>LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
<td>X</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>I1</td>
<td>I1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO</td>
<td>Output</td>
<td>1</td>
<td>Output of MUX to local routing</td>
</tr>
<tr>
<td>I0</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF7 LO out)</td>
</tr>
<tr>
<td>I1</td>
<td>Input</td>
<td>1</td>
<td>Input (tie to MUXF7 LO out)</td>
</tr>
<tr>
<td>S</td>
<td>Input</td>
<td>1</td>
<td>Input select to MUX</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- MUXF8_L: CLB MUX to tie two MUXF7's together with local output
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
MUXF8_L_inst : MUXF8_L
port map (
  LO => LO, -- Output of MUX to local routing
  I0 => I0, -- Input (tie to MUXF7 LO out)
  I1 => I1, -- Input (tie to MUXF7 LO out)
  S => S   -- Input select to MUX
);
-- End of MUXF8_L_inst instantiation

Verilog Instantiation Template

// MUXF8_L: CLB MUX to tie two MUXF7's together with local output
// Spartan-6
MUXF8_L MUXF8_L_inst {
  .LO(LO), // Output of MUX to local routing
  .I0(I0), // Input (tie to MUXF7 LO out)
  .I1(I1), // Input (tie to MUXF7 LO out)
  .S(S)   // Input select to MUX
};
// End of MUXF8_L_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

**OBUF**

**Primitive: Output Buffer**

```
OBUF
  I   O
```

### Introduction

This design element is a simple output buffer used to drive output signals to the FPGA device pins that do not need to be 3-stated (constantly driven). Either an OBUF, OBUFT, OBUFDS, or OBUFTDS must be connected to every output port in the design.

This element isolates the internal circuit and provides drive current for signals leaving a chip. It exists in input/output blocks (IOB). Its output (O) is connected to an OPAD or an IOPAD. The interface standard used by this element is LVTTTL. Also, this element has selectable drive and slew rates using the DRIVE and SLOW or FAST constraints. The defaults are DRIVE=12 mA and SLOW slew.

### Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Output of OBUF to be connected directly to top-level output port.</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Input of OBUF. Connect to the logic driving the output port.</td>
</tr>
</tbody>
</table>

### Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- OBUF: Single-ended Output Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

OBUF_inst : OBUF
generic map (    
    DRIVE => 12,  
    IOSTANDARD => "DEFAULT",  
    SLEW => "SLOW")
port map (    
    O => O,  
    I => I  
);

-- End of OBUF_inst instantiation
```

Verilog Instantiation Template

```verilog
// OBUF: Single-ended Output Buffer
// Spartan-6

OBUF #(    
    .DRIVE(12), // Specify the output drive strength
    .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
    .SLEW("SLOW") // Specify the output slew rate
) OBUF_inst (    
    .O(O), // Buffer output (connect directly to top-level port)
    .I(I) // Buffer input
);

// End of OBUF_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
Chapter 4: About Design Elements

**OBUFDS**

**Primitve: Differential Signaling Output Buffer**

![OBUFDS Symbol]

**Introduction**

This design element is a single output buffer that supports low-voltage, differential signaling (1.8 v CMOS). OBUFDS isolates the internal circuit and provides drive current for signals leaving the chip. Its output is represented as two distinct ports (O and OB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET and MYNETB).

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>O</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Port Descriptions**

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Diff_p output (connect directly to top level port)</td>
</tr>
<tr>
<td>OB</td>
<td>Output</td>
<td>1</td>
<td>Diff_n output (connect directly to top level port)</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Buffer input</td>
</tr>
</tbody>
</table>

**Design Entry Method**

<table>
<thead>
<tr>
<th>Method</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Recommended</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- OBUFDS: Differential Output Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

OBUFDS_inst : OBUFDS
generic map (IOSTANDARD => "DEFAULT")
port map (O => O, -- Diff_p output (connect directly to top-level port)
          OB => OB, -- Diff_n output (connect directly to top-level port)
          I => I    -- Buffer input);

-- End of OBUFDS_inst instantiation

Verilog Instantiation Template

// OBUFDS: Differential Output Buffer
// Spartan-6

OBUFDS #(IOSTANDARD("DEFAULT")) // Specify the output I/O standard
  O(C),  // Diff_p output (connect directly to top-level port)
  OB(CB), // Diff_n output (connect directly to top-level port)
  I(C)   // Buffer input;

// End of OBUFDS_inst instantiation

For More Information

- See the Spartan-6 FPGA SelectIO Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
OBUFT

Primitive: 3-State Output Buffer with Active Low Output Enable

Introduction

This design element is a single, 3-state output buffer with input I, output O, and active-Low output enables (T). This element uses the LVTTL standard and has selectable drive and slew rates using the DRIVE and SLOW or FAST constraints. The defaults are DRIVE=12 mA and SLOW slew.

When T is Low, data on the inputs of the buffers is transferred to the corresponding outputs. When T is High, the output is high impedance (off or Z state). OBUFTs are generally used when a single-ended output is needed with a 3-state capability, such as the case when building bidirectional I/O.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>I</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Buffer output (connect directly to top-level port)</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Buffer input</td>
</tr>
<tr>
<td>T</td>
<td>Input</td>
<td>1</td>
<td>3-state enable input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- OBUFT: Single-ended 3-state Output Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
OBUFT_inst : OBUFT
generic map (  
  DRIVE => 12,
  IOSTANDARD => "DEFAULT",
  SLEW => "SLOW")
port map (  
  O => O, -- Buffer output (connect directly to top-level port)
  I => I, -- Buffer input
  T => T -- 3-state enable input
);
-- End of OBUFT_inst instantiation

Verilog Instantiation Template

// OBUFT: Single-ended 3-state Output Buffer
// Spartan-6

OBUFT #(  
  .DRIVE(12), // Specify the output drive strength  
  .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
  .SLEW("SLOW") // Specify the output slew rate  
) OBUFT_inst (  
  .O(O), // Buffer output (connect directly to top-level port)
  .I(I), // Buffer input
  .T(T) // 3-state enable input
);
// End of OBUFT_inst instantiation

For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](http://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](http://www.xilinx.com).
Chapter 4: About Design Elements

OBUFTDS

Primitive: 3-State Output Buffer with Differential Signaling, Active-Low Output Enable

Introduction

This design element is an output buffer that supports low-voltage, differential signaling. For the OBUFTDS, a design level interface signal is represented as two distinct ports (O and OB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N).

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>T</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Diff_p output (connect directly to top level port)</td>
</tr>
<tr>
<td>OB</td>
<td>Output</td>
<td>1</td>
<td>Diff_n output (connect directly to top level port)</td>
</tr>
<tr>
<td>I</td>
<td>Input</td>
<td>1</td>
<td>Buffer input</td>
</tr>
<tr>
<td>T</td>
<td>Input</td>
<td>1</td>
<td>3-state enable input</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>String</td>
<td>See Data Sheet</td>
<td>&quot;DEFAULT&quot;</td>
<td>Assigns an I/O standard to the element.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- OBUFTDS: Differential 3-state Output Buffer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

OBUFTDS_inst : OBUFTDS
generic map (    IOSTANDARD => "DEFAULT")
port map (    O => O, -- Diff_p output (connect directly to top-level port)
              OB => OB, -- Diff_n output (connect directly to top-level port)
              I => I, -- Buffer input
              T => T -- 3-state enable input
    );

-- End of OBUFTDS_inst instantiation

Verilog Instantiation Template

// OBUFTDS: Differential 3-state Output Buffer
// Spartan-6

OBUFTDS #(    .IOSTANDARD("DEFAULT") // Specify the output I/O standard ) OBUFTDS_inst {
    .O(O), // Diff_p output (connect directly to top-level port)
    .OB(OB), // Diff_n output (connect directly to top-level port)
    .I(I), // Buffer input
    .T(T) // 3-state enable input
};

// End of OBUFTDS_inst instantiation

For More Information

• See the Spartan-6 FPGA SelectIO Resources User Guide.
• See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

ODDR2

Primitive: Dual Data Rate Output D Flip-Flop with Optional Data Alignment, Clock Enable and Programmable Synchronous or Asynchronous Set/Reset

Introduction

The design element is an output double data rate (DDR) register useful in producing double data rate signals exiting the FPGA. The ODDR2 requires two clocks (C0 and C1) to be connected to the component so that data is provided at the positive edge of both clocks. The ODDR2 features an active high clock enable port, CE, which can be used to suspend the operation of the registers and both set and reset ports that can be configured to be synchronous or asynchronous to the respective clocks. The ODDR2 has an optional alignment feature, which allows data to be captured by a single clock and clocked out by two clocks.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Set/Reset can be synchronous via SRTYPE value

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR_ALIGNMENT</td>
<td>String</td>
<td>&quot;NONE&quot;, &quot;C0&quot;,</td>
<td>&quot;NONE&quot;</td>
<td>Sets the input capture behavior for the DDR register. &quot;NONE&quot; clocks in data to the D0 input on the positive transition of the C0 clock and D1 on the positive transition of the C1 clock. &quot;C0&quot; allows the input clocking of both D0 and D1 align to the positive edge of the C0 clock. &quot;C1&quot; allows the input clocking of both D0 and D1 align to the positive edge of the C1 clock.</td>
</tr>
<tr>
<td>INIT</td>
<td>Binary</td>
<td>0, 1</td>
<td>0</td>
<td>Sets initial state of the Q0 output to 0 or 1.</td>
</tr>
<tr>
<td>SRTYPE</td>
<td>String</td>
<td>&quot;SYNC&quot;, &quot;ASYNC&quot;</td>
<td>&quot;SYNC&quot;</td>
<td>Specifies &quot;SYNC&quot; or &quot;ASYNC&quot; set/reset.</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- ODDR2: Output Double Data Rate Output Register with Set, Reset
-- and Clock Enable.
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
ODDR2_inst : ODDR2
  generic map (DDR_ALIGNMENT => "NONE", -- Sets output alignment to "NONE", "C0", "C1"
               INIT => '0', -- Sets initial state of the Q output to '0' or '1'
               SRTYPE => "SYNC") -- Specifies "SYNC" or "ASYNC" set/reset
  port map (Q => Q, -- 1-bit output data
            C0 => C0, -- 1-bit clock input
            C1 => C1, -- 1-bit clock input
            CE => CE, -- 1-bit clock enable input
            D0 => D0, -- 1-bit data input (associated with C0)
            D1 => D1, -- 1-bit data input (associated with C1)
            R => R, -- 1-bit reset input
            S => S -- 1-bit set input
  );
-- End of ODDR2_inst instantiation
```

**Verilog Instantiation Template**

```verilog
// ODDR2: Output Double Data Rate Output Register with Set, Reset
// and Clock Enable.
// Spartan-6

ODDR2 #(DDR_ALIGNMENT("NONE"), // Sets output alignment to "NONE", "C0" or "C1"
       INIT(1'b0), // Sets initial state of the Q output to 1'b0 or 1'b1
       SRTYPE("SYNC") // Specifies "SYNC" or "ASYNC" set/reset
) ODDR2_inst (Q(Q), // 1-bit DDR output data
             .C0(C0), // 1-bit clock input
             .C1(C1), // 1-bit clock input
             .CE(CE), // 1-bit clock enable input
             .D0(D0), // 1-bit data input (associated with C0)
             .D1(D1), // 1-bit data input (associated with C1)
```
Chapter 4: About Design Elements

.R(R),  // 1-bit reset input
.S(S)  // 1-bit set input
;
// End of ODDR2_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
OSERDES2

Primitive: Dedicated IOB Output Serializer

Introduction

Each IOB contains a output serializer block that can be instantiated in a design by using the OSERDES2 primitive. OSERDES2 allows parallel-to-serial conversion with SerDes ratios of 2:1, 3:1, and 4:1. The SerDes ratio is the ratio between the high-speed I/O clock that is transmitting data, and the slower internal global clock used for processing the parallel data. For example, with an I/O clock running at 500 MHz to transmit data at 500 Mb/s, the OSERDES transfers four bits of data at one quarter of this rate (125 MHz) from the FPGA logic. When using differential outputs, the two OSERDES2 primitives associated with the two IOBs can be cascaded to allow higher SerDes ratios of 5:1, 6:1, 7:1 and 8:1.
## Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKDIV</td>
<td>Input</td>
<td>1</td>
<td>Global clock network input. This is the clock for the fabric domain.</td>
</tr>
<tr>
<td>CLK0</td>
<td>Input</td>
<td>1</td>
<td>I/O clock network input. Optionally invertible. This is the primary clock input used when the clock doubler circuit is not engaged. See DATA_RATE attribute for more information.</td>
</tr>
<tr>
<td>CLK1</td>
<td>Input</td>
<td>1</td>
<td>IO Clock network input. Optionally Invertible. This secondary clock input is only used when the clock doubler is engaged.</td>
</tr>
<tr>
<td>D1 - D4</td>
<td>Input</td>
<td>1</td>
<td>Parallel data inputs.</td>
</tr>
<tr>
<td>IOCE</td>
<td>Input</td>
<td>1</td>
<td>Data strobe signal derived from BUFIO2 CE. Strobes data capture to be correctly timed with respect to the I/O and global clocks for the SerDes mode selected.</td>
</tr>
<tr>
<td>OCE</td>
<td>Input</td>
<td>1</td>
<td>Clock enable for data inputs.</td>
</tr>
<tr>
<td>OQ</td>
<td>Output</td>
<td>1</td>
<td>Data path output to pad or IODELAY2.</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Shared data, 3-state reset pin. Asynchronous only.</td>
</tr>
<tr>
<td>SHIFTIN1</td>
<td>Input</td>
<td>1</td>
<td>Cascade data input signal (dummy in Master). Used for DATA_WIDTHs greater than four.</td>
</tr>
<tr>
<td>SHIFTIN2</td>
<td>Input</td>
<td>1</td>
<td>Cascade 3-state input signal (dummy in master). Used for DATA_WIDTHs greater than 4.</td>
</tr>
<tr>
<td>SHIFTIN3</td>
<td>Input</td>
<td>1</td>
<td>Differential data input Signals (dummy in Slave).</td>
</tr>
<tr>
<td>SHIFTIN4</td>
<td>Input</td>
<td>1</td>
<td>Differential 3-state input signal (dummy in Slave)</td>
</tr>
<tr>
<td>SHIFTOUT1</td>
<td>Output</td>
<td>1</td>
<td>Cascade data output signal (dummy in Slave). Used for DATA_WIDTHs greater than four.</td>
</tr>
<tr>
<td>SHIFTOUT2</td>
<td>Output</td>
<td>1</td>
<td>Cascade 3-state output signal (dummy in Slave). Used for DATA_WIDTHs greater than 4.</td>
</tr>
<tr>
<td>SHIFTOUT3</td>
<td>Output</td>
<td>1</td>
<td>Differential data output signals (dummy in Master).</td>
</tr>
<tr>
<td>SHIFTOUT4</td>
<td>Output</td>
<td>1</td>
<td>Differential 3-state output signal (dummy in Master)</td>
</tr>
<tr>
<td>TCE</td>
<td>Input</td>
<td>1</td>
<td>Clock enable for 3-state inputs.</td>
</tr>
<tr>
<td>TQ</td>
<td>Output</td>
<td>1</td>
<td>3-state path output to pad or IODELAY2.</td>
</tr>
<tr>
<td>TRAIN</td>
<td>Input</td>
<td>1</td>
<td>Enable use of the training pattern. The train function is a means of specifying a fixed output pattern that can be used to calibrate the receiver of the signal. This port allows the FPGA logic to control whether the output is that fixed pattern or the input data from the pins.</td>
</tr>
<tr>
<td>T1 - T4</td>
<td>Input</td>
<td>1</td>
<td>3-state control inputs.</td>
</tr>
</tbody>
</table>

## Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>Recommended</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYPASS_GCLK_FF</td>
<td>Boolean</td>
<td>FALSE, TRUE</td>
<td>FALSE</td>
<td>Bypass CLDIV synchronization registers.</td>
</tr>
<tr>
<td>DATA_RATE_OQ</td>
<td>String</td>
<td>“DDR”, “SDR”</td>
<td>“DDR”</td>
<td>Data rate setting. The DDR clock can be supplied by separate I/O clocks or a single I/O clock. If two clocks are supplied, they must be approximately 180 out of phase.</td>
</tr>
<tr>
<td>DATA_RATE_OT</td>
<td>String</td>
<td>“DDR”, “BUF”,</td>
<td>“DDR”</td>
<td>3-state data rate setting. The DDR clock can be supplied by separate I/O clocks or by a single I/O clock. If two clocks are supplied, they must be approximately 180 out of phase.</td>
</tr>
<tr>
<td>DATA_WIDTH</td>
<td>Integer</td>
<td>2, 1, 3, 4, 5, 6, 7, 8</td>
<td>2</td>
<td>Data width. Determines the parallel data input width of the parallel-to-serial converter. Values greater than four are only valid when two OSERDES2 blocks are cascaded. In this case, the same value should be applied to both the master and slave blocks.</td>
</tr>
<tr>
<td>OUTPUT_MODE</td>
<td>String</td>
<td>“SINGLE_ENDED”,</td>
<td>“SINGLE_ENDED”</td>
<td>Output Mode.</td>
</tr>
<tr>
<td>SERDES_MODE</td>
<td>String</td>
<td>“MASTER”, “SLAVE”</td>
<td>“MASTER”</td>
<td>Indicates whether OSERDES is being used alone, or as a Master or Slave when two OSERDES2 blocks are cascaded.</td>
</tr>
<tr>
<td>TRAIN_PATTERN</td>
<td>Integer</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15</td>
<td>0</td>
<td>Defines training pattern to be sent when TRAIN port is active.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- OSERDES2: Output SERial/DESerializer
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

OSERDES2_inst : OSERDES2
generic map (  
    BYPASS_GCLK_FF => FALSE,  -- Bypass CLDIV synchronization registers (TRUE/FALSE)  
    DATA_RATE_OQ  => "DDR",  -- Output Data Rate ("SDR" or "DDR")  
    DATA_RATE_OT  => "DDR",  -- 3-state Data Rate ("SDR" or "DDR")  
    DATA_WIDTH    => 2,      -- Parallel data width (2-8)  
    OUTPUT_MODE   => "SINGLE_ENDED",  -- "SINGLE_ENDED" or "DIFFERENTIAL"  
    SERDES_MODE   => "NONE",   -- "NONE", "MASTER" or "SLAVE"  
    TRAIN_PATTERN => 0        -- Training Pattern (0-15)  
)
port map (  
    OQ  => OQ,  -- 1-bit output: Data output to pad or IODELAY2  
    SHIFTOUT1 => SHIFTOUT1,  -- 1-bit output: Cascade data output  
    SHIFTOUT2 => SHIFTOUT2,  -- 1-bit output: Cascade 3-state output  
    SHIFTOUT3 => SHIFTOUT3,  -- 1-bit output: Cascade differential data output  
    SHIFTOUT4 => SHIFTOUT4,  -- 1-bit output: Cascade differential 3-state output  
    TQ  => TQ,  -- 1-bit output: 3-state output to pad or IODELAY2  
    CLK0 => CLK0,  -- 1-bit input: I/O clock input  
    CLK1 => CLK1,  -- 1-bit input: Secondary I/O clock input  
    CLKDIV => CLKDIV,  -- 1-bit input: Logic domain clock input  
    -- D1 - D4: 1-bit (each) input: Parallel data inputs  
    D1  => D1,  
    D2  => D2,  
    D3  => D3,  
```

Chapter 4: About Design Elements

Spartan-6 Libraries Guide for HDL Designs

UG615 (v 13.3) October 26, 2011  www.xilinx.com  229
D4 -> D4, IOCE -> IOCE, OCE -> OCE, -- 1-bit input: Data strobe input RST -> RST, -- 1-bit input: Clock enable input SHIFTIN1 -> SHIFTIN1, -- 1-bit input: Cascade data input SHIFTIN2 -> SHIFTIN2, -- 1-bit input: Cascade 3-state input SHIFTIN3 -> SHIFTIN3, -- 1-bit input: Cascade differential data input SHIFTIN4 -> SHIFTIN4, -- 1-bit input: Cascade differential 3-state input -- T1 - T4: 1-bit (each) input: 3-state control inputs T1 -> T1, T2 -> T2, T3 -> T3, T4 -> T4, TCE -> TCE, -- 1-bit input: 3-state clock enable input TRAIN -> TRAIN -- 1-bit input: Training pattern enable input

-- End of OSERDES2_inst instantiation

Verilog Instantiation Template

```verilog
// OSERDES2: Output SERial/DESerializer
// Spartan-6

OSERDES2 #!
  .BYPASS_CLKDIV("FALSE"), // Bypass CLKDIV synchronization registers (TRUE/FALSE)
  .DATA_RATE_QQ("DDR"), // Output Data Rate ("SDR" or "DDR")
  .DATA_RATE_OT("DDR"), // 3-state Data Rate ("SDR" or "DDR")
  .DATA_WIDTH(2), // Parallel data width (2-8)
  .OUTPUT_MODE("SINGLE_ENDED"), // "SINGLE_ENDED" or "DIFFERENTIAL"
  .SERDES_MODE("NONE"), // "NONE", "MASTER" or "SLAVE"
  .TRAIN_PATTERN(0) // Training Pattern (0-15)
)

OSERDES2_inst
  .QO(QO), // 1-bit output: Data output to pad or IODELAY2
  .SHIFTOUT1(SHIFTOUT1), // 1-bit output: Cascade data output
  .SHIFTOUT2(SHIFTOUT2), // 1-bit output: Cascade 3-state output
  .SHIFTOUT3(SHIFTOUT3), // 1-bit output: Cascade differential data output
  .SHIFTOUT4(SHIFTOUT4), // 1-bit output: Cascade differential 3-state output
  .TQ(TQ), // 1-bit output: 3-state output to pad or IODELAY2
  .CLK0(CLK0), // 1-bit input: I/O clock input
  .CLK1(CLK1), // 1-bit input: Secondary I/O clock input
  .CLKDIV(CLKDIV), // 1-bit input: Logic domain clock input
// D1 - D4: 1-bit (each) input: Parallel data inputs
  .D1(D1),
  .D2(D2),
  .D3(D3),
  .D4(D4),
  .IOCE(IOCE), // 1-bit input: Data strobe input
  .OCE(OCE), // 1-bit input: Clock enable input
  .RST(RST), // 1-bit input: Asynchronous reset input
  .SHIFTIN1(SHIFTIN1), // 1-bit input: Cascade data input
  .SHIFTIN2(SHIFTIN2), // 1-bit input: Cascade 3-state input
  .SHIFTIN3(SHIFTIN3), // 1-bit input: Cascade differential data input
  .SHIFTIN4(SHIFTIN4), // 1-bit input: Cascade differential 3-state input
// T1 - T4: 1-bit (each) input: 3-state control inputs
  .T1(T1),
  .T2(T2),
  .T3(T3),
  .T4(T4),
  .TCE(TCE), // 1-bit input: 3-state clock enable input
  .TRAIN(TRAIN) // 1-bit input: Training pattern enable input
);

// End of OSERDES2_inst instantiation
For More Information

- See the *Spartan-6 FPGA SelectIO Resources User Guide.*
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.*
PCIE_A1

Primitive: PCI Express
Introduction

This design element is intended for use in conjunction with other resources located in the FPGA, such as the RocketIO™ transceiver, block RAMs, and various clocking resources. To implement an PCI EXPRESS® design using PCIE_A1, designers must use the CORE Generator™ software tool (part of the ISE® Design Suite) to create a LogiCORE™ IP core for PCI EXPRESS designs. The LogiCORE IP instantiates the PCIE_A1 software primitive, connects the interfaces to the correct FPGA resources, sets all attributes, and presents a simple, user-friendly interface.

Design Entry Method

To instantiate this component, use the PCI EXPRESS® core or an associated core containing the component. Xilinx does not recommend direct instantiation of this component.

For More Information

- See the [Spartan-6 FPGA RocketIO GTP Transceivers User Guide](https://www.xilinx.com).
- See the [LogiCORE™ IP Spartan-6 FPGA Integrated Endpoint Block v1.1 for PCI EXPRESS® User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
PLL_BASE

Primitive: Basic Phase Locked Loop Clock Circuit

### Introduction

This design element is a direct sub-set of the PLL_ADV design element, an embedded Phase Locked Loop clock circuit that provides added capabilities for clock synthesis and management both within the FPGA and in circuits external to the FPGA. The PLL_BASE is provided in order to ease the integration for most PLL clocking circuits. However, this primitive does not contain all of the functionality that the PLL can possibly provide. This component allows the input clock to be phase shifted, multiplied and divided, and supports other features, such as modification of the duty cycle and jitter filtering.

### Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKOUT0-5</td>
<td>Output</td>
<td>1</td>
<td>One of six phase controlled output clocks from the PLL.</td>
</tr>
<tr>
<td>CLKFBOUT</td>
<td>Output</td>
<td>1</td>
<td>Dedicated PLL feedback output used to determine how the PLL compensates clock network delay. Depending on the type of compensation desired, this output might or might not need to be connected.</td>
</tr>
<tr>
<td>CLKIN</td>
<td>Input</td>
<td>1</td>
<td>Clock source input to the PLL. This pin can be driven by a dedicated clock pin to the FPGA, a DCM output clock pin, or a BUFG output.</td>
</tr>
<tr>
<td>CLKFBIN</td>
<td>Input</td>
<td>1</td>
<td>Clock feedback input. This pin should only be sourced from the CLKFBOUT port.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>Output</td>
<td>1</td>
<td>Asynchronous output from the PLL that provides you with an indication the PLL has achieved phase alignment and is ready for operation.</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>1</td>
<td>Asynchronous reset of the PLL.</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPENSATION</td>
<td>String</td>
<td>&quot;SYSTEM_SYNCHRONOUS&quot;, &quot;SOURCE_SYNCHRONOUS&quot;</td>
<td>&quot;SYSTEM_SYNCHRONOUS&quot;</td>
<td>Specifies the PLL phase compensation for the incoming clock. SYSTEM_SYNCHRONOUS attempts to compensate all clock delay while SOURCE_SYNCHRONOUS is used when a clock is provided with data and thus phased with the clock.</td>
</tr>
<tr>
<td>BANDWIDTH</td>
<td>String</td>
<td>&quot;HIGH&quot;, &quot;LOW&quot;, &quot;OPTIMIZED&quot;</td>
<td>&quot;OPTIMIZED&quot;</td>
<td>Specifies the PLL programming algorithm affecting the jitter, phase margin and other characteristics of the PLL.</td>
</tr>
<tr>
<td>CLKOUT0_DIVIDE,</td>
<td>Integer</td>
<td>1 to 128</td>
<td>1</td>
<td>Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the FBCLKOUT_MULT value determines the output frequency.</td>
</tr>
<tr>
<td>CLKOUT1_DIVIDE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT2_DIVIDE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT3_DIVIDE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT4_DIVIDE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT5_DIVIDE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT0_PHASE,</td>
<td>Real</td>
<td>0.01 to 360.0</td>
<td>0.0</td>
<td>Allows specification of the output phase relationship of the associated CLKOUT clock output in number of degrees offset (i.e. 90 indicates a 90 degree or ¼ cycle offset phase offset while 180 indicates a 180 degree offset or ½ cycle phase offset).</td>
</tr>
<tr>
<td>CLKOUT1_PHASE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT2_PHASE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT3_PHASE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT4_PHASE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT5_PHASE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT0_DUTY_CYCLE,</td>
<td>Real</td>
<td>0.01 to 0.99</td>
<td>0.50</td>
<td>Specifies the Duty Cycle of the associated CLKOUT clock output in percentage (i.e. 0.50 generates a 50% duty cycle).</td>
</tr>
<tr>
<td>CLKOUT1_DUTY_CYCLE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT2_DUTY_CYCLE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT3_DUTY_CYCLE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT4_DUTY_CYCLE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT5_DUTY_CYCLE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKFBOUT_MULTI</td>
<td>Integer</td>
<td>1 to 64</td>
<td>1</td>
<td>Specifies the amount to multiply all CLKOUT clock outputs if a different frequency is desired. This number in combination with the associated CLKOUT2_DIVIDE value determines the output frequency.</td>
</tr>
<tr>
<td>DIVCLK_DIVIDE</td>
<td>Integer</td>
<td>1 to 52</td>
<td>1</td>
<td>Specifies the division ratio for all output clocks.</td>
</tr>
<tr>
<td>CLKFBOUT_PHASE</td>
<td>Real</td>
<td>0.0 to 360.0</td>
<td>0.0</td>
<td>Specifies the phase offset in degrees of the clock feedback output.</td>
</tr>
</tbody>
</table>
### Attribute Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK_FEEDBACK</td>
<td>String</td>
<td>&quot;CLKFBOUT&quot;, &quot;CLKOUT0&quot;</td>
<td>&quot;CLKFBOUT&quot;</td>
<td>Specifies the clock source to drive CLKFB_IN. Refer to UG382: Spartan®-6 FPGA Clocking Resources User Guide for correct usage of feedback resources and calculating VCO frequency.</td>
</tr>
<tr>
<td>REF_JITTER</td>
<td>Real</td>
<td>0.000 to 0.999</td>
<td>0.100</td>
<td>The reference clock jitter is specified in terms of the UI which is a percentage of the reference clock. The number provided should be the maximum peak to peak value on the input clock.</td>
</tr>
<tr>
<td>CLKIN_PERIOD</td>
<td>Real</td>
<td>1.000 to 52.630</td>
<td>None</td>
<td>Specified the input period in ns to the PLL CLKIN input.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- PLL_BASE: Phase Locked Loop (PLL) Clock Management Component
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

PLL_BASE_inst : PLL_BASE
  generic map (  
    BANDWIDTH => "OPTIMIZED",  
    -- "HIGH", "LOW" or "OPTIMIZED"  
    CLKFBOUT_MULT => 1,  
    -- Multiply value for all CLKOUT clock outputs (1-64)  
    CLKFBOUT_PHASE => 0.0,  
    -- Phase offset in degrees of the clock feedback output  
    -- (0.0-360.0).  
    CLKIN_PERIOD => 0.0,  
    -- Input clock period in ns to ps resolution (i.e. 33.333 is 30 ns)  
    -- MHz).  
    -- CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for CLKOUT# clock output (1-128)  
    CLKOUT0_DIVIDE => 1,  
    CLKOUT1_DIVIDE => 1,  
    CLKOUT2_DIVIDE => 1,  
    CLKOUT3_DIVIDE => 1,  
    CLKOUT4_DIVIDE => 1,  
    CLKOUT5_DIVIDE => 1,  
    -- CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for CLKOUT# clock output (0.01-0.99).  
    CLKOUT0_DUTY_CYCLE => 0.5,  
    CLKOUT1_DUTY_CYCLE => 0.5,  
    CLKOUT2_DUTY_CYCLE => 0.5,  
    CLKOUT3_DUTY_CYCLE => 0.5,  
    CLKOUT4_DUTY_CYCLE => 0.5,  
    CLKOUT5_DUTY_CYCLE => 0.5,  
    -- CLKOUT0_PHASE - CLKOUT5_PHASE: Output phase relationship for CLKOUT# clock output (-360.0-360.0).  
    CLKOUT0_PHASE => 0.0,  
    CLKOUT1_PHASE => 0.0,  
    CLKOUT2_PHASE => 0.0,  
    CLKOUT3_PHASE => 0.0,  
    CLKOUT4_PHASE => 0.0,  
    CLKOUT5_PHASE => 0.0,  
    CLK_FEEDBACK => "CLKFBOUT",  
    -- Clock source to drive CLKFBIN ("CLKFBOUT" or "CLKOUT0")  
    COMPENSATION => "SYSTEM_SYNCHRONOUS",  
    -- "SYSTEM_SYNCHRONOUS", "SOURCE_SYNCHRONOUS", "EXTERNAL"  
    DIVCLK_DIVIDE => 1,  
    -- Division value for all output clocks (1-52)  
    REF_JITTER => 0.1,  
    -- Reference Clock Jitter in UI (0.000-0.999).  
    RESET_ON_LOSS_OF_LOCK => FALSE  
    -- Must be set to FALSE  
  )  
  port map (  
    CLKFBOUT => CLKFBOUT,  
    -- 1-bit output: PLL_BASE feedback output  
    -- CLKOUT0 - CLKOUT5: 1-bit (each) output: Clock outputs  
    CLKOUT0 => CLKOUT0,  
    CLKOUT1 => CLKOUT1,  
    ...  
  );
```
CLKOUT2 => CLKOUT2,
CLKOUT3 => CLKOUT3,
CLKOUT4 => CLKOUT4,
CLKOUT5 => CLKOUT5,
LOCKED => LOCKED, -- 1-bit output: PLL_BASE lock status output
CLKFBIN => CLKFBIN, -- 1-bit input: Feedback clock input
CLKIN => CLKIN, -- 1-bit input: Clock input
RST => RST -- 1-bit input: Reset input
);

-- End of PLL_BASE_inst instantiation

Verilog Instantiation Template

// PLL_BASE: Phase Locked Loop (PLL) Clock Management Component
// Spartan-6
PLL_BASE #(
    .BANDWIDTH("OPTIMIZED"), // "HIGH", "LOW" or "OPTIMIZED"
    .CLKFBOUT_MULT(1), // Multiply value for all CLKOUT clock outputs (1-64)
    .CLKFBOUT_PHASE(0.0), // Phase offset in degrees of the clock feedback output (0.0-360.0).
    .CLKIN_PERIOD(0.0), // Input clock period in ns to ps resolution (i.e. 33.333 is 30 // MHz).
    .CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for CLKOUT# clock output (1-128)
    .CLKOUT0_DIVIDE(1),
    .CLKOUT1_DIVIDE(1),
    .CLKOUT2_DIVIDE(1),
    .CLKOUT3_DIVIDE(1),
    .CLKOUT4_DIVIDE(1),
    .CLKOUT5_DIVIDE(1),
    .CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for CLKOUT# clock output (0.01-0.99).
    .CLKOUT0_DUTY_CYCLE(0.5),
    .CLKOUT1_DUTY_CYCLE(0.5),
    .CLKOUT2_DUTY_CYCLE(0.5),
    .CLKOUT3_DUTY_CYCLE(0.5),
    .CLKOUT4_DUTY_CYCLE(0.5),
    .CLKOUT5_DUTY_CYCLE(0.5),
    .CLKOUT0_PHASE - CLKOUT5_PHASE: Output phase relationship for CLKOUT# clock output (-360.0-360.0).
    .CLKOUT0_PHASE(0.0),
    .CLKOUT1_PHASE(0.0),
    .CLKOUT2_PHASE(0.0),
    .CLKOUT3_PHASE(0.0),
    .CLKOUT4_PHASE(0.0),
    .CLKOUT5_PHASE(0.0),
    .CLK_FEEDBACK("CLKFBOUT"), // Clock source to drive CLKFBIN ("CLKFBOUT" or "CLKOUT0")
    .COMPENSATION("SYSTEM_SYNCHRONOUS"), // "SYSTEM_SYNCHRONOUS", "SOURCE_SYNCHRONOUS", "EXTERNAL"
    .DIVCLK_DIVIDE(1), // Division value for all output clocks (1-52)
    .REF_JITTER(0.1), // Reference Clock Jitter in UI (0.000-0.999).
    .RESET_ON_LOSS_OF_LOCK("FALSE") // Must be set to FALSE
)
PLL_BASE_inst ( PLL_BASE_inst PORT MAP
    .CLKFBOUT(CLKFBOUT), // 1-bit output: PLL_BASE feedback output
    .CLKOUT0(CLKOUT0),
    .CLKOUT1(CLKOUT1),
    .CLKOUT2(CLKOUT2),
    .CLKOUT3(CLKOUT3),
    .CLKOUT4(CLKOUT4),
    .CLKOUT5(CLKOUT5),
    .LOCKED(LOCKED), -- 1-bit output: PLL_BASE lock status output
    .CLKFBIN(CLKFBIN), // 1-bit input: Feedback clock input
    .CLKIN(CLKIN), // 1-bit input: Clock input
    .RST(RST) // 1-bit input: Reset input
);

// End of PLL_BASE_inst instantiation
For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
POST_CRC_INTERNAL

Primitive: Post-configuration CRC error detection

Introduction

This primitive provides fabric access to post CRC error. This new primitive is added to provide more flexibility of POST_CRC usage. It is also the only access to POST CRC status when CRC_EXTSTAT_DISABLE is activated.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRCERROR</td>
<td>Output</td>
<td>1</td>
<td>Post-configuration CRC error</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- POST_CRC_INTERNAL: Post-configuration CRC error detection
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

POST_CRC_INTERNAL_inst : POST_CRC_INTERNAL
port map (CRCERROR => CRCERROR -- 1-bit output: Post-configuration CRC error output);

-- End of POST_CRC_INTERNAL_inst instantiation
```

Verilog Instantiation Template

```
// POST_CRC_INTERNAL: Post-configuration CRC error detection
// Spartan-6

POST_CRC_INTERNAL POST_CRC_INTERNAL_inst (.
  .CRCERROR(CRCERROR) // 1-bit output: Post-configuration CRC error output
);

// End of POST_CRC_INTERNAL_inst instantiation
```
For More Information

- See the Spartan-6 FPGA Configuration User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

PULLDOWN

Primitive: Resistor to GND for Input Pads, Open-Drain, and 3-State Outputs

### Introduction

This resistor element is connected to input, output, or bidirectional pads to guarantee a logic Low level for nodes that might float.

### Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Pulldown output (connect directly to top level port)</td>
</tr>
</tbody>
</table>

### Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- PULLDOWN: I/O Buffer Weak Pull-down
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

PULLDOWN_inst : PULLDOWN
port map ( O => O -- Pulldown output (connect directly to top-level port) );

-- End of PULLDOWN_inst instantiation
```

Spartan-6 Libraries Guide for HDL Designs
UG615 (v 13.3) October 26, 2011

www.xilinx.com 241
Verilog Instantiation Template

// PULLDOWN: I/O Buffer Weak Pull-down
// Spartan-6

PULLDOWN PULLDOWN_inst (  
    .O(O)  // Pulldown output (connect directly to top-level port)
);  

// End of PULLDOWN_inst instantiation

For More Information

- See the Spartan-6 FPGA SelectIO Resources User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
Chapter 4: About Design Elements

PULLUP

Primitive: Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs

Introduction

This design element allows for an input, 3-state output or bi-directional port to be driven to a weak high value when not being driven by an internal or external source. This element establishes a High logic level for open-drain elements and macros when all the drivers are off.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Pullup output (connect directly to top level port)</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- PULLUP: I/O Buffer Weak Pull-up
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
PULLUP_inst : PULLUP
port map (  
  O => O    -- Pullup output (connect directly to top-level port)
);

-- End of PULLUP_inst instantiation
```
Verilog Instantiation Template

```verilog
// PULLUP: I/O Buffer Weak Pull-up
// Spartan-6

PULLUP PULLUP_inst ( // Pullup output (connect directly to top-level port)
 .O(O) )

// End of PULLUP_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA SelectIO Resources User Guide](http://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](http://www.xilinx.com).
Chapter 4: About Design Elements

RAM128X1D

Primitive: 128-Deep by 1-Wide Dual Port Random Access Memory (Select RAM)

Introduction

This design element is a 128-bit deep by 1-bit wide random access memory and has a read/write port that writes the value on the D input data pin when the write enable (WE) is high to the location specified by the A address bus. This happens shortly after the rising edge of the WCLK and that same value is reflected in the data output SPO. When WE is low, an asynchronous read is initiated in which the contents of the memory location specified by the A address bus is output asynchronously to the SPO output. The read port can perform asynchronous read access of the memory by changing the value of the address bus DPRA, and by outputing that value to the DPO data output.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPO</td>
<td>Output</td>
<td>1</td>
<td>Read/Write port data output addressed by A</td>
</tr>
<tr>
<td>DPO</td>
<td>Output</td>
<td>1</td>
<td>Read port data output addressed by DPRA</td>
</tr>
<tr>
<td>D</td>
<td>Input</td>
<td>1</td>
<td>Write data input addressed by A</td>
</tr>
<tr>
<td>A</td>
<td>Input</td>
<td>7</td>
<td>Read/Write port address bus</td>
</tr>
<tr>
<td>DPRA</td>
<td>Input</td>
<td>7</td>
<td>Read port address bus</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>1</td>
<td>Write Enable</td>
</tr>
<tr>
<td>WCLK</td>
<td>Input</td>
<td>1</td>
<td>Write clock (reads are asynchronous)</td>
</tr>
</tbody>
</table>

If instantiated, the following connections should be made to this component:

- Tie the WCLK input to the desired clock source, the D input to the data source to be stored and the DPO output to an FDCE D input or other appropriate data destination.
- Optionally, the SPO output can also be connected to the appropriate data destination or else left unconnected.
- The WE clock enable pin should be connected to the proper write enable source in the design.
- The 7-bit A bus should be connected to the source for the read/write addressing and the 7-bit DPRA bus should be connected to the appropriate read address connections.
- An optional INIT attribute consisting of a 128-bit Hexadecimal value can be specified to indicate the initial contents of the RAM.

If left unspecified, the initial contents default to all zeros.
Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 128-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the RAM.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM128X1D: 128-deep by 1-wide positive edge write, asynchronous read
-- dual-port distributed LUT RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM128X1D_inst : RAM128X1D
generic map (    
  INIT => X"00000000000000000000000000000000"
) port map (  
  DPO => DPO, -- Read/Write port 1-bit output
  SPO => SPO, -- Read port 1-bit output
  A => A, -- Read/Write port 7-bit address input
  D => D, -- RAM data input
  DPRA => DPRA, -- Read port 7-bit address input
  WCLK => WCLK, -- Write clock input
  WE => WE -- RAM data input
);
```

-- End of RAM128X1D_inst instantiation

Verilog Instantiation Template

```verilog
// RAM128X1D: 128-deep by 1-wide positive edge write, asynchronous read
// dual-port distributed LUT RAM
// Spartan-6

RAM128X1D #(    
  .INIT(128'h00000000000000000000000000000000)
) RAM128X1D_inst (   
  .DPO(DPO), -- Read port 1-bit output
  .SPO(SPO), -- Read/write port 1-bit output
  .A(A), -- Read/write port 7-bit address input
  .D(D), -- RAM data input
  .DPRA(DPRA), -- Read port 7-bit address input
  .WCLK(WCLK), -- Write clock input
  .WE(WE) -- Write enable input
);
```

// End of RAM128X1D_inst instantiation
For More Information

- See the *Spartan-6 FPGA Configurable Logic Block User Guide*.
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics*. 
Chapter 4: About Design Elements

RAM256X1S

Primitive: 256-Deep by 1-Wide Random Access Memory (Select RAM)

Introduction

This design element is a 256-bit deep by 1-bit wide random access memory with synchronous write and asynchronous read capability. This RAM is implemented using the LUT resources of the device (also known as Select RAM), and does not consume any of the block RAM resources of the device. If a synchronous read capability is preferred, a register can be attached to the output and placed in the same slice as long as the same clock is used for both the RAM and the register. The RAM256X1S has an active, High write enable, WE, so that when that signal is High, and a rising edge occurs on the WCLK pin, a write is performed recording the value of the D input data pin into the memory array. The output O displays the contents of the memory location addressed by A, regardless of the WE value. When a write is performed, the output is updated to the new value shortly after the write completes.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
<td>1</td>
<td>Read/Write port data output addressed by A</td>
</tr>
<tr>
<td>D</td>
<td>Input</td>
<td>1</td>
<td>Write data input addressed by A</td>
</tr>
<tr>
<td>A</td>
<td>Input</td>
<td>8</td>
<td>Read/Write port address bus</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>1</td>
<td>Write Enable</td>
</tr>
<tr>
<td>WCLK</td>
<td>Input</td>
<td>1</td>
<td>Write clock (reads are asynchronous)</td>
</tr>
</tbody>
</table>

Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

If instantiated, the following connections should be made to this component:

- Tie the WCLK input to the desired clock source, the D input to the data source to be stored, and the O output to an FDCE D input or other appropriate data destination.
- The WE clock enable pin should be connected to the proper write enable source in the design.
- The 8-bit A bus should be connected to the source for the read/write.
- An optional INIT attribute consisting of a 256-bit Hexadecimal value can be specified to indicate the initial contents of the RAM.

If left unspecified, the initial contents default to all zeros.
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 256-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the RAM.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM256X1S: 256-deep by 1-wide positive edge write, asynchronous read
-- single-port distributed LUT RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM256X1S_inst : RAM256X1S
generic map (  
  INIT => X"0000000000000000000000000000000000000000"
)
port map (  
  O => O, -- Read/Write port 1-bit output  
  A => A, -- Read/Write port 8-bit address input  
  D => D, -- RAM data input  
  WCLK => WCLK, -- Write clock input  
  WE => WE -- Write enable input
 );

-- End of RAM256X1S_inst instantiation
```

Verilog Instantiation Template

```verilog
// RAM256X1S: 256-deep by 1-wide positive edge write, asynchronous read
// single-port distributed LUT RAM
// Spartan-6

RAM256X1S #(  
  .INIT(256'h00000000000000000000000000000000)
) RAM256X1S_inst (  
  .O(O), // Read/Write port 1-bit output  
  .A(A), // Read/Write port 8-bit address input  
  .WE(WE), // Write enable input  
  .WCLK(WCLK), // Write clock input  
  .D(D) // RAM data input
 );

// End of RAM256X1S_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).

Spartan-6 Libraries Guide for HDL Designs

UG615 (v 13.3) October 26, 2011  www.xilinx.com
Chapter 4: About Design Elements

RAM32M

Primitive: 32-Deep by 8-bit Wide Multi Port Random Access Memory (Select RAM)

Introduction

This design element is a 32-bit deep by 8-bit wide, multi-port, random access memory with synchronous write and asynchronous independent, 2-bit, wide-read capability. This RAM is implemented using the LUT resources of the device known as SelectRAM™, and does not consume any of the Block RAM resources of the device. The RAM32M is implemented in a single slice and consists of one 8-bit write, 2-bit read port and three separate 2-bit read ports from the same memory. This configuration allows for byte-wide write and independent 2-bit read access RAM. If the DIA, DIB, DIC and DID inputs are all tied to the same data inputs, the RAM can become a 1 read/write port, 3 independent read port, 32x2 quad port memory. If DID is grounded, DOD is not used, while ADDRA, ADDRB and ADDRC are tied to the same address, the RAM becomes a 32x6 simple dual port RAM. If ADDRD is tied to ADDRA, ADDRB, and ADDRC, then the RAM is a 32x8 single port RAM. There are several other possible configurations for this RAM.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOA</td>
<td>Output</td>
<td>2</td>
<td>Read port data outputs addressed by ADDRA</td>
</tr>
<tr>
<td>DOB</td>
<td>Output</td>
<td>2</td>
<td>Read port data outputs addressed by ADDRB</td>
</tr>
<tr>
<td>DOC</td>
<td>Output</td>
<td>2</td>
<td>Read port data outputs addressed by ADDRC</td>
</tr>
<tr>
<td>DOD</td>
<td>Output</td>
<td>2</td>
<td>Read/Write port data outputs addressed by ADDRD</td>
</tr>
<tr>
<td>DIA</td>
<td>Input</td>
<td>2</td>
<td>Write data inputs addressed by ADDRD (read output is addressed by ADDRA)</td>
</tr>
<tr>
<td>DIB</td>
<td>Input</td>
<td>2</td>
<td>Write data inputs addressed by ADDRD (read output is addressed by ADDRB)</td>
</tr>
<tr>
<td>DIC</td>
<td>Input</td>
<td>2</td>
<td>Write data inputs addressed by ADDRD (read output is addressed by ADDRC)</td>
</tr>
<tr>
<td>DID</td>
<td>Input</td>
<td>2</td>
<td>Write data inputs addressed by ADDRD</td>
</tr>
<tr>
<td>ADDRA</td>
<td>Input</td>
<td>5</td>
<td>Read address bus A</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

This element can be inferred by some synthesis tools by describing a RAM with a synchronous write and asynchronous read capability. Consult your synthesis tool documentation for details on RAM inference capabilities and coding examples. Xilinx suggests that you instantiate RAM32Ms if you have a need to implicitly specify the RAM function, or if you need to manually place or relatively place the component. If a synchronous read capability is desired, the RAM32M outputs can be connected to an FDRSE (FDCPE is asynchronous set/reset is necessary) in order to improve the output timing of the function. However, this is not necessary for the proper operation of the RAM.

If you want to have the data clocked on the negative edge of a clock, an inverter can be described on the clock input to this component. This inverter will be absorbed into the block, giving you the ability to write to the RAM on falling clock edges.

If instantiated, the following connections should be made to this component. Tie the WCLK input to the desired clock source, the DIA, DIB, DIC and DID inputs to the data source to be stored and the DOA, DOB, DOC and DOD outputs to an FDCE D input or other appropriate data destination or left unconnected if not used. The WE clock enable pin should be connected to the proper write enable source in the design. The 5-bit ADDRD bus should be connected to the source for the read/write addressing and the 5-bit ADDRA, ADDRB and ADDRDC buses should be connected to the appropriate read address connections. The optional INIT_A, INIT_B, INIT_C and INIT_D attributes consisting of a 64-bit hexadecimal values that specifies each port's initial memory contents can be specified. The INIT value correlates to the RAM addressing by the following equation: ADDRy[z] = INIT_y[2*z+1:2*z]. For instance, if the RAM ADDRC port is addressed to 00001, then the INIT_C[3:2] values would be the initial values shown on the DOC port before the first write occurs at that address. If left unspecified, the initial contents will default to all zeros.

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_A</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the RAM on the A port.</td>
</tr>
<tr>
<td>INIT_B</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the RAM on the B port.</td>
</tr>
<tr>
<td>INIT_C</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the RAM on the C port.</td>
</tr>
<tr>
<td>INIT_D</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the RAM on the D port.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM32M: 32-deep by 8-wide Multi Port LUT RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM32M_inst : RAM32M
generic map (
  INIT_A => X"0000000000000000", -- Initial contents of A port
  INIT_B => X"0000000000000000", -- Initial contents of B port
  INIT_C => X"0000000000000000", -- Initial contents of C port
  INIT_D => X"0000000000000000", -- Initial contents of D port
) port map (DOA => DOA, -- Read port A 2-bit output
  DOB => DOB, -- Read port B 2-bit output
  DOC => DOC, -- Read port C 2-bit output
  DOD => DOD, -- Read/Write port D 2-bit output
  ADDR_A => ADDRA, -- Read port A 5-bit address input
  ADDR_B => ADDRB, -- Read port B 5-bit address input
  ADDR_C => ADDRC, -- Read port C 5-bit address input
  ADDR_D => ADDRD, -- Read/Write port D 5-bit address input
  DIA => DIA, -- RAM 2-bit data write input addressed by ADDRD,
            -- read addressed by ADDRA
  DIB => DIB, -- RAM 2-bit data write input addressed by ADDRD,
            -- read addressed by ADDRB
  DIC => DIC, -- RAM 2-bit data write input addressed by ADDRD,
            -- read addressed by ADDRC
  DID => DID, -- RAM 2-bit data write input addressed by ADDRD,
            -- read addressed by ADDRD
  WCLK => WCLK, -- Write clock input
  WE => WE    -- Write enable input)

-- End of RAM32M_inst instantiation
```
Verilog Instantiation Template

// RAM32M: 32-deep by 8-wide Multi Port LUT RAM
// Spartan-6

RAM32M #( 
    .INIT_A(64'h0000000000000000), // Initial contents of A Port
    .INIT_B(64'h0000000000000000), // Initial contents of B Port
    .INIT_C(64'h0000000000000000), // Initial contents of C Port
    .INIT_D(64'h0000000000000000)  // Initial contents of D Port
) RAM32M_inst(
    .DOA(DOA), // Read port A 2-bit output
    .DOB(DOB), // Read port B 2-bit output
    .DOC(DOC), // Read port C 2-bit output
    .DOD(DOD), // Read/write port D 2-bit output
    .ADDRA(ADDRA), // Read port A 5-bit address input
    .ADDRB(ADDRB), // Read port B 5-bit address input
    .ADDRC(ADDRC), // Read port C 5-bit address input
    .ADDRD(ADDRD), // Read/write port D 5-bit address input
    .DIA(DIA), // RAM 2-bit data write input addressed by ADDRD, // read addressed by ADDRA
    .DIB(DIB), // RAM 2-bit data write input addressed by ADDRD, // read addressed by ADDRB
    .DIC(DIC), // RAM 2-bit data write input addressed by ADDRD, // read addressed by ADDRC
    .DID(DID), // RAM 2-bit data write input addressed by ADDRD, // read addressed by ADDRD
    .WCLK(WCLK), // Write clock input
    .WE(WE) // Write enable input
);

// End of RAM32M_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
RAM32X1D

Primitive: 32-Deep by 1-Wide Static Dual Port Synchronous RAM

Introduction

The design element is a 32-word by 1-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA4:DPRA0) and the write address (A4:A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 5-bit write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block. You can initialize RAM32X1D during configuration using the INIT attribute. Mode selection is shown in the following logic table.

The SPO output reflects the data in the memory cell addressed by A4:A0. The DPO output reflects the data in the memory cell addressed by DPRA4:DPRA0. The write process is not affected by the address on the read address port.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (Mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All Zeros</td>
<td>Initializes ROMs, RAMs, registers, and look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM32X1D: 32 x 1 positive edge write, asynchronous read
-- dual-port distributed RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM32X1D_inst : RAM32X1D
generic map (    
  INIT => X"00000000") -- Initial contents of RAM
port map (    
  DPO => DPO,     -- Read-only 1-bit data output
  SPO => SPO,     -- R/W 1-bit data output
  A0 => A0,       -- R/W address[0] input bit
  A1 => A1,       -- R/W address[1] input bit
  A2 => A2,       -- R/W address[2] input bit
  A3 => A3,       -- R/W address[3] input bit
  A4 => A4,       -- R/W address[4] input bit
  D => D,         -- Write 1-bit data input
  DPRA0 => DPRA0, -- Read-only address[0] input bit
  DPRA1 => DPRA1, -- Read-only address[1] input bit
  DPRA2 => DPRA2, -- Read-only address[2] input bit
  DPRA3 => DPRA3, -- Read-only address[3] input bit
  DPRA4 => DPRA4, -- Read-only address[4] input bit
  WCLK => WCLK,   -- Write clock input
  WE => WE        -- Write enable input
);

-- End of RAM32X1D_inst instantiation
```
Verilog Instantiation Template

// RAM32X1D: 32 x 1 positive edge write, asynchronous read dual-port distributed RAM
// Spartan-6

RAM32X1D #( .INIT(32'h00000000) ) // Initial contents of RAM
RAM32X1D_inst ( .DPO(DPO), // Read-only 1-bit data output
                .SPO(SPO), // Rw/ 1-bit data output
                .A0(A0),  // Rw/ address[0] input bit
                .D(D),    // Write 1-bit data input
                .DPRA0(DPRA0), // Read-only address[0] input bit
                .DPRA1(DPRA1), // Read-only address[1] input bit
                .DPRA2(DPRA2), // Read-only address[2] input bit
                .DPRA3(DPRA3), // Read-only address[3] input bit
                .DPRA4(DPRA4), // Read-only address[4] input bit
                .WCLK(WCLK), // Write clock input
                .WE(WE)     // Write enable input
  );

// End of RAM32X1D_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
Chapter 4: About Design Elements

RAM32X1S

Primitive: 32-Deep by 1-Wide Static Synchronous RAM

Introduction

The design element is a 32-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any positive transition on (WCLK) loads the data on the data input (D) into the word selected by the 5-bit address (A4-A0). For predictable performance, address and data inputs must be stable before a Low-to-High (WCLK) transition. This RAM block assumes an active-High (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins. You can initialize RAM32X1S during configuration using the INIT attribute.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WCLK</td>
</tr>
<tr>
<td>WE (Mode)</td>
<td>0 (read)</td>
</tr>
<tr>
<td></td>
<td>1 (read)</td>
</tr>
<tr>
<td></td>
<td>1 (read)</td>
</tr>
<tr>
<td></td>
<td>1 (write)</td>
</tr>
<tr>
<td></td>
<td>1 (read)</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Yes</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
<td></td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>Specifies initial contents of the RAM.</td>
</tr>
</tbody>
</table>
# VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM32X1S: 32 x 1 posedge write distributed (LUT) RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
RAM32X1S_inst : RAM32X1S
  generic map
    (INIT => X"00000000")
  port map
    (O => O,
     A0 => A0,
     A1 => A1,
     A2 => A2,
     A3 => A3,
     A4 => A4,
     D => D,
     WCLK => WCLK,
     WE => WE)
);  
-- End of RAM32X1S_inst instantiation
```

# Verilog Instantiation Template

```
// RAM32X1S: 32 x 1 posedge write distributed (LUT) RAM
// Spartan-6

RAM32X1S #(  
  .INIT(32'h00000000) // Initial contents of RAM
) RAM32X1S_inst (  
  .O(O), // RAM output
  .A0(A0), // RAM address[0] input
  .D(D), // RAM data input
  .WCLK(WCLK), // Write clock input
  .WE(WE) // Write enable input
);  
// End of RAM32X1S_inst instantiation
```

# For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
**RAM32X1S_1**

Primitive: 32-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock

**Introduction**

The design element is a 32-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any negative transition on (WCLK) loads the data on the data input (D) into the word selected by the 5-bit address (A4:A0). For predictable performance, address and data inputs must be stable before a High-to-Low (WCLK) transition. This RAM block assumes an active-Low (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins. You can initialize RAM32X1S_1 during configuration using the INIT attribute.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WE (Mode)</strong></td>
<td><strong>WCLK</strong></td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A4:A0

**Design Entry Method**

| Instantiation | Yes |
| Inference | Recommended |
| CORE Generator™ and wizards | No |
| Macro support | No |

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>0</td>
<td>Initializes RAMs, registers, and look-up tables.</td>
</tr>
</tbody>
</table>
**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM32X1S_1: 32 x 1 negedge write distributed (LUT) RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM32X1S_1_inst : RAM32X1S_1
generic map (
  INIT => X"00000000"
) port map (  
  O => O,   -- RAM output
  A0 => A0,   -- RAM address[0] input
  A1 => A1,   -- RAM address[1] input
  A2 => A2,   -- RAM address[2] input
  A3 => A3,   -- RAM address[3] input
  A4 => A4,   -- RAM address[4] input
  D => D,   -- RAM data input
  WCLK => WCLK,   -- Write clock input
  WE => WE   -- Write enable input
);

-- End of RAM32X1S_1_inst instantiation
```

**Verilog Instantiation Template**

```verilog
// RAM32X1S_1: 32 x 1 negedge write distributed (LUT) RAM
// Spartan-6

RAM32X1S_1 #(  
  .INIT(32'h00000000) // Initial contents of RAM
)RAM32X1S_1_inst (  
  .O(O),   // RAM output
  .A0(A0),   // RAM address[0] input
  .D(D),   // RAM data input
  .WCLK(WCLK),   // Write clock input
  .WE(WE)   // Write enable input
);

// End of RAM32X1S_1_inst instantiation
```

**For More Information**

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](www.xilinx.com).
**Chapter 4: About Design Elements**

**RAM32X2S**

Primitive: 32-Deep by 2-Wide Static Synchronous RAM

**Introduction**

The design element is a 32-word by 2-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any positive transition on (WCLK) loads the data on the data input (D1-D0) into the word selected by the 5-bit address (A4-A0). For predictable performance, address and data inputs must be stable before a Low-to-High (WCLK) transition. This RAM block assumes an active-High (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block. The signal output on the data output pins (O1-O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can use the INIT_00 and INIT_01 properties to specify the initial contents of RAM32X2S.

**Logic Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (Mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A4:A0

**Design Entry Method**

| Instantiation | Yes |
| Inference | Recommended |
| CORE Generator™ and wizards | No |
| Macro support | No |
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_00</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>INIT for bit 0 of RAM.</td>
</tr>
<tr>
<td>INIT_01</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>INIT for bit 1 of RAM.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
-- RAM32X2S: 32 x 2 posedge write distributed (LUT) RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM32X2S_inst : RAM32X2S
generic map (
  INIT_00 => X"00000000", -- INIT for bit 0 of RAM
  INIT_01 => X"00000000") -- INIT for bit 1 of RAM
port map (
  O0 => O0,  -- RAM data[0] output
  O1 => O1,  -- RAM data[1] output
  A0 => A0,  -- RAM address[0] input
  A1 => A1,  -- RAM address[1] input
  A2 => A2,  -- RAM address[2] input
  A3 => A3,  -- RAM address[3] input
  A4 => A4,  -- RAM address[4] input
  D0 => D0,  -- RAM data[0] input
  D1 => D1,  -- RAM data[1] input
  WCLK => WCLK, -- Write clock input
  WE => WE -- Write enable input
);
-- End of RAM32X2S_inst instantiation
```

Verilog Instantiation Template

```verilog
// RAM32X2S: 32 x 2 posedge write distributed (LUT) RAM
// Spartan-6

RAM32X2S #(  
  .INIT_00(32'h00000000), // INIT for bit 0 of RAM
  .INIT_01(32'h00000000) // INIT for bit 1 of RAM
) RAM32X2S_inst (  
  .O0(O0), // RAM data[0] output
  .O1(O1), // RAM data[1] output
  .A0(A0), // RAM address[0] input
  .D0(D0), // RAM data[0] input
  .D1(D1), // RAM data[1] input
  .WCLK(WCLK), // Write clock input
  .WE(WE) // Write enable input
);
// End of RAM32X2S_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](https://www.xilinx.com).
RAM64M

Primitive: 64-Deep by 4-bit Wide Multi Port Random Access Memory (Select RAM)

Introduction

This design element is a 64-bit deep by 4-bit wide, multi-port, random access memory with synchronous write and asynchronous independent bit wide read capability. This RAM is implemented using the LUT resources of the device (also known as SelectRAM™) and does not consume any of the block RAM resources of the device. The RAM64M component is implemented in a single slice, and consists of one 4-bit write, 1-bit read port, and three separate 1-bit read ports from the same memory allowing for 4-bit write and independent bit read access RAM. If the DIA, DIB, DIC and DID inputs are all tied to the same data inputs, the RAM can become a 1 read/write port, 3 independent read port 64x1 quad port memory. If DID is grounded, DOD is not used. While ADDRA, ADDRB and ADDRC are tied to the same address the RAM becomes a 64x3 simple dual port RAM. If ADDRD is tied to ADDRA, ADDRB, and ADDRC; then the RAM is a 64x4 single port RAM. There are several other possible configurations for this RAM.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOA</td>
<td>Output</td>
<td>1</td>
<td>Read port data outputs addressed by ADDRA</td>
</tr>
<tr>
<td>DOB</td>
<td>Output</td>
<td>1</td>
<td>Read port data outputs addressed by ADDRB</td>
</tr>
<tr>
<td>DOC</td>
<td>Output</td>
<td>1</td>
<td>Read port data outputs addressed by ADDRC</td>
</tr>
<tr>
<td>DOD</td>
<td>Output</td>
<td>1</td>
<td>Read/Write port data outputs addressed by ADDRD</td>
</tr>
<tr>
<td>DIA</td>
<td>Input</td>
<td>1</td>
<td>Write data inputs addressed by ADDRD (read output is addressed by ADDRA)</td>
</tr>
<tr>
<td>DIB</td>
<td>Input</td>
<td>1</td>
<td>Write data inputs addressed by ADDRD (read output is addressed by ADDRB)</td>
</tr>
<tr>
<td>DIC</td>
<td>Input</td>
<td>1</td>
<td>Write data inputs addressed by ADDRD (read output is addressed by ADDRC)</td>
</tr>
</tbody>
</table>
### Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

This element can be inferred by some synthesis tools by describing a RAM with a synchronous write and asynchronous read capability. Consult your synthesis tool documentation for details on RAM inference capabilities and coding examples. Xilinx suggests that you instantiate RAM64Ms if you have a need to implicitly specify the RAM function, or if you need to manually place or relationally place the component. If a synchronous read capability is desired, the RAM64M outputs can be connected to an FDRSE (FDCE is asynchronous set/reset is necessary) in order to improve the output timing of the function. However, this is not necessary for the proper operation of the RAM. If you want to have the data clocked on the negative edge of a clock, an inverter can be described on the clock input to this component. This inverter will be absorbed into the block giving the ability to write to the RAM on falling clock edges.

If instantiated, the following connections should be made to this component. Tie the WCLK input to the desired clock source, the DIA, DIB, DIC and DID inputs to the data source to be stored and the DOA, DOB, DOC and DOD outputs to an FDCE D input or other appropriate data destination or left unconnected if not used. The WE clock enable pin should be connected to the proper write enable source in the design. The 5-bit ADDRD bus should be connected to the source for the read/write addressing and the 5-bit ADDRA, ADDRb and ADDRC buses should be connected to the appropriate read address connections. The optional INIT_A, INIT_B, INIT_C and INIT_D attributes consisting of a 64-bit hexadecimal values that specifies each port’s initial memory contents can be specified. The INIT value correlates to the RAM addressing by the following equation: ADDRy[z] = INIT_y[z].

For instance, if the RAM ADDRC port is addressed to 00001, then the INIT_C[1] values would be the initial values shown on the DOC port before the first write occurs at that address. If left unspecified, the initial contents will default to all zeros.

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_A</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zero</td>
<td>Specifies the initial contents of the RAM on the A port.</td>
</tr>
<tr>
<td>INIT_B</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zero</td>
<td>Specifies the initial contents of the RAM on the B port.</td>
</tr>
<tr>
<td>INIT_C</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zero</td>
<td>Specifies the initial contents of the RAM on the C port.</td>
</tr>
<tr>
<td>INIT_D</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zero</td>
<td>Specifies the initial contents of the RAM on the D port.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM64M: 64-deep by 4-wide Multi Port LUT RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
RAM64M_inst : RAM64M
generic map (  
  INIT_A => X"0000000000000000", -- Initial contents of A port  
  INIT_B => X"0000000000000000", -- Initial contents of B port  
  INIT_C => X"0000000000000000", -- Initial contents of C port  
  INIT_D => X"0000000000000000") -- Initial contents of D port
port map (  
  DOA => DOA, -- Read port A 1-bit output  
  DOB => DOB, -- Read port B 1-bit output  
  DOC => DOC, -- Read port C 1-bit output  
  DOD => DOD, -- Read/Write port D 1-bit output  
  ADDRA => ADDRA, -- Read port A 6-bit address input  
  ADDRB => ADDRB, -- Read port B 6-bit address input  
  ADDRC => ADDRC, -- Read port C 6-bit address input  
  ADDRD => ADDRD, -- Read/Write port D 6-bit address input  
  DIA => DIA, -- RAM 1-bit data write input addressed by ADDRD, -- read addressed by ADDRA  
  DIB => DIB, -- RAM 1-bit data write input addressed by ADDRD, -- read addressed by ADDRB  
  DIC => DIC, -- RAM 1-bit data write input addressed by ADDRD, -- read addressed by ADDRC  
  DID => DID, -- RAM 1-bit data write input addressed by ADDRD, -- read addressed by ADDRD  
  WCLK => WCLK, -- Write clock input  
  WE => WE -- Write enable input
);
```

-- End of RAM64M_inst instantiation
Verilog Instantiation Template

// RAM64M: 64-deep by 4-wide Multi Port LUT RAM
// Spartan-6

RAM64M #(
    .INIT_A(64'h0000000000000000), // Initial contents of A Port
    .INIT_B(64'h0000000000000000), // Initial contents of B Port
    .INIT_C(64'h0000000000000000), // Initial contents of C Port
    .INIT_D(64'h0000000000000000)   // Initial contents of D Port
) RAM64M_inst (
    .DOA(DOA),  // Read port A 1-bit output
    .DOB(DOB),  // Read port B 1-bit output
    .DOC(DOC),  // Read port C 1-bit output
    .DOD(DOD),  // Read/write port D 1-bit output
    .DIA(DIA),  // RAM 1-bit data write input addressed by ADDRD,
    // read addressed by ADDRA
    .DIB(DIB),  // RAM 1-bit data write input addressed by ADDRD,
    // read addressed by ADDR B
    .DIC(DIC),  // RAM 1-bit data write input addressed by ADDRD,
    // read addressed by ADDRC
    .DID(DID),  // RAM 1-bit data write input addressed by ADDRD,
    // read addressed by ADDRD
    .ADDRA(ADDRA), // Read port A 6-bit address input
    .ADDRB(ADDRB), // Read port B 6-bit address input
    .ADDRC(ADDRC), // Read port C 6-bit address input
    .ADDRD(ADDRD), // Read/write port D 6-bit address input
    .WE(WE),    // Write enable input
    .WCLK(WCLK) // Write clock input
);

// End of RAM64M_inst instantiation

For More Information

• See the Spartan-6 FPGA Configurable Logic Block User Guide.
• See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
RAM64X1D

Primitive: 64-Deep by 1-Wide Dual Port Static Synchronous RAM

Introduction

This design element is a 64-word by 1-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA5:DPRA0) and the write address (A5:A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected.

When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 6-bit (A0:A5) write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The SPO output reflects the data in the memory cell addressed by A5:A0. The DPO output reflects the data in the memory cell addressed by DPRA5:DPRA0.

Note  The write process is not affected by the address on the read address port.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A5:A0
data_d = word addressed by bits DPRA5:DPRA0
Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td></td>
</tr>
<tr>
<td>Recommended</td>
<td></td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Initializes RAMs, registers, and look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM64X1D: 64 x 1 negative edge write, asynchronous read
-- dual-port distributed RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
RAM64X1D_1_inst : RAM64X1D_1
generic map (    INIT => X"0000000000000000") -- Initial contents of RAM
port map (    DPO => DPO,    -- Read-only 1-bit data output
              SPO => SPO,    -- R/W 1-bit data output
              A0 => A0,     -- R/W address[0] input bit
              A1 => A1,     -- R/W address[1] input bit
              A2 => A2,     -- R/W address[2] input bit
              A3 => A3,     -- R/W address[3] input bit
              A4 => A4,     -- R/W address[4] input bit
              A5 => A5,     -- R/W address[5] input bit
              D => D,       -- Write 1-bit data input
              DPRA0 => DPRA0, -- Read-only address[0] input bit
              DPRA1 => DPRA1, -- Read-only address[1] input bit
              DPRA2 => DPRA2, -- Read-only address[2] input bit
              DPRA3 => DPRA3, -- Read-only address[3] input bit
              DPRA4 => DPRA4, -- Read-only address[4] input bit
              DPRA5 => DPRA5, -- Read-only address[5] input bit
              WCLK => WCLK, -- Write clock input
              WE => WE      -- Write enable input        |
    );
-- End of RAM64X1D_1_inst instantiation
```
Verilog Instantiation Template

// RAM64X1D: 64 x 1 positive edge write, asynchronous read dual-port distributed RAM
// Spartan-6

RAM64X1D #( \n  .INIT(64'h0000000000000000) // Initial contents of RAM \n ) RAM64X1D_inst ( \n  .DPO(DPO), // Read-only 1-bit data output  
  .SPO(SPO), // Rw/ 1-bit data output  
  .A0(A0), // Rw/ address[0] input bit  
  .D(D), // Write 1-bit data input  
  .DPRA0(DPRA0), // Read-only address[0] input bit  
  .DPRA1(DPRA1), // Read-only address[1] input bit  
  .DPRA2(DPRA2), // Read-only address[2] input bit  
  .DPRA3(DPRA3), // Read-only address[3] input bit  
  .DPRA4(DPRA4), // Read-only address[4] input bit  
  .DPRA5(DPRA5), // Read-only address[5] input bit  
  .WCLK(WCLK), // Write clock input  
  .WE(WE) // Write enable input \n );

// End of RAM64X1D_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
**Chapter 4: About Design Elements**

**RAM64X1S**

**Primitive: 64-Deep by 1-Wide Static Synchronous RAM**

---

**Introduction**

This design element is a 64-word by 1-bit static random access memory (RAM) with synchronous write capability. When the write enable is set Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is set High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 6-bit address (A5:A0). This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize this element during configuration using the INIT attribute.

**Logic Table**

Mode selection is shown in the following logic table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A5:A0

**Design Entry Method**

<table>
<thead>
<tr>
<th>Method</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>
Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Initializes ROMs, RAMs, registers, and look-up tables.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM64X1S: 64 x 1 positive edge write, asynchronous read single-port distributed RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM64X1S_inst : RAM64X1S
generic map (  
  INIT => X"0000000000000000")
port map (  
  O => O,       -- 1-bit data output
  A0 => A0,     -- Address[0] input bit
  A1 => A1,     -- Address[1] input bit
  A2 => A2,     -- Address[2] input bit
  A3 => A3,     -- Address[3] input bit
  A4 => A4,     -- Address[4] input bit
  A5 => A5,     -- Address[5] input bit
  D => D,       -- 1-bit data input
  WCLK => WCLK, -- Write clock input
  WE => WE      -- Write enable input
);

-- End of RAM64X1S_inst instantiation

Verilog Instantiation Template

// RAM64X1S: 64 x 1 positive edge write, asynchronous read single-port distributed RAM
// Spartan-6

RAM64X1S #(
  .INIT(64'h0000000000000000) // Initial contents of RAM
) RAM64X1S_inst (  
  .O(O),     // 1-bit data output
  .A0(A0),   // Address[0] input bit
  .D(D),     // 1-bit data input
  .WCLK(WCLK), // Write clock input
  .WE(WE)    // Write enable input
);

// End of RAM64X1S_inst instantiation

For More Information

- See the Spartan-6 FPGA Configurable Logic Block User Guide.
- See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
RAM64X1S_1

Primitive: 64-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock

Introduction

This design element is a 64-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any negative transition on (WCLK) loads the data on the data input (D) into the word selected by the 6-bit address (A5:A0). For predictable performance, address and data inputs must be stable before a High-to-Low (WCLK) transition. This RAM block assumes an active-Low (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize this element during configuration using the INIT attribute.

Logic Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A5:A0

Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Initializes ROMs, RAMs, registers, and look-up tables.</td>
</tr>
</tbody>
</table>
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- RAM64X1S_1: 64 x 1 negative edge write, asynchronous read single-port distributed RAM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAM64X1S_1_inst : RAM64X1S_1
generic map
  (INIT => X"0000000000000000")
port map
  (O => O, -- 1-bit data output
   A0 => A0, -- Address[0] input bit
   A1 => A1, -- Address[1] input bit
   A2 => A2, -- Address[2] input bit
   A3 => A3, -- Address[3] input bit
   A4 => A4, -- Address[4] input bit
   A5 => A5, -- Address[5] input bit
   D => D, -- 1-bit data input
   WCLK => WCLK, -- Write clock input
   WE => WE -- Write enable input);

-- End of RAM64X1S_1_inst instantiation

Verilog Instantiation Template

// RAM64X1S_1: 64 x 1 negative edge write, asynchronous read single-port distributed RAM
// Spartan-6

RAM64X1S_1 #( .INIT(64'h0000000000000000) // Initial contents of RAM
  ) RAM64X1S_1_inst ( .O(O), // 1-bit data output
     .A0(A0), // Address[0] input bit
     .D(D), // 1-bit data input
     .WCLK(WCLK), // Write clock input
     .WE(WE) // Write enable input
  );

// End of RAM64X1S_1_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
**RAMB16BWER**

*Primitive: 16K-bit Data and 2K-bit Parity Configurable Synchronous Dual Port Block RAM with Optional Output Registers*

---

**Introduction**

This design element contains several block RAM memories that can be configured as general-purpose 16kb data + 2kb parity RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. This component can be configured and used as a 1-bit wide by 16K deep to a 36-bit wide by 512 deep, single-port or dual port RAM. Both read and write operations are fully synchronous to the supplied clock(s) to the component. However, Port A and Port B can operate fully independently and asynchronously to each other, accessing the same memory array. When these ports are configured in the wider data width modes, byte-enable write operations are possible. This RAM also offers a configurable output register that can be enabled to improve clock-to-out times of the RAM while incurring an extra clock cycle of latency during the read operation.

**Port Descriptions**

The following table shows the necessary input and output connections for the variable input ports for each DATA_WIDTH value for either Port A or Port B.
<table>
<thead>
<tr>
<th>DATA_WIDTH Value</th>
<th>DI, DIP Connections</th>
<th>ADDR Connections</th>
<th>WE Connections</th>
</tr>
</thead>
</table>

Alternatively, the older RAMB16_Sm_Sn and RAMB16BWER_Sm_Sn elements can be instantiated if the output registers are not necessary. If any of these components are used, the software will automatically retarget them to a properly configured RAMB16BWER element.

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRA[13:0]</td>
<td>Input</td>
<td>14</td>
<td>Port A address input bus. MSB always exists on ADDRA[13] while the LSB is determined by the settings for DATA_WIDTH_A.</td>
</tr>
<tr>
<td>ADDRB[13:0]</td>
<td>Input</td>
<td>14</td>
<td>Port B address input bus. MSB always exists on ADDRB[13] while the LSB is determined by the settings for DATA_WIDTH_B.</td>
</tr>
<tr>
<td>CLKA</td>
<td>Input</td>
<td>1</td>
<td>Port A clock input.</td>
</tr>
<tr>
<td>CLKB</td>
<td>Input</td>
<td>1</td>
<td>Port B clock input.</td>
</tr>
<tr>
<td>DIA[31:0]</td>
<td>Input</td>
<td>32</td>
<td>Port A data input bus.</td>
</tr>
<tr>
<td>DIB[31:0]</td>
<td>Input</td>
<td>32</td>
<td>Port B data input bus.</td>
</tr>
<tr>
<td>DIPA[3:0]</td>
<td>Input</td>
<td>4</td>
<td>Port A parity input bus.</td>
</tr>
<tr>
<td>DIPB[3:0]</td>
<td>Input</td>
<td>4</td>
<td>Port B parity input bus.</td>
</tr>
<tr>
<td>DOA[31:0]</td>
<td>Output</td>
<td>32</td>
<td>Port A data output bus.</td>
</tr>
<tr>
<td>DOB[31:0]</td>
<td>Output</td>
<td>32</td>
<td>Port B data output bus.</td>
</tr>
<tr>
<td>ENA</td>
<td>Input</td>
<td>1</td>
<td>Port A enable.</td>
</tr>
<tr>
<td>ENB</td>
<td>Input</td>
<td>1</td>
<td>Port B enable.</td>
</tr>
<tr>
<td>REGCEA</td>
<td>Input</td>
<td>1</td>
<td>Output register clock enable.</td>
</tr>
<tr>
<td>REGCEB</td>
<td>Input</td>
<td>1</td>
<td>Output register clock enable.</td>
</tr>
<tr>
<td>RSTA</td>
<td>Input</td>
<td>1</td>
<td>Port A output registers set/reset. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.</td>
</tr>
<tr>
<td>RSTB</td>
<td>Input</td>
<td>1</td>
<td>Port B output registers set/reset. This reset is configurable to be synchronous or asynchronous, depending on the value of the RSTTYPE attribute.</td>
</tr>
<tr>
<td>WEA[3:0]</td>
<td>Input</td>
<td>4</td>
<td>Port A byte-wide write enable.</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_WIDTH_A</td>
<td>Integer</td>
<td>0, 1, 2, 4, 9, 18, 36</td>
<td>0</td>
<td>Specifies the configurable data width for port A. Need not equal the width for port B.</td>
</tr>
<tr>
<td>DATA_WIDTH_B</td>
<td>Integer</td>
<td>0, 1, 2, 4, 9, 18, 36</td>
<td>0</td>
<td>Specifies the configurable data width for port B. Need not equal the width for port A.</td>
</tr>
<tr>
<td>DOA_REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Set to 1 to use the A port output registers.</td>
</tr>
<tr>
<td>DOB_REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Set to 1 to use the B port output registers.</td>
</tr>
<tr>
<td>EN_RSTRAM_A</td>
<td>String</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Enables A port RST capability when equal to FALSE and enables this capability when equal to TRUE.</td>
</tr>
<tr>
<td>EN_RSTRAM_B</td>
<td>String</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Enables B port RST capability when equal to FALSE and enables this capability when equal to TRUE.</td>
</tr>
<tr>
<td>INIT_A</td>
<td>Hexadecimal</td>
<td>36'h0000000000 to 36'hffffffff</td>
<td>All zeros</td>
<td>Specifies the initial value on the port A output after configuration.</td>
</tr>
<tr>
<td>INIT_B</td>
<td>Hexadecimal</td>
<td>36'h0000000000 to 36'hffffffff</td>
<td>All zeros</td>
<td>Specifies the initial value on the Port B output after configuration.</td>
</tr>
<tr>
<td>INIT_FILE</td>
<td>String</td>
<td>String representing file name and location</td>
<td>NONE</td>
<td>File name of file used to specify initial RAM contents.</td>
</tr>
<tr>
<td>INIT_00 to INIT_3F</td>
<td>Hexadecimal</td>
<td>Any 256 bit value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the 16 kb data memory array.</td>
</tr>
<tr>
<td>INITP_01 to INITP_07</td>
<td>Hexadecimal</td>
<td>Any 256 bit value</td>
<td>All zeros</td>
<td>Specifies the initial contents of the 2 kb parity data memory array.</td>
</tr>
<tr>
<td>RST_PRIORITY_A</td>
<td>String</td>
<td>CE, SR</td>
<td>CE</td>
<td>When DOA_REG=0, selects the priority between the A port RAM EN and RST pin. When DOA_REG=1 (using the optional output register), selects priority between REGCE and RST pin.</td>
</tr>
</tbody>
</table>

Connect all necessary inputs to the desired signals in the design. The CLKA/CLKB clock signals must be tied to an active clock for RAM operation, and the SRA/SRB reset signals must be either tied to a logic zero or to the proper reset signal. ENA/ENB must either be tied to a logic one or a proper RAM port enable signal. REGCEA and REGCEB must be tied to the proper output register clock enable, or a logic one if the respective DOA_REG or DOB_REG attribute is set to 1. If DOA_REG is set to 0, then REGCEA and REGCEB must be set to a logic 0.

Refer to the DATA_WIDTH column in the Port Description table (above) for the necessary data input, data output, write enable and address connection information for each DATA_WIDTH setting, since the necessary connections for these signals change, based on this attribute. All other output signals can be left unconnected (open) and all unused input signals should be tied to a logic zero.

Available Attributes
### Chapter 4: About Design Elements

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST_PRIORITY_B</td>
<td>String</td>
<td>CE, SR</td>
<td>CE</td>
<td>When DOB_REG=0, selects the priority between the B port RAM EN and RST pin. When DOB_REG=1 (using the optional output register), selects priority between REGCE and RST pin.</td>
</tr>
<tr>
<td>RSTTYPE</td>
<td>String</td>
<td>SYNC, ASYNC</td>
<td>SYNC</td>
<td>Selects whether the RAM outputs should have a synchronous or asynchronous reset capability. Due to improved timing and circuit stability, it is recommended to always have this set to &quot;SYNC&quot; unless an asynchronous reset is absolutely necessary.</td>
</tr>
</tbody>
</table>
| SIM_COLLISION_CHECK    | String    | ALL, GENERATE_X_ONLY, WARNING_ONLY, NONE | ALL     | Allows modification of the simulation behavior so that if a memory collision occurs:  
  - ALL - Warning produced and affected outputs/memory go unknown (X).  
  - WARNING_ONLY - Warning produced and affected outputs/memory retain last value.  
  - GENERATE_X_ONLY - No warning, but affected outputs/memory go unknown (X).  
  - NONE - No warning and affected outputs/memory retain last value.  
  **Note** Setting this to a value other than ALL can allow problems in the design to go unnoticed during simulation. Care should be taken when changing the value of this attribute. |
| SRVAL_A                | Hexadecimal | 36'h000000000 to 36'hfffffffff | All zeros | Specifies the output value of Port A upon the assertion of the reset (RSTA) signal.                                                               |
| SRVAL_B                | Hexadecimal | 36'h000000000 to 36'hfffffffff | All zeros | Specifies the output value of Port B upon the assertion of the reset (RSTB) signal.                                                              |
| WRITE_MODE_A           | String    | WRITE_FIRST, READ_FIRST, NO_CHANGE | WRITE_FIRST | Specifies output behavior of the port being written to:  
  - WRITE_FIRST - Written value appears on output port of the RAM.  
  - READ_FIRST - Previous RAM contents for that memory location appear on the output port.  
  - NO_CHANGE - Previous value on the output port remains the same. |
| WRITE_MODE_B           | String    | WRITE_FIRST, READ_FIRST, NO_CHANGE | WRITE_FIRST | Specifies output behavior of the port being written to:  
  - WRITE_FIRST - Written value appears on output port of the RAM.  
  - READ_FIRST - Previous RAM contents for that memory location appear on the output port.  
  - NO_CHANGE - Previous value on the output port remains the same. |
Chapter 4: About Design Elements

VHDL Instantiation Template
Unless they already exist, copy the following two statements and paste them before the entity declaration.
Library UNISIM;
use UNISIM.vcomponents.all;
-- RAMB16BWER: 16k-bit Data and 2k-bit Parity Configurable Synchronous Dual Port Block RAM with Optional Output Registers

-Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
RAMB16BWER_inst : RAMB16BWER
generic map (
-- DATA_WIDTH_A/DATA_WIDTH_B: 0, 1, 2, 4, 9, 18, or 36
DATA_WIDTH_A => 0,
DATA_WIDTH_B => 0,
-- DOA_REG/DOB_REG: Optional output register (0 or 1)
DOA_REG => 0,
DOB_REG => 0,
-- EN_RSTRAM_A/EN_RSTRAM_B: Enable/disable RST
EN_RSTRAM_A => TRUE,
EN_RSTRAM_B => TRUE,
-- INITP_00 to INITP_07: Initial memory contents.
INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
-- INIT_00 to INIT_3F: Initial memory contents.
INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",

278

www.xilinx.com

Spartan-6 Libraries Guide for HDL Designs
UG615 (v 13.3) October 26, 2011


Chapter 4: About Design Elements

Sparc Designs Libraries

Port DIPB input:

Port B input (each)

Port WEB =>

RSTB, B RSTB input

REGCEB, enable register input:

1-bit enable ENB port

Port CLKB, clock input:

Signals:

Address/Control --

DIPA --

32-bit (each)

--

4-bit input:

Port WEA, byte-wide enable

--

Enable A enable port

Port ADDRA, input:

14-bit signals A (each)

Control 14-bit Signals:

output B output data =>

B Port --

4-bit DOPA, A data

32-bit output:

Data:

A =>

"WRITE_FIRST", "READ_FIRST", "NONE"

SRVAL_A =>

Set/Reset

"SPARTAN6" to SIM_DEVICE:

RST_PRIORITY_A/RST_PRIORITY_B: "SYNC", RSTTYPE =

"SYNC",

--

RST_PRIORITY_A/RST_PRIORITY_B: "CE" or "SR"

RST_PRIORITY_A =

"CE",

RST_PRIORITY_B =

"CE",

--

SIM_COLLISION_CHECK: Collision check enable "ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE"

SIM_COLLISION_CHECK =

"ALL",

--

SIM_DEVICE: Must be set to "SPARTAN6" for proper simulation behavior

SIM_DEVICE =

"SPARTAN3ADS9",

--

SRVAL_A/ SRVAL_B: Set/Reset value for RAM output

SRVAL_A =

X"0000000000",

SRVAL_B =

X"0000000000",

--

WRITE_MODE_A/ WRITE_MODE_B: "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"

WRITE_MODE_A =

"WRITE_FIRST",

WRITE_MODE_B =

"WRITE_FIRST"

)

port map (  
  -- Port A Data: 32-bit (each) output: Port A data  
  DOA => DOA, -- 32-bit output: A port address output  
  DOPA => DOPA, -- 4-bit output: A port parity output  
  -- Port B Data: 32-bit (each) output: Port B data  
  DOB => DOB, -- 32-bit output: B port data output  
  DOPB => DOPB, -- 4-bit output: B port parity output  
  -- Port A Address/Control Signals: 14-bit (each) input: Port A address and control signals  
  ADDRA => ADDRA, -- 14-bit input: A port address input  
  CLKA => CLKA, -- 1-bit input: A port clock input  
  ENA => ENA, -- 1-bit input: A port enable input  
  REGCEA => REGCEA, -- 1-bit input: A port register clock enable input  
  RSTA => RSTA, -- 1-bit input: A port register set/reset input  
  WEI => WEI, -- 4-bit input: Port A byte-wide write enable input  
  -- Port A Data: 32-bit (each) input: Port A data  
  DIA => DIA, -- 32-bit input: A port data input  
  DIPA => DIPA, -- 4-bit input: A port parity input  
  -- Port B Address/Control Signals: 14-bit (each) input: Port B address and control signals  
  ADDRB => ADDRB, -- 14-bit input: B port address input  
  CLKB => CLKB, -- 1-bit input: B port clock input  
  ENB => ENB, -- 1-bit input: B port enable input  
  REGCEB => REGCEB, -- 1-bit input: B port register clock enable input  
  RSTB => RSTB, -- 1-bit input: B port register set/reset input  
  WEB => WEB, -- 4-bit input: Port B byte-wide write enable input  
  -- Port B Data: 32-bit (each) input: Port B data  
  DIB => DIB, -- 32-bit input: B port data input  
  DIPB => DIPB -- 4-bit input: B port parity input )

Spartan-6 Libraries Guide for HDL Designs

UG615 (v 13.3) October 26, 2011 www.xilinx.com 279
Verilog Instantiation Template

// RAMB16BWER: 16k-bit Data and 2k-bit Parity Configurable Synchronous Dual Port Block RAM with Optional Output Registers
// Spartan-6

RAMB16BWER #(
    // DATA_WIDTH_A/DATA_WIDTH_B: 0, 1, 2, 4, 9, 18, or 36
    .DATA_WIDTH_A(0),
    .DATA_WIDTH_B(0),
    // DOA_REG/DOB_REG: Optional output register (0 or 1)
    .DOA_REG(0),
    .DOB_REG(0),
    // EN_RSTRAM_A/EN_RSTRAM_B: Enable/disable RST
    .EN_RSTRAM_A("TRUE"),
    .EN_RSTRAM_B("TRUE"),
    // INITP_00 to INITP_07: Initial memory contents.
    .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
    // INITP_00 to INITP_3F: Initial memory contents.
    .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
)
Signals: and Address/Control (each)
.DOPB(DOPB), parity 4-bit
.RSTTYPE("SYNC"), or
.RSTTYPE("SYNC"), or
.RST_PRIORITY_A/RST_PRIORITY_B: "CE" or "SR"
.RST_PRIORITY_A/RST_PRIORITY_B: "CE", or "SR"
.RST_PRIORITY_A/RST_PRIORITY_B: "CE", or "SR"

// SIM_COLLISION_CHECK: Collision check enable "ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE"
.SIM_COLLISION_CHECK("ALL"),
.SIM_COLLISION_CHECK("ALL"),

// SIM_DEVICE: Must be set to "SPARTAN6" for proper simulation behavior
.SIM_DEVICE("SPARTAN6ADSP"),
.SIM_DEVICE("SPARTAN6ADSP"),

.SRVAL_A(36'h000000000),
.SRVAL_B(36'h000000000),

// WRITE_MODE_A/WRITE_MODE_B: "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
.WRITE_MODE_A("WRITE_FIRST"),
.WRITE_MODE_A("WRITE_FIRST"),

.WRITE_MODE_B("WRITE_FIRST")
)

RAMB16WBR_inst ( 
  // Port A Data: 32-bit (each) output: Port A data
  .DOA(DOA), // 32-bit output: A port data output
  .DOPA(DOPA), // 4-bit output: A port parity output
  // Port B Data: 32-bit (each) output: Port B data
  .DOB(DOB), // 32-bit output: B port data output
  .DOPB(DOPB), // 4-bit output: B port parity output
  // Port A Address/Control Signals: 14-bit (each) input: Port A address and control signals
  .ADDRA(ADDRA), // 14-bit input: A port address input
  .CLKA(CCLKA), // 1-bit input: A port clock input
  .ENA(ENA), // 1-bit input: A port enable input
  .REGCEA(EREGCEA), // 1-bit input: A port register clock enable input
  .RSTA(RSTA), // 1-bit input: A port register set/reset input
  .WEA(WEA), // 4-bit input: Port A byte-wide write enable input
  // Port A Data: 32-bit (each) input: Port A data
  .DIA(DIA), // 32-bit input: A port data input
  .DIPA(DIPA), // 4-bit input: A port parity input
  // Port B Address/Control Signals: 14-bit (each) input: Port B address and control signals
  .ADDRB(ADDRB), // 14-bit input: B port address input
  .CLKB(CCLKB), // 1-bit input: B port clock input
  .ENB(ENB), // 1-bit input: B port enable input
  .REGCEB(EREGCEB), // 1-bit input: B port register clock enable input
  .RSTB(RSTB), // 1-bit input: B port register set/reset input
  .WEB(WEB), // 4-bit input: Port B byte-wide write enable input
  // Port B Data: 32-bit (each) input: Port B data
  .DIB(DIB), // 32-bit input: B port data input
  .DIPB(DIPB) // 4-bit input: B port parity input
);

// End of RAMB16WBR_inst instantiation

Chapter 4: About Design Elements
For More Information

• See the Spartan-6 FPGA Block RAM User Guide.
• See the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.
RAMB8BBWER

Primitive: 8K-bit Data and 1K-bit Parity Configurable Synchronous Dual Port Block RAM with Optional Output Registers

Introduction

Spartan®-6 devices contain several block RAM memories that can be configured as general-purpose RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. The RAMB8BBWER allows access to the block RAM in the 8KB data + 1KB parity configuration. This element can be configured and used as a 1-bit wide by 8K deep to an 18-bit wide by 512-bit deep true dual port RAM. This element can also be configured as a 36-bit wide by 246 deep simple dual port RAM. Both read and write operations are fully synchronous to the supplied clock(s) to the component. However, the READ and WRITE ports can operate fully independent and asynchronous to each other, accessing the same memory array. When configured in the wider data width modes, byte-enable write operations are possible. This RAM also offers a configurable output register that can be enabled in order to improve clock-to-out times of the RAM while incurring an extra clock cycle of latency during the read operation.
## Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRAWRADDR[12:0]</td>
<td>Input</td>
<td>13</td>
<td>Port A address input bus when RAM_MODE=TDP. MSB always exists on ADDRAWRADDR[12] while the LSB is determined by the settings for DATA_WIDTH_A. Write address input bus when RAM_MODE=SDP.</td>
</tr>
<tr>
<td>ADDRBRDADDR[12:0]</td>
<td>Input</td>
<td>13</td>
<td>Port B address input bus when RAM_MODE=TDP. MSB always exists on ADDRBRDADDR[12] while the LSB is determined by the settings for DATA_WIDTH_B. Write address input bus when RAM_MODE=SDP.</td>
</tr>
<tr>
<td>CLKAWRCLK</td>
<td>Input</td>
<td>1</td>
<td>Port A clock input/Write clock input.</td>
</tr>
<tr>
<td>CLKBRDCLK</td>
<td>Input</td>
<td>1</td>
<td>Port B clock input/Read clock input.</td>
</tr>
<tr>
<td>DIADI[15:0]</td>
<td>Input</td>
<td>16</td>
<td>Port A data input bus when RAM_MODE=TDP. Data input bus addressed by WRADDR when RAM_MODE=SDP. DIADI is the logical DI[15:0] for SDP mode.</td>
</tr>
<tr>
<td>DIBDI[15:0]</td>
<td>Input</td>
<td>16</td>
<td>Port B data input bus when RAM_MODE=TDP. Data input bus addressed by WRADDR when RAM_MODE=SDP. DIBDI is the logical DI[31:16] for SDP mode.</td>
</tr>
<tr>
<td>DIPADIP[1:0]</td>
<td>Input</td>
<td>2</td>
<td>Port A parity data input bus when RAM_MODE=TDP. Data parity input bus addressed by WRADDR when RAM_MODE=SDP. DIPADIP is the logical DI[1:0] for SDP mode.</td>
</tr>
<tr>
<td>DIPBDIP[1:0]</td>
<td>Input</td>
<td>2</td>
<td>Port B parity data input bus when RAM_MODE=TDP. Data parity input bus addressed by WRADDR when RAM_MODE=SDP. DIPBDIP is the logical DI[3:2] for SDP mode.</td>
</tr>
<tr>
<td>DOADO[15:0]</td>
<td>Output</td>
<td>16</td>
<td>Port A data output bus/Data output bus addressed by RDADDR. When RAM_MODE=SDP, DOADO is the logical DO[15:0].</td>
</tr>
<tr>
<td>DOBDO[15:0]</td>
<td>Output</td>
<td>16</td>
<td>Port B data output bus/Data output bus addressed by RDADDR. When RAM_MODE=SDP, DOBDO is the logical DO[31:16].</td>
</tr>
<tr>
<td>DOPADOP[1:0]</td>
<td>Output</td>
<td>2</td>
<td>Port A parity data output bus/Data parity output bus addressed by RDADDR. When RAM_MODE=SDP, DOPADOP is the logical DO[1:0].</td>
</tr>
<tr>
<td>DOPBDOP[1:0]</td>
<td>Output</td>
<td>2</td>
<td>Port B parity data output bus/Data parity output bus addressed by RDADDR. When RAM_MODE=SDP, DOPBDOP is the logical DO[3:2].</td>
</tr>
<tr>
<td>ENAWREN</td>
<td>Input</td>
<td>1</td>
<td>Port A RAM enable/Write enable.</td>
</tr>
<tr>
<td>ENBRDREN</td>
<td>Input</td>
<td>1</td>
<td>Port B RAM enable/Read enable.</td>
</tr>
<tr>
<td>REGCEA</td>
<td>Input</td>
<td>1</td>
<td>Port A output register clock enable input (valid only when DOA_REG=1). Not used when RAM_MODE=SDP.</td>
</tr>
<tr>
<td>REGCEBREGCE</td>
<td>Input</td>
<td>1</td>
<td>Port B output register clock enable input (valid only when DOB_REG=1). Output register clock enable input when RAM_MODE=SDP.</td>
</tr>
<tr>
<td>RSTA</td>
<td>Input</td>
<td>1</td>
<td>Port A set/reset to value indicated by SRVAL_A. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYP attribute. Affects the output value on the output registers (DOA_REG=1) as well as on the output latches. Not used when RAM_MODE=SDP.</td>
</tr>
</tbody>
</table>
### Design Entry Method

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
<td></td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Macro support</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_WIDTH_A</td>
<td>Integer</td>
<td>0, 1, 2, 4, 9, 18, 36</td>
<td>0</td>
<td>Specifies the configurable data width for port A. Need not equal the width for port B. A width of 36 is valid for SDP mode only.</td>
</tr>
<tr>
<td>DATA_WIDTH_B</td>
<td>Integer</td>
<td>0, 1, 2, 4, 9, 18, 36</td>
<td>0</td>
<td>Specifies the configurable data width for port B. Need not equal the width for port A. A width of 36 is valid for SDP mode only.</td>
</tr>
<tr>
<td>DOA_REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Set to 1 to use the A port output registers. Applies to port A in TDP mode and up to 18 low order bits (including parity bits) in SDP mode.</td>
</tr>
<tr>
<td>DOB_REG</td>
<td>Integer</td>
<td>0, 1</td>
<td>0</td>
<td>Set to 1 to use the B port output registers. Applies to port B in TDP mode and up to 18 high order bits (including parity bits) in SDP mode.</td>
</tr>
<tr>
<td>EN_RSTRAM_A</td>
<td>String</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Disables A port RST capability when equal to FALSE and enables this capability when equal to TRUE.</td>
</tr>
<tr>
<td>EN_RSTRAM_B</td>
<td>String</td>
<td>TRUE, FALSE</td>
<td>TRUE</td>
<td>Disables B port RST capability when equal to FALSE and enables this capability when equal to TRUE.</td>
</tr>
<tr>
<td>INIT_A</td>
<td>Hex-decimal</td>
<td>18'h00000 to 18'h3fff</td>
<td>All zeros</td>
<td>Specifies the initial value on the port A output after configuration. Applies to port A in TDP mode and up to 18 low order bits (including parity bits) in SDP mode.</td>
</tr>
<tr>
<td>INIT_B</td>
<td>Hex-decimal</td>
<td>18'h00000 to 18'h3fff</td>
<td>All zeros</td>
<td>Specifies the initial value on the port B output after configuration. Applies to port B in TDP mode and up to 18 high order bits (including parity bits) in SDP mode.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Allowed Values</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------</td>
<td>-----------------------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INIT_FILE</td>
<td>String</td>
<td>String representing file name and location</td>
<td>None</td>
<td>File name of file used to specify initial RAM contents.</td>
</tr>
<tr>
<td>INIT_00 to INIT_1F</td>
<td>Hexadecimal</td>
<td>Any 256 bit value</td>
<td>All zeros</td>
<td>Allows specification of the initial contents of the 8KB data memory array.</td>
</tr>
<tr>
<td>INITP_01 to INITP_03</td>
<td>Hexadecimal</td>
<td>Any 256 bit value</td>
<td>All zeros</td>
<td>Allows specification of the initial contents of the 1KB parity data memory array.</td>
</tr>
<tr>
<td>RAM_MODE</td>
<td>String</td>
<td>TDP, SDP</td>
<td>TDP</td>
<td>Select ”SDP” to configure this element as a simple dual port RAM (write-only on one port and read-only on the other). Select ”TDP” to configure this element as a true dual port RAM (read and write capability on one or both ports).</td>
</tr>
<tr>
<td>RST_PRIORITY_A</td>
<td>String</td>
<td>CE, SR</td>
<td>CE</td>
<td>When DOA_REG=0, selects the priority between the A port RAM EN and RST pin. When DOA_REG=1 (using the optional output register), selects priority between REGCE and RST pin.</td>
</tr>
<tr>
<td>RST_PRIORITY_B</td>
<td>String</td>
<td>CE, SR</td>
<td>CE</td>
<td>When DOB_REG=0, selects the priority between the B port RAM EN and RST pin. When DOB_REG=1 (using the optional output register), selects priority between REGCE and RST pin.</td>
</tr>
<tr>
<td>RSTTYPE</td>
<td>String</td>
<td>SYNC, ASYNC</td>
<td>SYNC</td>
<td>Selects whether the RAM outputs should have a synchronous or asynchronous reset capability. Due to improved timing and circuit stability, it is recommended to always have this set to ”SYNC” unless an asynchronous reset is absolutely necessary.</td>
</tr>
</tbody>
</table>
| SIM_COLLISION_CHECK    | String                | ALL, GENERATE_X_ONLY, WARNING_ONLY, NONE | ALL     | Allows modification of the simulation behavior so that if a memory collision occurs:
  - ALL - Warning produced and affected outputs/memory location go unknown (X).
  - WARNING_ONLY - Warning produced and affected outputs/memory retain last value.
  - GENERATE_X_ONLY - No warning, but affected outputs/memory go unknown (X).
  - NONE - No warning and affected outputs/memory retain last value. |

**Note** Setting this to a value other than ALL can allow problems in the design to go unnoticed during simulation. Care should be taken when changing the value of this attribute.
### Chapter 4: About Design Elements

#### Attribute | Type | Allowed Values | Default | Description
--- | --- | --- | --- | ---
**SRVAL_A** | Hexadecimal | 18'h00000 to 18'h3ffff | All zeros | Specifies the output value of Port A upon the assertion of the reset (RSTA) signal. Applies to port A in TDP mode. In SDP mode, use only SRVAL_A if the port width is 18 bits or less. If the port width is greater than 18 bits, SRVAL_A applies to the 18 low order bits (including parity bits).

**SRVAL_B** | Hexadecimal | 18'h00000 to 18'h3ffff | All zeros | Specifies the output value of Port B upon the assertion of the reset (RSTB) signal. Applies to port B in TDP mode. In SDP mode, use only SRVAL_A if the port width is 18 bits or less. If the port width is greater than 18 bits, SRVAL_B applies to the 18 high order bits (including parity bits).

**WRITE_MODE_A** | String | WRITE_FIRST, READ_FIRST, NO_CHANGE | WRITE_FIRST | Specifies output behavior of the port being written to:
- **WRITE_FIRST** - Written value appears on output port of the RAM.
- **READ_FIRST** - Previous RAM contents for that memory location appear on the output port.
- **NO_CHANGE** - Previous value on the output port remains the same.

When RAM_MODE=SDP, WRITE_MODE_A must equal "READ_FIRST" (when using a common clock on both ports) or "WRITE_FIRST" (when using different clocks on both ports).

**WRITE_MODE_B** | String | WRITE_FIRST, READ_FIRST, NO_CHANGE | WRITE_FIRST | Specifies output behavior of the port being written to:
- **WRITE_FIRST** - Written value appears on output port of the RAM.
- **READ_FIRST** - Previous RAM contents for that memory location appear on the output port.
- **NO_CHANGE** - Previous value on the output port remains the same.

When RAM_MODE=SDP, WRITE_MODE_B must equal "READ_FIRST" (when using a common clock on both ports) or "WRITE_FIRST" (when using different clocks on both ports).

---

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- RAMB8BWER: 8k-bit Data and 1k-bit Parity Configurable Synchronous Block RAM
```

---

Spartan-6 Libraries Guide for HDL Designs
UG615 (v 13.3) October 26, 2011  [www.xilinx.com](http://www.xilinx.com)
Chapter 4: About Design Elements

-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

RAMBBWER_inst : RAMBBWER
generic map (  
  -- DATA_WIDTH_A/DATA_WIDTH_B: 'If RAM_MODE="TDP": 0, 1, 2, 4, 9 or 18; If RAM_MODE="SDP": 36'  
  DATA_WIDTH_A => 0,  
  DATA_WIDTH_B => 0,  
  -- DOA_REG/DOB_REG: Optional output register (0 or 1)  
  DOA_REG => 0,  
  DOB_REG => 0,  
  -- EN_RSTRAM_A/EN_RSTRAM_B: Enable/disable RST  
  EN_RSTRAM_A => TRUE,  
  EN_RSTRAM_B => TRUE,  
  -- INITP_00 to INITP_03: Initial memory contents.  
  INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  -- INIT_00 to INIT_0F: Initial memory contents.  
  INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",  
  -- INIT_1A/INIT_1B: Initial values on output port  
  INIT_A => X"000000",  
  INIT_B => X"00000",  
  -- INIT_FILE: Not Supported  
  INIT_FILE => "NONE",  
  -- RAM_MODE: "SDP" or "TDP"  
  RAM_MODE => "TDP",  
  -- RSTTYPE: "SYNC" or "ASYNC"  
  RSTTYPE => "SYNC",  
  -- RST_PRIORITY_A/RST_PRIORITY_B: "CE" or "SR"  
  RST_PRIORITY_A => "CE",  
  RST_PRIORITY_B => "CE",  
  -- SIM_COLLISION_CHECK: Collision check enable "ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE"  
  SIM_COLLISION_CHECK => "ALL",  
  -- SRVAL_A/SRVAL_B: Set/Reset value for RAM output  
  SRVAL_A => X"000000",  
  SRVAL_B => X"000000",  
  -- WRITE_MODE_A/WRITE_MODE_B: "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"  
  WRITE_MODE_A => "WRITE_FIRST",  
  WRITE_MODE_B => "WRITE_FIRST" )
Verilog Instantiation Template

```verilog
// RAMB8BWER: 8k-bit Data and 1k-bit Parity Configurable Synchronous Block RAM
// Spartan-6

RAMB8BWER #(  
  // DATA_WIDTH_A/DATA_WIDTH_B: 'If RAM_MODE="TDP": 0, 1, 2, 4, 9 or 18; If RAM_MODE="SDP": 36
  .DATA_WIDTH_A(0),
  .DATA_WIDTH_B(0),
  // DOA_REG/DOB_REG: Optional output register (0 or 1)
  .DOA_REG(0),
  .DOB_REG(0),
  // EN_RSTRAM_A/EN_RSTRAM_B: Enable/disable RST
  .EN_RSTRAM_A("TRUE"),
  .EN_RSTRAM_B("TRUE"),
  // INIT_00 to INIT_03: Initial memory contents.
  .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
  // INIT_04 to INIT_1F: Initial memory contents.
  .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
  .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000)
); // End of RAMB8BWER_inst instantiation
```
Chapter 4: About Design Elements

...
For More Information

- See the *Spartan-6 FPGA Block RAM User Guide*
- See the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.*
Chapter 4: About Design Elements

ROM128X1

Primitive: 128-Deep by 1-Wide ROM

Introduction

This design element is a 128-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 7-bit address (A6:A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of 32 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H. An error occurs if the INIT=value is not specified.

Logic Table

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>I1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Design Entry Method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 128-Bit Value</td>
<td>All zeros</td>
<td>Specifies the contents of the ROM.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- ROM128X1: 128 x 1 Asynchronous Distributed (LUT) ROM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

ROM128X1_inst : ROM128x1
generic map (INIT => X"00000000000000000000000000000000")
port map (O => O, -- ROM output
A0 => A0, -- ROM address[0]
A1 => A1, -- ROM address[1]
A2 => A2, -- ROM address[2]
A3 => A3, -- ROM address[3]
A4 => A4, -- ROM address[4]
A5 => A5, -- ROM address[5]
A6 => A6 -- ROM address[6]
);

-- End of ROM128X1_inst instantiation
```

Verilog Instantiation Template

```verilog
// ROM128X1: 128 x 1 Asynchronous Distributed (LUT) ROM
// Spartan-6

ROM128X1 #(.
  .INIT(128'h00000000000000000000000000000000) // Contents of ROM)
) ROM128X1_inst (.
  .O(O), // ROM output
  .A0(A0), // ROM address[0]
  .A2(A2), // ROM address[2]
  .A3(A3), // ROM address[3]
);

// End of ROM128X1_inst instantiation
```
For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
Chapter 4: About Design Elements

ROM256X1

Primitive: 256-Deep by 1-Wide ROM

Introduction

This design element is a 256-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 8-bit address (A7:A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of 64 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H.

An error occurs if the INIT=value is not specified.

Logic Table

<table>
<thead>
<tr>
<th>Input</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT(1)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>INIT(2)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT(3)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(4)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT(5)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT(6)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT(7)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(8)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT(9)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT(10)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT(11)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(12)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT(13)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INIT(14)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>INIT(15)</td>
</tr>
</tbody>
</table>
Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 256-Bit Value</td>
<td>All zeros</td>
<td>Specifies the contents of the ROM.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- ROM256X1: 256 x 1 Asynchronous Distributed (LUT) ROM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

ROM256X1_inst : ROM256X1
generic map (  
  INIT => X"0000000000000000000000000000000000000000000000000000000000000000")
port map (  
  O => O,  -- ROM output
  A0 => A0,  -- ROM address[0]
  A1 => A1,  -- ROM address[1]
  A2 => A2,  -- ROM address[2]
  A3 => A3,  -- ROM address[3]
  A4 => A4,  -- ROM address[4]
  A5 => A5,  -- ROM address[5]
  A6 => A6,  -- ROM address[6]
  A7 => A7,  -- ROM address[7]
);

-- End of ROM256X1_inst instantiation

Verilog Instantiation Template

// ROM256X1: 256 x 1 Asynchronous Distributed (LUT) ROM
// Spartan-6

ROM256X1 #(  
  .INIT(256'h0000000000000000000000000000000000000000000000000000000000000000) // Contents of ROM
) ROM256X1_inst (  
  .O(O),  // ROM output
  .A0(A0),  // ROM address[0]
  .A2(A2),  // ROM address[2]
  .A3(A3),  // ROM address[3]
  .A6(A6),  // ROM address[6]
  .A7(A7)  // ROM address[7]
);

// End of ROM256X1_inst instantiation
For More Information

See the *Spartan-6 FPGA User Documentation (User Guides and Data Sheets)*.
ROM32X1

Primitive: 32-Deep by 1-Wide ROM

Introduction

This design element is a 32-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 5-bit address (A4:A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of eight hexadecimal digits that are written into the ROM from the most-significant digit A=1FH to the least-significant digit A=00H.

For example, the INIT=10A78F39 parameter produces the data stream: 0001 0000 1010 0111 1000 1111 0011 1001. An error occurs if the INIT=value is not specified.

Logic Table

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>INIT(1)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT(2)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT(3)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(4)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT(5)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT(6)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT(7)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(8)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT(9)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT(10)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT(11)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INIT(12)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>INIT(13)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INIT(14)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>INIT(15)</td>
</tr>
</tbody>
</table>
Design Entry Method

<table>
<thead>
<tr>
<th>Design Entry Method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>Specifies the contents of the ROM.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- ROM32X1: 32 x 1 Asynchronous Distributed (LUT) ROM
--   Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

ROM32X1_inst : ROM32X1
generic map (  
  INIT => "X"00000000")
port map (  
  O => O,  -- ROM output
  A0 => A0,  -- ROM address[0]
  A1 => A1,  -- ROM address[1]
  A2 => A2,  -- ROM address[2]
  A3 => A3,  -- ROM address[3]
  A4 => A4  -- ROM address[4]
);
-- End of ROM32X1_inst instantiation
```

Verilog Instantiation Template

```verilog
// ROM32X1: 32 x 1 Asynchronous Distributed (LUT) ROM
//   Spartan-6

ROM32X1 #(  
  .INIT('32'h00000000)) // Contents of ROM
) ROM32X1_inst (  
  .O(O),  // ROM output
  .A0(A0),  // ROM address[0]
  .A2(A2),  // ROM address[2]
  .A3(A3),  // ROM address[3]
);
// End of ROM32X1_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](https://www.xilinx.com).
ROM64X1

Primitive: 64-Deep by 1-Wide ROM

Introduction

This design element is a 64-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 6-bit address (A5:A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of 16 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H. An error occurs if the INIT=value is not specified.

Logic Table

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>I1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Design Entry Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Yes</td>
</tr>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 64-Bit Value</td>
<td>All zeros</td>
<td>Specifies the contents of the ROM.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- ROM64X1: 64 x 1 Asynchronous Distributed (LUT) ROM
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

ROM64X1_inst : ROM64X1
generic map (    
  INIT => X"0000000000000000"
)  
port map (    
  O => O,    
  A0 => A0,    
  A1 => A1,    
  A2 => A2,    
  A3 => A3,    
  A4 => A4,    
  A5 => A5    
);

-- End of ROM64X1_inst instantiation
```

Verilog Instantiation Template

```verilog
// ROM64X1: 64 x 1 Asynchronous Distributed (LUT) ROM
// Spartan-6

ROM64X1 #(    
  .INIT(64'h0000000000000000) // Contents of ROM
) ROM64X1_inst {    
  .O(O), // ROM output    
  .A0(A0), // ROM address[0]    
  .A2(A2), // ROM address[2]    
  .A3(A3), // ROM address[3]    
  .A5(A5) // ROM address[5]    
};

// End of ROM64X1_inst instantiation
```

For More Information

See the [Spartan-6 FPGA User Documentation (User Guides and Data Sheets)](www.xilinx.com).
SIM_CONFIG_S6

Simulation: Configuration Simulation Model

Introduction

This simulation component allows the functional simulation of many of the common configuration interface, functions and commands to assist with board-level understanding and debug of configuration behaviors. The model can also simulate some startup-up behaviors such as the global set/reset (CSR) and global 3-state (GTS) assertion in the design. This model does not map to a specific primitive in the FPGA software and cannot be directly instantiated in the design, however it can be used in conjunction with the source design if specified either in a simulation-only file like a testbench or by some means guarded from synthesis so that it is not synthesized into the design netlist. This model may be used for either functional (RTL) simulation or timing simulation. This model is also indirectly used when instantiating the ICAP_SPARTAN6 in simulating configuration access to that component.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>Output</td>
<td>1</td>
<td>This output pin is used during read back.</td>
</tr>
<tr>
<td>CSOB</td>
<td>Output</td>
<td>1</td>
<td>Parallel daisy-chain active-Low chip select output. Not used in single FPGA applications.</td>
</tr>
<tr>
<td>DONE</td>
<td>Inout</td>
<td>1</td>
<td>Active-High signal indicating configuration is complete:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 = FPGA not configured</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 = FPGA configured</td>
</tr>
<tr>
<td>CCLK</td>
<td>Input</td>
<td>1</td>
<td>Configuration clock source for all configuration modes except JTAG.</td>
</tr>
<tr>
<td>CSIB</td>
<td>Input</td>
<td>1</td>
<td>Active-Low chip select to enable the SelectMAP data bus:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 = SelectMAP data bus enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 = SelectMAP data bus disabled</td>
</tr>
<tr>
<td>D</td>
<td>Input</td>
<td>32</td>
<td>Configuration and read back data bus, clocked on the rising edge of CCLK.</td>
</tr>
<tr>
<td>INITB</td>
<td>Input</td>
<td>1</td>
<td>Before the Mode pins are sampled, INIT_B is an input that can be held Low to delay configuration. After the Mode pins are sampled, INIT_B is an open-drain, active-Low output indicating whether a CRC error occurred during configuration:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 = CRC error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 = No CRC error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When the SEU detection function is enabled, INIT_B is optionally driven Low when a read back CRC error is detected.</td>
</tr>
</tbody>
</table>
### Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>In testbench or simulation-only file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Xilinx suggests that you instantiate this in the testbench file and not an implementation file or file used during synthesis of the design. It may be used in conjunction with the design in order to help determine interaction and start-up sequences between configuration loading and device start-up. In general, a configuration bitstream file is to be used in conjunction with this model in order to observe configuration behavior.

More information on simulating and using this component can be found in the Xilinx Synthesis and Simulation Design Guide. Please refer to that guide for further detail on using this component.

### Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE_ID</td>
<td>32-bit hexadecimal</td>
<td>Valid device ID codes</td>
<td>32’h00000000</td>
<td>Specify the Device ID code for the target device. Used during bitstream processing and device identification reads.</td>
</tr>
</tbody>
</table>

### VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- SIM_CONFIG_S6: Behavioral Simulation-only Model of FPGA SelectMap Configuration
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

SIM_CONFIG_S6_inst : SIM_CONFIG_S6
generic map ( DEVICE_ID => X"00000000") -- Specifies the Pre-programmed Device ID value
port map (           
    BUSY => BUSY,   -- 1-bit output Busy pin
    CSOB => CSOB,   -- 1-bit output chip select pin
    DONE => DONE,   -- 1-bit bi-directional Done pin
    CCLK => CCLK,   -- 1-bit input configuration clock
    D => D,         -- 8-bit bi-directional configuration data
    INITB => INITB, -- 1-bit bi-directional INIT status pin
    M => M,         -- 3-bit input Mode pins
    PROGB => PROGB, -- 1-bit input Program pin
    RDWRB => RDWRB, -- 1-bit input Read/Write pin
);```

UG615 (v 13.3) October 26, 2011

www.xilinx.com
};

-- End of SIM_CONFIG_S6_inst instantiation

Verilog Instantiation Template

// SIM_CONFIG_S6: Behavioral Simulation-only Model of FPGA SelectMap Configuration
//       Spartan-6

SIM_CONFIG_S6 #(
   .DEVICE_ID(32'h00000000) // Specify DEVICE_ID
) SIM_CONFIG_S6_inst {
   .BUSY(BUSY), // 1-bit output Busy pin
   .CSOB(CSOB), // 1-bit output chip select pin
   .DONE(DONE), // 1-bit bi-directional Done pin
   .CCLK(CCLK), // 1-bit input configuration clock
   .CSIB(CSIB), // 1-bit input chip select
   .D(D), // 16-bit bi-directional configuration data
   .INITB(INITB), // 1-bit bi-directional INIT status pin
   .M(M), // 2-bit input Mode pins
   .PROGB(PROGB), // 1-bit input Program pin
   .RDWRB(RDWRB) // 1-bit input Read/write pin
);

// End of SIM_CONFIG_S6_inst instantiation

For More Information

- See the *Synthesis and Simulation Design Guide*.
- See the *Spartan-6 FPGA Configuration User Guide*. 
Chapter 4: About Design Elements

SIM_CONFIG_S6_SERIAL
Simulation: Serial Configuration Simulation Model

Introduction

This simulation component allows the functional simulation of many of the common serial configuration interface, functions and commands to assist with board-level understanding and debug of configuration behaviors. The model can also simulate some startup-up behaviors such as the global set/reset (GSR) and global 3-state (GTS) assertion in the design. This model does not map to a specific primitive in the FPGA software and cannot be directly instantiated in the design, however it can be used in conjunction with the source design if specified either in a simulation-only file like a testbench or by some means guarded from synthesis so that it is not synthesized into the design netlist. This model may be used for either functional (RTL) simulation or timing simulation.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>Inout</td>
<td>1</td>
<td>Active-High signal indicating configuration is complete:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 = FPGA not configured</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 = FPGA configured</td>
</tr>
<tr>
<td>DOUT</td>
<td>Output</td>
<td>1</td>
<td>Serial data output for downstream daisy-chained devices. Data provided on the falling edge of CCLK.</td>
</tr>
<tr>
<td>CCLK</td>
<td>Input</td>
<td>1</td>
<td>Configuration clock source for all configuration modes except JTAG.</td>
</tr>
<tr>
<td>DIN</td>
<td>Input</td>
<td>1</td>
<td>Serial configuration data input, synchronous to rising CCLK edge.</td>
</tr>
<tr>
<td>INITB</td>
<td>Input</td>
<td>1</td>
<td>Before the Mode pins are sampled, INIT_B is an input that can be held Low to delay configuration. After the Mode pins are sampled, INIT_B is an open-drain, active-Low output indicating whether a CRC error occurred during configuration:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 = CRC error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 = No CRC error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When the SEU detection function is enabled, INIT_B is optionally driven Low when a read back CRC error is detected.</td>
</tr>
<tr>
<td>M</td>
<td>Input</td>
<td>2</td>
<td>Mode pins - determine configuration mode.</td>
</tr>
<tr>
<td>PROGB</td>
<td>Input</td>
<td>1</td>
<td>Active-Low asynchronous full-chip reset.</td>
</tr>
</tbody>
</table>
**Design Entry Method**

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>In testbench or simulation-only file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Xilinx suggests that you instantiate this in the testbench file and not an implementation file or file used during synthesis of the design. It may be used in conjunction with the design in order to help determine interaction and start-up sequences between configuration loading and device start-up. In general, a configuration bitstream file is to be used in conjunction with this model in order to observe configuration behavior.

More information on simulating and using this component can be found in the *Xilinx Synthesis and Simulation Design Guide*. Please refer to that guide for further detail on using this component.

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE_ID</td>
<td>32-bit hexadecimal</td>
<td>Valid device codes</td>
<td>32'h00000000</td>
<td>Specify the Device ID code for the target device. Used during bitstream processing and device identification reads.</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- SIM_CONFIG_S6_SERIAL: Behavioral Simulation-only Model of FPGA Serial Configuration
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
SIM_CONFIG_S6_SERIAL_inst : SIM_CONFIG_S6_SERIAL
generic map (DEVICE_ID => X"00000000") -- Specifies the Pre-programmed Device ID value
port map (DONE => DONE, -- 1-bit bi-directional Done pine
  CCLK => CCLK, -- 1-bit input configuration clock
  DIN => DIN, -- 1-bit input configuration data
  INITB => INITB, -- 1-bit bi-directional INIT status pin
  M => M, -- 3-bit input Mode pins
  PROGB => PROGB -- 1-bit input Program pin
);
```

-- End of SIM_CONFIG_S6_SERIAL_inst instantiation
Verilog Instantiation Template

// SIM_CONFIG_S6_SERIAL: Behavioral Simulation-only Model of FPGA Serial Configuration
// Spartan-6
SIM_CONFIG_S6_SERIAL #(
    .DEVICE_ID(32'h00000000) // Specify DEVICE_ID
) SIM_CONFIG_S6_SERIAL_inst(
    .DONE(DONE),    // 1-bit bi-directional Done pin
    .CCLK(CCLK),    // 1-bit input configuration clock
    .DIN(DIN),      // 1-bit input configuration data
    .INITB(INITB),  // 1-bit bi-directional INIT status pin
    .M(M),          // 2-bit input Mode pins
    .PROGB(PROGB)   // 1-bit input Program pin
);

// End of SIM_CONFIG_S6_SERIAL_inst instantiation

For More Information

- See the *Synthesis and Simulation Design Guide*.
- See the *Spartan-6 FPGA Configuration User Guide*. 
SRL16E

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Clock Enable

Introduction

This design element is a shift register look-up table (LUT). The inputs A3, A2, A1, and A0 select the output length of the shift register.

The shift register can be of a fixed, static length or it can be dynamically adjusted.

- **To create a fixed-length shift register** - Drive the A3 through A0 inputs with static values. The length of the shift register can vary from 1 bit to 16 bits, as determined by the following formula: Length = (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1 If A3, A2, A1, and A0 are all zeros (0000), the shift register is one bit long. If they are all ones (1111), it is 16 bits long.

- **To change the length of the shift register dynamically** - Change the values driving the A3 through A0 inputs. For example, if A2, A1, and A0 are all ones (111) and A3 toggles between a one (1) and a zero (0), the length of the shift register changes from 16 bits to 8 bits. Internally, the length of the shift register is always 16 bits and the input lines A3 through A0 select which of the 16 bits reach the output.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the Low-to-High clock (CLK) transition. During subsequent Low-to-High clock transitions, when CE is High, data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached. When CE is Low, the register ignores clock transitions.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am</td>
<td>CE</td>
</tr>
<tr>
<td>Am</td>
<td>0</td>
</tr>
<tr>
<td>Am</td>
<td>1</td>
</tr>
<tr>
<td>m= 0, 1, 2, 3</td>
<td></td>
</tr>
</tbody>
</table>
Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Output</td>
<td>1</td>
<td>Shift register data output</td>
</tr>
<tr>
<td>D</td>
<td>Input</td>
<td>1</td>
<td>Shift register data input</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable</td>
</tr>
<tr>
<td>A</td>
<td>Input</td>
<td>4</td>
<td>Dynamic depth selection of the SRL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A=0000 ==&gt; 1-bit shift length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A=1111 ==&gt; 16-bit shift length</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

Available Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexa-decimal</td>
<td>Any 16-Bit Value</td>
<td>All zeros</td>
<td>Sets the initial value of content and output of shift register after configuration.</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- SRL16E: 16-bit shift register LUT with clock enable operating on posedge of clock
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

SRL16E_inst : SRL16E
generic map (    
    INIT => X"0000")
port map (    
    Q => Q, -- SRL data output
    A0 => A0, -- Select[0] input
    A1 => A1, -- Select[1] input
    A2 => A2, -- Select[2] input
    A3 => A3, -- Select[3] input
    CE => CE, -- Clock enable input
    CLK => CLK, -- Clock input
    D => D -- SRL data input
    );

-- End of SRL16E_inst instantiation
Verilog Instantiation Template

// SRL16E: 16-bit shift register LUT with clock enable operating on posedge of clock
// Spartan-6

SRL16E #(   
  .INIT(16'h0000) // Initial Value of Shift Register
) SRL16E_inst (   
  .Q(Q),       // SRL data output
  .A0(A0),     // Select[0] input
  .CE(CE),     // Clock enable input
  .CLK(CLK),   // Clock input
  .D(D)        // SRL data input
);

// End of SRL16E_inst instantiation

For More Information

See the Spartan-6 FPGA User Documentation (User Guides and Data Sheets).
SRLC32E

Primitive: 32 Clock Cycle, Variable Length Shift Register Look-Up Table (LUT) with Clock Enable

Introduction

This design element is a variable length, 1 to 32 clock cycle shift register implemented within a single look-up table (LUT). The shift register can be of a fixed length, static length, or it can be dynamically adjusted by changing the address lines to the component. This element also features an active, high-clock enable and a cascading feature in which multiple SRLC32Es can be cascaded in order to create greater shift lengths.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Output</td>
<td>1</td>
<td>Shift register data output</td>
</tr>
<tr>
<td>Q31</td>
<td>Output</td>
<td>1</td>
<td>Shift register cascaded output (connect to the D input of a subsequent SRLC32E)</td>
</tr>
<tr>
<td>D</td>
<td>Input</td>
<td>1</td>
<td>Shift register data input</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>Clock</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>1</td>
<td>Active high clock enable</td>
</tr>
</tbody>
</table>
| A    | Input     | 5     | Dynamic depth selection of the SRL  
                  A=00000 ==> 1-bit shift length  
                  A=11111 ==> 32-bit shift length |
**Design Entry Method**

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Recommended</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

If instantiated, the following connections should be made to this component:

- Connect the CLK input to the desired clock source, the D input to the data source to be shifted/stored and the Q output to either an FDCPE or an FDRSE input or other appropriate data destination.
- The CE clock enable pin can be connected to a clock enable signal in the design or else tied to a logic one if not used.
- The 5-bit A bus can either be tied to a static value between 0 and 31 to signify a fixed 1 to 32 bit static shift length, or else it can be tied to the appropriate logic to enable a varying shift depth anywhere between 1 and 32 bits.
- If you want to create a longer shift length than 32, connect the Q31 output pin to the D input pin of a subsequent SRLC32E to cascade and create larger shift registers.
- It is not valid to connect the Q31 output to anything other than another SRLC32E.
- The selectable Q output is still available in the cascaded mode, if needed.
- An optional INIT attribute consisting of a 32-bit Hexadecimal value can be specified to indicate the initial shift pattern of the shift register.
- (INIT[0] will be the first value shifted out.)

**Available Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Allowed Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>Hexadecimal</td>
<td>Any 32-Bit Value</td>
<td>All zeros</td>
<td>Specifies the initial shift pattern of the SRLC32E.</td>
</tr>
</tbody>
</table>

**VHDL Instantiation Template**

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- SRLC32E: 32-bit variable length shift register LUT
-- with clock enable
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

SRLC32E_inst : SRLC32E
generic map (
  INIT => X"00000000"
)
port map (
  Q => Q,  -- SRL data output
  Q31 => Q31,  -- SRL cascade output pin
  A => A,  -- 5-bit shift depth select input
  CE => CE,  -- Clock enable input
  CLK => CLK,  -- Clock input
  D => D  -- SRL data input
);

-- End of SRLC32E_inst instantiation
```

---

**Spartan-6 Libraries Guide for HDL Designs**

www.xilinx.com

UG615 (v 13.3) October 26, 2011
Verilog Instantiation Template

```verilog
// SRLC32E: 32-bit variable length cascadable shift register LUT
// with clock enable
// Spartan-6

SRLC32E #(.
    .INIT(32'h00000000) // Initial Value of Shift Register
) SRLC32E_inst (.
    .Q(Q), // SRL data output
    .Q31(Q31), // SRL cascade output pin
    .A(A), // 5-bit shift depth select input
    .CE(CE), // Clock enable input
    .CLK(CLK), // Clock input
    .D(D) // SRL data input
);

// End of SRLC32E_inst instantiation
```

For More Information

- See the [Spartan-6 FPGA Configurable Logic Block User Guide](https://www.xilinx.com).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com).
Chapter 4: About Design Elements

STARTUP_SPARTAN6

Primitive: Spartan®-6 Global Set/Reset, Global 3-State and Configuration Start-Up Clock Interface

Introduction

This design element is used to interface device pins and logic to the Global Set/Reset (GSR) signal, the Global Tristate (GTS) dedicated routing, the internal configuration signals, or the input pins for the SPI PROM if an SPI PROM is used to configure the device. This primitive can also be used to specify a different clock for the device startup sequence at the end of configuring the device, and to access the configuration clock to the internal logic.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFGCLK</td>
<td>Output</td>
<td>1</td>
<td>Configuration logic main clock output.</td>
</tr>
<tr>
<td>CFGMCLK</td>
<td>Output</td>
<td>1</td>
<td>Configuration internal oscillator clock output.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>User startup-clock input</td>
</tr>
<tr>
<td>EOS</td>
<td>Output</td>
<td>1</td>
<td>Active high output signal indicates the End Of Configuration.</td>
</tr>
<tr>
<td>GSR</td>
<td>Input</td>
<td>1</td>
<td>Global Set/Reset (GSR) input (GSR cannot be used for the port name).</td>
</tr>
<tr>
<td>GTS</td>
<td>Input</td>
<td>1</td>
<td>Global Tristate (GTS) input (GTS cannot be used for the port name).</td>
</tr>
<tr>
<td>KEYCLEARB</td>
<td>Input</td>
<td>1</td>
<td>Clear AES Decrypter Key input from Battery-Backed RAM (BBRAM).</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Recommended</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

To use the dedicated GSR circuitry, connect the sourcing pin or logic to the GSR pin. However, avoid using the GSR circuitry of this component unless certain precautions are taken first. Since the skew of the GSR net cannot be guaranteed, either use general routing for the set/reset signal in which routing delays and skew can be calculated as a part of the timing analysis of the design, or to take preventative measures to ensure that possible skew on the release of the clock cycle does not interfere with circuit operation.

Similarly, if the dedicated global 3-state is used, connect the appropriate sourcing pin or logic to the GTS input pin of the primitive. To specify a clock for the startup sequence of configuration, connect a clock from the design to the CLK pin of this design element.
VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

Library UNISIM;
use UNISIM.vcomponents.all;

-- STARTUP_SPARTAN6: STARTUP Block
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3

STARTUP_SPARTAN6_inst : STARTUP_SPARTAN6
port map (  
  CFGCLK => CFGCLK,  -- 1-bit output: Configuration logic main clock output.
  CFGMCLK => CFGMCLK,  -- 1-bit output: Configuration internal oscillator clock output.
  EOS => EOS,  -- 1-bit output: Active high output signal indicates the End Of Configuration.
  CLK => CLK,  -- 1-bit input: User startup-clock input
  GSR => GSR,  -- 1-bit input: Global Set/Reset input (GSR cannot be used for the port name)
  GTS => GTS,  -- 1-bit input: Global 3-state input (GTS cannot be used for the port name)
  KEYCLEARB => KEYCLEARB  -- 1-bit input: Clear AES Decrypter Key input from Battery-Backed RAM (BBRAM)
);

-- End of STARTUP_SPARTAN6_inst instantiation

Verilog Instantiation Template

// STARTUP_SPARTAN6: STARTUP Block
// Spartan-6

STARTUP_SPARTAN6 STARTUP_SPARTAN6_inst (  
  .CFGCLK(CFGCLK),  // 1-bit output: Configuration logic main clock output.
  .CFGMCLK(CFGMCLK),  // 1-bit output: Configuration internal oscillator clock output.
  .EOS(EOS),  // 1-bit output: Active high output signal indicates the End Of Configuration.
  .CLK(CLK),  // 1-bit input: User startup-clock input
  .GSR(GSR),  // 1-bit input: Global Set/Reset input (GSR cannot be used for the port name)
  .GTS(GTS),  // 1-bit input: Global 3-state input (GTS cannot be used for the port name)
  .KEYCLEARB(KEYCLEARB)  // 1-bit input: Clear AES Decrypter Key input from Battery-Backed RAM (BBRAM)
);

// End of STARTUP_SPARTAN6_inst instantiation

For More Information

- See the [Spartan-6 FPGA Configuration User Guide](https://www.xilinx.com)
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](https://www.xilinx.com)
SUSPEND_SYNC

Primitive: Suspend Mode Access

Introduction

The SUSPEND primitive extends the capabilities of the user to synchronize the design for applications using the suspend mode. It uses a three-pin interface to allow synchronization of the trigger to start the suspend mode, even when there are several clock domains requiring synchronization.

Port Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Input</td>
<td>1</td>
<td>User clock input.</td>
</tr>
<tr>
<td>SACK</td>
<td>Input</td>
<td>1</td>
<td>SUSPEND acknowledgement output.</td>
</tr>
<tr>
<td>SREQ</td>
<td>Output</td>
<td>1</td>
<td>Suspend request output.</td>
</tr>
</tbody>
</table>

Design Entry Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation</td>
<td>Recommended</td>
</tr>
<tr>
<td>Inference</td>
<td>No</td>
</tr>
<tr>
<td>CORE Generator™ and wizards</td>
<td>No</td>
</tr>
<tr>
<td>Macro support</td>
<td>No</td>
</tr>
</tbody>
</table>

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```vhdl
Library UNISIM;
use UNISIM.vcomponents.all;

-- SUSPEND_SYNC: Suspend Mode Access
-- Spartan-6
-- Xilinx HDL Libraries Guide, version 13.3
SUSPEND_SYNC_inst : SUSPEND_SYNC
port map (  
  SREQ => SREQ, -- 1-bit output: Suspend request output  
  CLK => CLK, -- 1-bit input: User clock input  
  SACK => SACK -- 1-bit input: SUSPEND acknowledgement output  
);
-- End of SUSPEND_SYNC_inst instantiation
```
Verilog Instantiation Template

// SUSPEND_SYNC: Suspend Mode Access
// Spartan-6

SUSPEND_SYNC SUSPEND_SYNC_inst (  
    .SREQ(SREQ), // 1-bit output: Suspend request output
    .CLK(CLK),  // 1-bit input: User clock input
    .SACK(SACK) // 1-bit input: SUSPEND acknowledgement output
);

// End of SUSPEND_SYNC_inst instantiation

For More Information

- See the [Spartan-6 FPGA Configuration User Guide](#).
- See the [Spartan-6 FPGA Data Sheet: DC and Switching Characteristics](#).