Spartan-3E Libraries Guide for HDL Designs

ISE 8.2i

XILINX®

About this Guide

The *Spartan™-3E Libraries Guide for HDL Designs* is part of the ISE documentation collection. A separate version of this guide is also available for users who prefer to work with schematics in their circuit design activities. (See the *Spartan™-3E Libraries Guide for Schematic Designs*.)

Guide Contents

This guide contains the following:

- Information about additional resources and conventions used in this guide.
- A general introduction to the Spartan-3E primitives.
- A listing of the primitives and macros that are supported by the Spartan-3E architecture, organized by functional categories.
- Individual sections for each of the primitive design elements, including VHDL and Verilog instantiation and inference code examples.
- Referrals to additional sources of information.

Additional Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/literature.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*

Convention	Meaning or Use	Example
Helvetica bold	Commands that you select from a menu	**File →Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**ngdbuild** *design_name*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **bus[7:0]**, they are required.	**ngdbuild** [*option_name*] *design_name*
Braces { }	A list of items from which you must choose one or more	**lowpwr** ={**on**|**off**}
Vertical bar |	Separates items in a list of choices	**lowpwr** ={**on**|**off**}
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	**allow block** *block_name loc1 loc2 ... locn*;

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-4 Handbook*.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Introduction

This version of the *Libraries Guide* describes the primitive and macro design elements that make up the Xilinx Unified Libraries and are supported by the Spartan-3E architecture, and includes examples of instantiation and inference code for each primitive.

Xilinx maintains software libraries with hundreds of functional design elements (primitives and macros) for a variety of 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, beginning in 2004, Xilinx developed a separate library for each architecture. This Spartan-3E guide is one in a series of architecture-specific libraries.

This guide describes the primitive elements available for Xilinx Spartan-3E FPGA devices. Common logic functions can be implemented with these elements and more complex functions can be built by combining macros and primitives.

Functional Categories

The functional categories list the available design elements in each category, along with a brief description of each element that is supported under each Xilinx architecture.

Attributes and Constraints

The terms attribute and constraint have been used interchangeably by some in the engineering community, while others ascribe different meanings to these terms. In addition, language constructs use the terms attribute and directive in similar yet different senses. For the purpose of clarification, the following distinction can be drawn between these terms.

An *attribute* is a property associated with a device architecture primitive that affects an instantiated primitive's functionality or implementation. Attributes are typically conveyed as follows:

- In VHDL, by means of generic maps.
- In Verilog, by means of defparams or inline parameter passing during the instantiation process.

Constraints impose user-defined parameters on the operation of ISE tools. There are two types of constraints:

- *Synthesis Constraints* direct the synthesis tool optimization technique for a particular design or piece of HDL code. They are either embedded within the VHDL or Verilog code, or within a separate synthesis constraints file.
- *Implementation Constraints* are instructions given to the FPGA implementation tools to direct the mapping, placement, timing, or other guidelines for the implementation tools to follow while processing an FPGA design. Implementation constraints are generally placed in the UCF file, but can exist in the HDL code, or in a synthesis constraints file.

Attributes are identified with the components to which they apply in the libraries guide for those components. Constraints are documented in the Xilinx *Constraints Guide*. Both resources are available from the Xilinx Software Manuals collection.

Table of Contents

About this Guide

Functional Categories

About the Spartan-3E Design Elements

Functional Categories

This section categorizes, by function, the Spartan-3E design elements described in detail later in this guide. The elements (primitive and macro implementations) are listed in alphanumeric order under each of the following functional categories:

Arithmetic Functions

Design Element	Description
MULT18X18SIO	Primitive: 18x18 Cascadable Signed Multiplier with Optional Input and Output registers, Clock Enable, and Synchronous Reset

Clock Components

Design Element	Description
BUFG	Primitive : Global Clock Buffer
BUFGCE	Primitive : Global Clock with Clock Enable
BUFGCE_1	Primitive : Global Clock Buffer with Clock Enable and Output State 1
BUFGMUX	Primitive : Global Clock MUX Buffer
BUFGMUX_1	Primitive : Global Clock MUX Buffer with Output State 1
DCM_SP	Primitive: Digital Clock Manager

Config/BSCAN Components

Design Element	Description
BSCAN_SPARTAN3	Primitive: Spartan-3 Boundary Scan Logic Control Circuit
CAPTURE_SPARTAN3	Primitive: Spartan-3 Register State Capture for Bitstream Readback
STARTUP_SPARTAN3E	Primitive : Spartan-3E User Interface to the GSR, GTS, Configuration Startup Sequence and Multi-Boot Trigger Circuitry

I/O Components

Design Element	Description
IBUF	Primitive : Single-Ended Input Buffer with Selectable I/O Standard
IBUFDS	Primitive : Differential Signaling Input Buffer with Selectable I/O Interface
IBUFG	Primitive : Dedicated Input Buffer with Selectable I/O Interface
IBUFGDS	Primitive : Dedicated Differential Signaling Input Buffer with Selectable I/O Interface
IDDR2	Primitive: Dual Data Rate Input D Flip-Flop with Optional Data Alignment, Clock Enable and Programmable Synchronous or Asynchronous Set/Reset

Design Element	Description
OBUF	Primitive: Single- Ended Output Buffer
ODDR2	Primitive: Dual Data Rate Output D Flip-Flop with Optional Data Alignment, Clock Enable and Programmable Synchronous or Asynchronous Set/Reset
IOBUF	Primitive : bidirectional Buffer with Selectable I/O Interface with Active Low Output Enable
IOBUFDS	Primitive : 3-State Differential Signaling I/O Buffer with Active Low Output Enable and with Selectable I/O Interface
KEEPER	Primitive : KEEPER Symbol
OBUFDS	Primitive : Differential Signaling Output Buffer with Selectable I/O Interface
OBUFT	Primitive: 3-State Output Buffer with Active Low Output Enable and with Selectable I/O Interface
OBUFTDS	Primitive : 3-State Output Buffer with Differential Signaling, Active-Low Output Enable, and Selectable I/O Interface
PULLDOWN	Primitive : Resistor to GND for Input Pads
PULLUP	Primitive : Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs

RAM/ROM

Design Element	Description
RAM16X1D	Primitive : 16-Deep by 1-Wide Static Dual Port Synchronous RAM
RAM16X1S	Primitive : 16-Deep by 1-Wide Static Synchronous RAM
RAM32X1D	Primitive : 32-Deep by 1-Wide Static Dual Static Port Synchronous RAM
RAM32X1S	Primitive: 32-Deep by 1-Wide Static Synchronous RAM
RAM64X1S	Primitive: 64-Deep by 1-Wide Static Synchronous RAM
RAM128X1S	Primitive : 128-Deep by 1-Wide Static Synchronous RAM
RAMB16_Sm_Sn	Primitive : 16384-Bit Data Memory and 2048-Bit Parity Memory, Dual-Port Synchronous Block RAM with Port Width (m or n) Configured to 1, 2, 4, 9, 18, or 36 Bits
RAMB16_Sn	Primitive : 16384-Bit Data Memory and 2048-Bit Parity Memory, Single-Port Synchronous Block RAM with Port Width (n) Configured to 1, 2, 4, 9, 18, or 36 Bits
ROM16X1	Primitive: 16-Deep by 1-Wide ROM
ROM32X1	Primitive: 32-Deep by 1-Wide ROM
ROM64X1	Primitive: 64-Deep by 1-Wide ROM
ROM128X1	Primitive: 128-Deep by 1-Wide ROM
ROM256X1	Primitive: 256-Deep by 1-Wide ROM

Registers & Latches

Design Element	Description
FDCPE	Primitive : D Flip-Flop with Clock Enable and Asynchronous Preset and Clear
FDRSE	Primitive : D Flip-Flop with Synchronous Reset and Set and Clock Enable
LDCPE	Primitive : Transparent Data Latch with Asynchronous Clear and Preset and Gate Enable

Shift Registers

Design Element	Description
SRLC16E	Primitive : 16-Bit Shift Register Look-Up-Table (LUT) with Carry and Clock Enable

Slice/CLB Primitives

Design Element	Description
LUT1	Primitive : 1-Bit Look-Up-Table with General Output
LUT2	Primitive : 2-Bit Look-Up-Table with General Output

Design Element	Description
LUT3	Primitive : 3-Bit Look-Up-Table with General Output
LUT4	Primitive : 4-Bit Look-Up-Table with General Output
LUT1_D	Primitive : 1-Bit Look-Up-Table with Dual Output
LUT2_D	Primitive : 2-Bit Look-Up-Table with Dual Output
LUT3_D	Primitive : 3-Bit Look-Up-Table with Dual Output
LUT4_D	Primitive : 4-Bit Look-Up-Table with Dual Output
LUT1_L	Primitive : 1-Bit Look-Up-Table with Local Output
LUT2_L	Primitive : 2-Bit Look-Up-Table with Local Output
LUT3_L	Primitive : 3-Bit Look-Up-Table with Local Output
LUT4_L	Primitive : 4-Bit Look-Up-Table with Local Output
MULT_AND	Primitive : Fast Multiplier AND
MUXCY	Primitive : 2-to-1 Multiplexer for Carry Logic with General Output
MUXCY_D	Primitive : 2-to-1 Multiplexer for Carry Logic with Dual Output
MUXCY_L	Primitive : 2-to-1 Multiplexer for Carry Logic with Local Output
MUXF5	Primitive : 2-to-1 Look-Up Table Multiplexer with General Output
MUXF5_D	Primitive : 2-to-1 Look-Up Table Multiplexer with Dual Output
MUXF5_L	Primitive : 2-to-1 Look-Up Table Multiplexer with Local Output
MUXF6	Primitive : 2-to-1 Look-Up Table Multiplexer with General Output
MUXF6_D	Primitive : 2-to-1 Look-Up Table Multiplexer with Dual Output
MUXF6_L	Primitive : 2-to-1 Look-Up Table Multiplexer with Local Output
MUXF7	Primitive : 2-to-1 Look-Up Table Multiplexer with General Output
MUXF7_D	Primitive : 2-to-1 Look-Up Table Multiplexer with Dual Output
MUXF7_L	Primitive : 2-to-1 Look-Up Table Multiplexer with Local Output
MUXF8	Primitive : 2-to-1 Look-Up Table Multiplexer with General Output
MUXF8_D	Primitive : 2-to-1 Look-Up Table Multiplexer with Dual Output
MUXF8_L	Primitive : 2-to-1 Look-Up Table Multiplexer with Local Output
XORCY	Primitive : XOR for Carry Logic with General Output
XORCY_D	Primitive : XOR for Carry Logic with Dual Output
XORCY_L	Primitive : XOR for Carry Logic with Local Output

About the Spartan-3E Design Elements

The remaining sections in this guide describe each primitive design element that can be used under the Spartan-3E architecture.

The design elements are organized in alphanumeric order, with all numeric suffixes in ascending order. For example, FDCPE precedes FDRSE, and IBUF precedes IBUFDS.

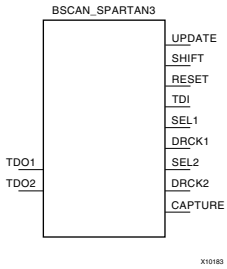
The following information is provided for each library element, where applicable:

- Name of each element.
- Description of each element, including truth tables, where applicable.
- A description of the attributes associated with each design element, where appropriate.
- Examples of VHDL and Verilog instantiation and inference code, where applicable.
- Referrals to additional sources of information.

Designers who prefer to work with schematics are encouraged to consult the *Spartan-3E Libraries Guide for Schematic Designs*.

BSCAN_SPARTAN3

Primitive: Spartan-3 Boundary Scan Logic Control Circuit



BSCAN_SPARTAN3 provides access to the BSCAN sites on a Spartan-3E device. It creates internal boundary scan chains. The 4-pin JTAG interface (TDI, TDO, TCK, and TMS) consists of dedicated pins in Spartan-3E devices. To use normal JTAG for boundary scan purposes, hook up the JTAG pins to the port and go. The pins on the BSCAN_SPARTAN3 symbol do not need to be connected, unless those special functions are needed to drive an internal scan chain.

A signal on the TDO1 input is passed to the external TDO output when the USER1 instruction is executed; the SEL1 output goes High to indicate that the USER1 instruction is active.The DRCK1 output provides USER1 access to the data register clock (generated by the TAP controller). The TDO2 and SEL2 pins perform a similar function for the USER2 instruction and the DRCK2 output provides USER2 access to the data register clock (generated by the TAP controller). The RESET, UPDATE, SHIFT, and CAPTURE pins represent the decoding of the corresponding state of the boundary scan internal state machine. The TDI pin provides access to the TDI signal of the JTAG port in order to shift data into an internal scan chain.

Usage

This design element is instantiated, rather than inferred.

VHDL Instantiation Template

```
--  BSCAN_SPARTAN3 : In order to incorporate this function into the design,
--      VHDL       : the following instance declaration needs to be placed
--    instance     : in the architecture body of the design code.  The
--   declaration   : instance name (BSCAN_SPARTAN3_inst) and/or the port declarations
--      code       : after the "=>" assignment maybe changed to properly
--                 : connect this function to the design.  Delete or comment
--                 : out inputs/outs that are not necessary.

--     Library     : In addition to adding the instance declaration, a use
--   declaration   : statement for the UNISIM.vcomponents library needs to be
--       for       : added before the entity declaration.  This library
--     Xilinx      : contains the component declarations for all Xilinx
--    primitives   : primitives and points to the models that are used
--                 : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- BSCAN_SPARTAN3: Boundary Scan primitive for connecting internal logic to
   --                 JTAG interface. Spartan-3
   -- Xilinx  HDL Libraries Guide version 8.1i

   BSCAN_SPARTAN3_inst : BSCAN_SPARTAN3
   port map (
      CAPTURE => CAPTURE, -- CAPTURE output from TAP controller
      DRCK1 => DRCK1,     -- Data register output for USER1 functions
      DRCK2 => DRCK2,     -- Data register output for USER2 functions
      RESET => RESET,     -- Reset output from TAP controller
      SEL1 => SEL1,       -- USER1 active output
      SEL2 => SEL2,       -- USER2 active output
      SHIFT => SHIFT,     -- SHIFT output from TAP controller
      TDI => TDI,         -- TDI output from TAP controller
      UPDATE => UPDATE,   -- UPDATE output from TAP controller
      TDO1 => TDO1,       -- Data input for USER1 function
```

```
      TDO2 => TDO2        -- Data input for USER2 function
   );

   -- End of BSCAN_SPARTAN3_inst instantiation
```

Verilog Instantiation Template

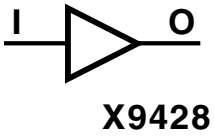
```
//  BSCAN_SPARTAN3 : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (BSCAN_SPARTAN3_inst) and/or the port declarations within the
//      code       : parenthesis maybe changed to properly reference and
//                 : connect this function to the design.  Delete or comment
//                 : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // BSCAN_SPARTAN3: Boundary Scan primitive for connecting internal logic to
   //                 JTAG interface. Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   BSCAN_SPARTAN3 BSCAN_SPARTAN3_inst (
      .CAPTURE(CAPTURE), // CAPTURE output from TAP controller
      .DRCK1(DRCK1),     // Data register output for USER1 functions
      .DRCK2(DRCK2),     // Data register output for USER2 functions
      .RESET(RESET),     // Reset output from TAP controller
      .SEL1(SEL1),       // USER1 active output
      .SEL2(SEL2),       // USER2 active output
      .SHIFT(SHIFT),     // SHIFT output from TAP controller
      .TDI(TDI),         // TDI output from TAP controller
      .UPDATE(UPDATE),   // UPDATE output from TAP controller
      .TDO1(TDO1),       // Data input for USER1 function
      .TDO2(TDO2)        // Data input for USER2 function
   );

   // End of BSCAN_SPARTAN3_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

BUFG

Primitive: Global Clock Buffer



BUFG distributes high fan-out clock signals throughout a PLD device.

In general, the BUFG component is inferred by the synthesis tool by selecting the highest fanout clocks in the design and appropriately inserting a BUFG until all global clocks have been exhausted. If you desire control over BUFG insertion, this can generally be done using synthesis directives. You can also instantiate a BUFG in the case of either controlling the insertion of the buffer or in the cases of defining more complex clocking networks using DCMs or other more advanced components. To instantiate this component, use the instantiation template within the ISE Language Template or use the code below and connect the BUFG to the appropriate clock source and destinations in the design.

BUFG is a clock buffer with one clock input and one clock output.

Usage

To use a specific type of buffer, instantiate it manually. This design element is supported for schematics and instantiation. Synthesis tools usually infer a BUFG on any clock net. If there are more clock nets than BUFGs, the synthesis tool usually instantiates BUFGs for the clocks that are most used. The BUFG contains both a BUFG and an IBUFG.

VHDL Instantiation Template

```
--    BUFG     : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (BUFG_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- BUFG: Global Clock Buffer (source by an internal signal)
   -- Xilinx HDL Libraries Guide Version 8.1i

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

   -- End of BUFG_inst instantiation
```

Verilog Instantiation Template

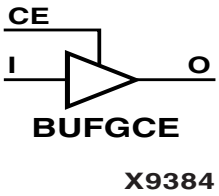
```
//    BUFG     : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (BUFG_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // BUFG: Global Clock Buffer (source by an internal signal)
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFG BUFG_inst (
      .O(O),      // Clock buffer output
      .I(I)       // Clock buffer input
   );

   // End of BUFG_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

BUFGCE

Primitive: Global Clock Buffer with Clock Enable



BUFGCE is a clock buffer with one clock input, one clock output, and a clock enable line. 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.

Inputs		Outputs
I	CE	O
X	0	0
I	1	I

Usage

This design element is supported for instantiations but not for inference.

VHDL Instantiation Template

```
--   BUFGCE     : In order to incorporate this function into the design,
--    VHDL      : the following instance declaration needs to be placed
--  instance    : in the architecture body of the design code.  The
-- declaration  : instance name (BUFGCE_inst) and/or the port declarations
--    code      : after the "=>" assignment maybe changed to properly
--              : reference and connect this function to the design.
--              : All inputs and outputs must be connected.

--   Library    : In addition to adding the instance declaration, a use
-- declaration  : statement for the UNISIM.vcomponents library needs to be
--     for      : added before the entity declaration.  This library
--   Xilinx     : contains the component declarations for all Xilinx
--  primitives  : primitives and points to the models that are used
--              : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- BUFGCE: Global Clock Buffer with Clock Enable (active high)
   --         FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   BUFGCE_inst : BUFGCE
   port map (
      O => O,   -- Clock buffer ouptput
      CE => CE, -- Clock enable input
      I => I    -- Clock buffer input
   );

   -- End of BUFGCE_inst instantiation
```

Verilog Instantiation Template

```
//   BUFGCE     : In order to incorporate this function into the design,
//   Verilog    : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration  : (BUFGCE_inst) and/or the port declarations within the
//    code      : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  All inputs
//              : and outputs must be connect.
```

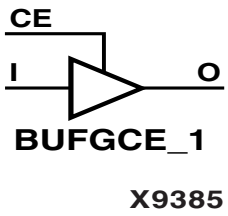
```
//  <-----Cut code below this line---->

   // BUFGCE: Global Clock Buffer with Clock Enable (active high)
   // FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFGCE BUFGCE_inst (
      .O(O),   // Clock buffer output
      .CE(CE), // Clock enable input
      .I(I)    // Clock buffer input
   );

   // End of BUFGCE_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

BUFGCE_1

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



BUFGCE is a clock buffer with one clock input, one clock output, and a clock enable line. 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.

Inputs		Outputs
I	CE	O
X	0	1
I	1	I

Usage

This design element is supported for schematics and instantiations, but not for inference.

VHDL Instantiation Template

```
--  BUFGCE_1   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (BUFGCE_1_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- BUFGCE_1: Global Clock Buffer with Clock Enable (active low)
   -- FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   BUFGCE_1_inst : BUFGCE_1
   port map (
      O => O,   -- Clock buffer ouptput
      CE => CE, -- Clock enable input
      I => I    -- Clock buffer input
   );

   -- End of BUFGCE_1_inst instantiation
```

Verilog Instantiation Template

```
//  BUFGCE_1   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
```

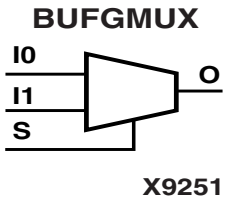
```
// declaration : (BUFGCE_1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // BUFGCE_1: Global Clock Buffer with Clock Enable (active low)
   // FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFGCE_1 BUFGCE_1_inst (
      .O(O),   // Clock buffer output
      .CE(CE), // Clock enable input
      .I(I)    // Clock buffer input
   );

   // End of BUFGCE_1_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

BUFGMUX

Primitive: Global Clock MUX Buffer



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 input. BUGFMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

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

Inputs			Outputs
I0	**I1**	**S**	**O**
I0	X	0	I0
X	I1	1	I1
X	X	↑	0
X	X	↓	0

Usage

This design element is supported for schematics and instantiations but not for inference.

VHDL Instantiation Template

```
--   BUFGMUX   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (BUFGMUX_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- BUFGMUX: Global Clock Buffer 2-to-1 MUX
   -- FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   BUFGMUX_inst : BUFGMUX
   port map (
      O => O,     -- Clock MUX output
      I0 => I0,   -- Clock0 input
      I1 => I1,   -- Clock1 input
      S => S      -- Clock select input
```

```
   );

   -- End of BUFGMUX_inst instantiation
```

Verilog Instantiation Template

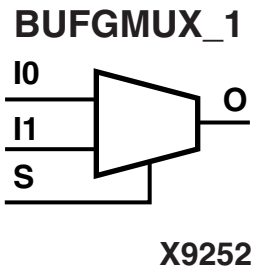
```
//   BUFGMUX   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (BUFGMUX_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // BUFGMUX: Global Clock Buffer 2-to-1 MUX
   // FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFGMUX BUFGMUX_inst (
      .O(O),    // Clock MUX output
      .I0(I0),  // Clock0 input
      .I1(I1),  // Clock1 input
      .S(S)     // Clock select input
   );

   // End of BUFGMUX_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

BUFGMUX_1

Primitive: Global Clock MUX Buffer with Output State 1



BUFGMUX_1 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 input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Inputs			Outputs
I0	**I1**	**S**	**O**
I0	X	0	I0
X	I1	1	I1
X	X	↑	1
X	X	↓	1

Usage

This design element is supported for schematics and instantiations but not for inference.

VHDL Instantiation Template

```
--  BUFGMUX_1  : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (BUFGMUX_1_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- BUFGMUX_1: Global Clock Buffer 2-to-1 MUX (inverted select)
   -- FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   BUFGMUX_1_inst : BUFGMUX_1
   port map (
      O => O,     -- Clock MUX output
      I0 => I0,   -- Clock0 input
      I1 => I1,   -- Clock1 input
      S => S      -- Clock select input
   );

   -- End of BUFGMUX_1_inst instantiation
```

Verilog Instantiation Template

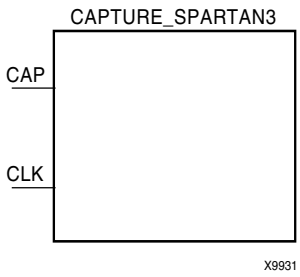
```
//  BUFGMUX_1  : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (BUFGMUX_1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // BUFGMUX_1: Global Clock Buffer 2-to-1 MUX (inverted select)
   // FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFGMUX_1 BUFGMUX_1_inst (
      .O(O),     // Clock MUX output
      .I0(I0),   // Clock0 input
      .I1(I1),   // Clock1 input
      .S(S)      // Clock select input
   );

   // End of BUFGMUX_1_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

CAPTURE_SPARTAN3

Primitive: Spartan-3 Register State Capture for Bitstream Readback



CAPTURE_SPARTAN3 devices provide user control over when to capture register (flip-flop and latch) information for readback. Spartan-3E devices provide the readback function through dedicated configuration port instructions.

The CAPTURE_SPARTAN3 symbol is optional. Without it, readback is still performed, but the asynchronous capture function it provides for register states is not available.

Spartan-3E devices allow users to capture register (flip-flop and latch) states only. Although LUT RAM, SRL, and block RAM states are read back, they cannot be captured. An asserted high CAP signal indicates that the registers in the device are to be captured at the next Low-to-High clock transition.

By default, data is captured after every trigger (transition on CLK while CAP is asserted). To limit the readback operation to a single data capture, add the ONESHOT attribute to CAPTURE_SPARTAN3 devices.

Usage

This design element is instantiated rather than inferred.

VHDL Instantiation Template

```
-- CAPTURE_SPARTAN3 : In order to incorporate this function into the design,
--      VHDL        : the following instance declaration needs to be placed
--    instance      : in the architecture body of the design code.  The
--   declaration    : instance name (CAPTURE_SPARTAN3_inst) and/or the port declarations
--      code        : after the "=>" assignment maybe changed to properly
--                  : connect this function to the design.  Delete or comment
--                  : out inputs/outs that are not necessary.

--     Library      : In addition to adding the instance declaration, a use
--   declaration    : statement for the UNISIM.vcomponents library needs to be
--       for        : added before the entity declaration.  This library
--      Xilinx      : contains the component declarations for all Xilinx
--    primitives    : primitives and points to the models that are used
--                  : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- CAPTURE_SPARTAN3: Register State Capture for Bitstream Readback
   --                   Spartan-3
   -- Xilinx  HDL Libraries Guide version 8.1i

   CAPTURE_SPARTAN3_inst : CAPTURE_SPARTAN3
   port map (
      CAP => CAP,    -- Capture input
      CLK => CLK     -- Clock input
   );

   -- End of CAPTURE_SPARTAN3_inst instantiation
```

Verilog Instantiation Template

```
// CAPTURE_SPARTAN3 : In order to incorporate this function into the design,
```

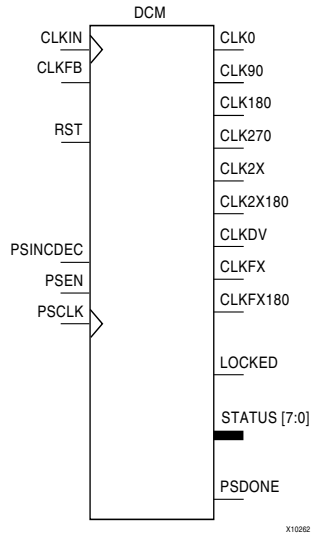
```
//     Verilog      : the following instance declaration needs to be placed
//    instance      : in the body of the design code.  The instance name
//   declaration    : (CAPTURE_SPARTAN3_inst) and/or the port declarations within the
//      code        : parenthesis maybe changed to properly reference and
//                  : connect this function to the design.  Delete or comment
//                  : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // CAPTURE_SPARTAN3: Register State Capture for Bitstream Readback
   //                   Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   CAPTURE_SPARTAN3 CAPTURE_SPARTAN3_inst (
      .CAP(CAP),   // Capture input
      .CLK(CLK)    // Clock input
   );

   // End of CAPTURE_SPARTAN3_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

DCM_SP

Primitive: Digital Clock Manager



DCM_SP is a digital clock manager that provides multiple functions. It can implement a clock delay locked loop, a digital frequency synthesizer, digital phase shifter.

Note: All unused inputs must be driven Low, automatically tying the inputs Low if they are unused. The DSSEN input pin for the DCM_SP is no longer recommended for use and should remain unconnected in the design.

Clock Delay Locked Loop (DLL)

DCM_SP includes a clock delay locked loop used to minimize clock skew for Spartan-3E devices. DCM_SP synchronizes the clock signal at the feedback clock input (CLKFB) to the clock signal at the input clock (CLKIN). The locked output (LOCKED) is high when the two signals are in phase. The signals are considered to be in phase when their rising edges are within a specified time (ps) of each other.

On-chip synchronization is achieved by connecting the CLKFB input to a point on the global clock network driven by a BUFG, a global clock buffer. The BUFG connected to the CLKFB input of the DCM_SP must be sourced from either the CLK0 or CLK2X outputs of the same DCM_SP. The CLKIN input should be connected to the output of an IBUFG, with the IBUFG input connected to a pad driven by the system clock.

Off-chip synchronization is achieved by connecting the CLKFB input to the output of an IBUFG, with the IBUFG input connected to a pad. Either the CLK0 or CLK2X output can be used but not both. The CLK0 or CLK2X must be connected to the input of OBUF, an output buffer. The CLK_FEEDBACK attribute controls whether the CLK0 output, the default, or the CLK2X output is the source of the CLKFB input.

DCM_SP Clock Delay Lock Loop Outputs

Output	Description
CLK0	Clock at 1x CLKIN frequency
CLK180	Clock at 1x CLK0 frequency, shifted 180º with regards to CLK0
CLK270	Clock at 1x CLK0 frequency, shifted 270º with regards to CLK0
CLK2X	Clock at 2x CLK0 frequency, in phase with CLK0
CLK2X180	Clock at 2x CLK0 frequency shifted 180º with regards to CLK2X
CLK90	Clock at 1x CLK0 frequency, shifted 90º with regards to CLK0
CLKDV	Clock at (1/n) x CLK0 frequency, where n=CLKDV_DIVIDE value. CLKDV is in phase with CLK0.
LOCKED	All enabled DCM_SP features locked.

Digital Frequency Synthesizer (DFS)

The CLKFX and CLKFX180 outputs in conjunction with the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes provide a frequency synthesizer that can be any multiple or division of CLKIN. CLKFX and CLKIN are in phase every CLKFX_MULTIPLY cycles of CLKFX and every CLKFX_DIVIDE cycles of CLKIN when a feedback is provided to the CLKFB input of the DLL. The frequency of CLKFX is defined by the following equation.

$$Frequency_{CLKFX} = (CLKFX\_MULTIPLY\_value / CLKFX\_DIVIDE\_value) * Frequency_{CLKIN}$$

Both the CLKFX or CLKFX180 output can be used simultaneously.

CLKFX180 is 1x the CLKFX frequency, shifted 180º with regards to CLKFX. CLKFX and CLKFX180 always have a 50/50 duty cycle.

The CLK_FEEDBACK attribute set to NONE causes the DCM_SP to be in the Digital Frequency Synthesizer mode. The CLKFX and CLKFX180 are generated without phase correction with respect to CLKIN.

The DSSEN input pin for the DCM_SP is no longer recommended for use and should remain unconnected in the design.

Digital Phase Shifter (DPS)

The phase shift (skew) between the rising edges of CLKIN and CLKFB can be configured as a fraction of the CLKIN period with the PHASE_SHIFT attribute. This allows the phase shift to remain constant as ambient conditions change. The CLKOUT_PHASE_SHIFT attribute controls the use of the PHASE_SHIFT value. By default, the CLKOUT_PHASE_SHIFT attribute is set to NONE and the PHASE_SHIFT attribute has no effect.

By creating skew between CLKIN and CLKFB, all DCM_SP output clocks are phase shifted by the amount of the skew.

When the CLKOUT_PHASE_SHIFT attribute is set to FIXED, the skew set by the PHASE_SHIFT attribute is used at configuration for the rising edges of CLKIN and CLKFB. The skew remains constant.

When the CLKOUT_PHASE_SHIFT attribute is set to VARIABLE, the skew set at configuration is used as a starting point and the skew value can be changed dynamically during operation using the PS* signals. This digital phase shifter feature is controlled by a synchronous interface. The inputs PSEN (phase shift enable) and PSINCDEC (phase shift increment/decrement) are set up to the rising edge of PSCLK (phase shift clock). The PSDONE (phase shift done) output is clocked with the rising edge of PSCLK (the phase shift clock). PSDONE must be connected to implement the complete synchronous interface. The rising-edge skew between CLKIN and CLKFB can be dynamically adjusted after the LOCKED output goes High.

The PHASE_SHIFT attribute value specifies the initial phase shift amount when the device is configured. Then the PHASE_SHIFT value is changed one unit when PSEN is activated for one period of PSCLK. The PHASE_SHIFT value is incremented when PSINCDEC is High and decremented when PSINCDEC is Low during the period that PSEN is High. When the DCM_SP completes an increment or decrement operation, the PSDONE output goes High for a single PSCLK cycle to indicate the operation is complete. At this point the next change can be made. When RST (reset) is High, the PHASE_SHIFT attribute value is reset to the skew value set at configuration.

If CLKOUT_PHASE_SHIFT is FIXED or NONE, the PSEN, PSINCDEC, and PSCLK inputs must be tied to GND. The program automatically ties the inputs to GND if they are not connected by the user.

FACTORY_JF Attribute

The FACTORY_JF attribute affects the DCM_SP's jitter filter characteristic. This attribute is set the default value of C080 and should not be modified unless otherwise instructed by Xilinx.

Additional Status Bits

The STATUS output bits return the following information.

DCM_SP Additional Status Bits

Bit	Description
0	Phase Shift Overflow* 1 = |PHASE_SHIFT| > 255
1	DLL CLKIN stopped** 1 = CLKIN stopped toggling
2	DLL CLKFX stopped 1 = CLKFX stopped toggling
3	No
4	No
5	No
6	No
7	No

* Phase Shift Overflow also goes high if the end of the phase shift delay line is reached (see the product data sheet for the value of the maximum shifting delay).

** If only the DFS outputs are used (CLKFX & CLKFX180), this status bit does not go high if CLKIN stops.

LOCKED

When LOCKED is high, all enabled signals are locked.

RST

The master reset input (RST) resets DCM_SP to its initial (power-on) state. The signal at the RST input is asynchronous and must be held High for 2 ns.

Usage

This component is instantiated in the code as it cannot be easily inferred in synthesis tools. Some synthesis tools can allow inference via an attribute. See your synthesis tool documentation.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CLK_ FEEDBACK	String	"NONE", "1X" or "2X"	"1X"	Specifies clock feedback of NONE, 1X or 2X.
CLKDV_DIVIDE	Real	1.5, 2.0, 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 or 16.0	2.0	Specifies the extent to which the DCM_SP clock divider (CLKDV output) is to be frequency divided.
CLKFX_DIVIDE	Integer	1 to 32	1	Specifies the frequency divider value for the CLKFX output.
CLKFX_MULTI-PLY	Integer	1 to 32	4	Specifies the frequency multiplier value for the CLKFX output.
CLKIN_ DIVIDE_ BY_2	Boolean	FALSE, TRUE	FALSE	Enables CLKIN divide by two features.
CLKIN_ PERIOD	Real	Any value (in ns) within the operating frequency of the device.	0	Specifies the input period to the DCM_SP CLKIN input in ns).
CLKOUT_ PHASE_ SHIFT	String	"NONE", "FIXED" or "VARIABLE"	"NONE"	Specifies the phase shift of NONE, FIXED or VARIABLE.
DESKEW_ ADJUST	String	"SOURCE_ SYNCHRONOUS", "SYSTEM_ SYNCHRONOUS" or "0" to "15"	"SYSTEM_ SYNCHRO-NOUS"	Sets configuration bits affecting the clock delay alignment between the DCM_SP output clocks and an FPGA clock input pin.
FACTORY_JF	16-Bit Hexidecimal	Any 16-Bit Hexadecimal value	x8080	The FACTORY_JF attribute affects the DCMs jitter filter characteristic. This attribute is set the default value of F0F0 and should not be modified unless otherwise instructed by Xilinx.
PHASE_SHIFT	Integer	-255 to 255	0	Defines the amount of fixed phase shift from -255 to 255
STARTUP_ WAIT	Boolean	FALSE, TRUE	FALSE	Delays configuration DONE until DCM_SP LOCK.

VHDL Instantiation Template

```
--    DCM_SP      : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (DCM_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Unused inputs
--             : and outputs can be removed or commented out.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- DCM_SP: Digital Clock Manager Circuit for Spartan-3E
   -- Xilinx HDL Libraries Guide Version 8.1i

   DCM_inst : DCM_SP
   generic map (
      CLKDV_DIVIDE => 2.0, --  Divide by: 1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,6.5
                           --     7.0,7.5,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0 or 16.0
      CLKFX_DIVIDE => 1,   --  Can be any interger from 1 to 32
      CLKFX_MULTIPLY => 4, --  Can be any Integer from 1 to 32
      CLKIN_DIVIDE_BY_2 => FALSE, --  TRUE/FALSE to enable CLKIN divide by two feature
      CLKIN_PERIOD => 0.0,          --  Specify period of input clock
      CLKOUT_PHASE_SHIFT => "NONE", --  Specify phase shift of NONE, FIXED or VARIABLE
      CLK_FEEDBACK => "1X",         --  Specify clock feedback of NONE, 1X or 2X
      DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS", --  SOURCE_SYNCHRONOUS, SYSTEM_SYNCHRONOUS or
                                             --     an Integer from 0 to 15
      DFS_FREQUENCY_MODE => "LOW",     --  HIGH or LOW frequency mode for frequency synthesis
      DLL_FREQUENCY_MODE => "LOW",     --  HIGH or LOW frequency mode for DLL
      DUTY_CYCLE_CORRECTION => TRUE, --  Duty cycle correction, TRUE or FALSE
      FACTORY_JF => X"C080",          --  FACTORY JF Values
      PHASE_SHIFT => 0,          --  Amount of fixed phase shift from -255 to 255
      STARTUP_WAIT => FALSE) --  Delay configuration DONE until DCM_SP LOCK, TRUE/FALSE
   port map (
      CLK0 => CLK0,     -- 0 degree DCM_SP CLK ouptput
      CLK180 => CLK180, -- 180 degree DCM_SP CLK output
      CLK270 => CLK270, -- 270 degree DCM_SP CLK output
      CLK2X => CLK2X,   -- 2X DCM_SP CLK output
      CLK2X180 => CLK2X180, -- 2X, 180 degree DCM_SP CLK out
      CLK90 => CLK90,   -- 90 degree DCM_SP CLK output
      CLKDV => CLKDV,   -- Divided DCM_SP CLK out (CLKDV_DIVIDE)
      CLKFX => CLKFX,   -- DCM_SP CLK synthesis out (M/D)
      CLKFX180 => CLKFX180, -- 180 degree CLK synthesis out
      LOCKED => LOCKED, -- DCM_SP LOCK status output
      PSDONE => PSDONE, -- Dynamic phase adjust done output
      STATUS => STATUS, -- 8-bit DCM_SP status bits output
      CLKFB => CLKFB,   -- DCM_SP clock feedback
      CLKIN => CLKIN,   -- Clock input (from IBUFG, BUFG or DCM_SP)
      PSCLK => PSCLK,   -- Dynamic phase adjust clock input
      PSEN => PSEN,     -- Dynamic phase adjust enable input
      PSINCDEC => PSINCDEC, -- Dynamic phase adjust increment/decrement
      RST => RST        -- DCM_SP asynchronous reset input
   );

   -- End of DCM_inst instantiation
```

Verilog Instantiation Template

```
//     DCM_SP     : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (DCM_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  Unused inputs
```

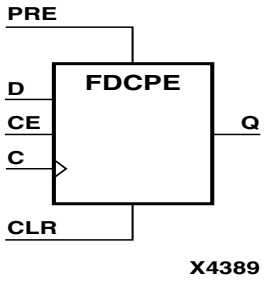
```
//             : and outputs can be removed or commented out.

//  <-----Cut code below this line---->

   // DCM_SP: Digital Clock Manager Circuit for Spartan-3E
   // Xilinx HDL Libraries Guide Version 8.1i

   DCM_SP #(
      .CLKDV_DIVIDE(2.0), // Divide by: 1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,6.5
                          //   7.0,7.5,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0 or 16.0
      .CLKFX_DIVIDE(1),    // Can be any Integer from 1 to 32
      .CLKFX_MULTIPLY(4),  // Can be any Integer from 2 to 32
      .CLKIN_DIVIDE_BY_2("FALSE"), // TRUE/FALSE to enable CLKIN divide by two feature
      .CLKIN_PERIOD(0.0),  // Specify period of input clock
      .CLKOUT_PHASE_SHIFT("NONE"), // Specify phase shift of NONE, FIXED or VARIABLE
      .CLK_FEEDBACK("1X"),  // Specify clock feedback of NONE, 1X or 2X
      .DESKEW_ADJUST("SYSTEM_SYNCHRONOUS"), // SOURCE_SYNCHRONOUS, SYSTEM_SYNCHRONOUS or
                                            //   an Integer from 0 to 15
      .DFS_FREQUENCY_MODE("LOW"),  // HIGH or LOW frequency mode for frequency synthesis
      .DLL_FREQUENCY_MODE("LOW"),  // HIGH or LOW frequency mode for DLL
      .DUTY_CYCLE_CORRECTION("TRUE"), // Duty cycle correction, TRUE or FALSE
      .FACTORY_JF(16'hC080),   // FACTORY JF values
      .PHASE_SHIFT(0),     // Amount of fixed phase shift from -255 to 255
      .STARTUP_WAIT("FALSE")   // Delay configuration DONE until DCM_SP LOCK, TRUE/FALSE
   ) DCM_inst (
      .CLK0(CLK0),     // 0 degree DCM_SP CLK output
      .CLK180(CLK180), // 180 degree DCM_SP CLK output
      .CLK270(CLK270), // 270 degree DCM_SP CLK output
      .CLK2X(CLK2X),   // 2X DCM_SP CLK output
      .CLK2X180(CLK2X180), // 2X, 180 degree DCM_SP CLK out
      .CLK90(CLK90),   // 90 degree DCM_SP CLK output
      .CLKDV(CLKDV),   // Divided DCM_SP CLK out (CLKDV_DIVIDE)
      .CLKFX(CLKFX),   // DCM_SP CLK synthesis out (M/D)
      .CLKFX180(CLKFX180), // 180 degree CLK synthesis out
      .LOCKED(LOCKED), // DCM_SP LOCK status output
      .PSDONE(PSDONE), // Dynamic phase adjust done output
      .STATUS(STATUS), // 8-bit DCM_SP status bits output
      .CLKFB(CLKFB),   // DCM_SP clock feedback
      .CLKIN(CLKIN),   // Clock input (from IBUFG, BUFG or DCM_SP)
      .PSCLK(PSCLK),   // Dynamic phase adjust clock input
      .PSEN(PSEN),     // Dynamic phase adjust enable input
      .PSINCDEC(PSINCDEC), // Dynamic phase adjust increment/decrement
      .RST(RST)        // DCM_SP asynchronous reset input
   );

   // End of DCM_inst instantiation
```

For More Information

Consult the *Spartan-3E Data Sheet*.

FDCPE

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



FDCPE is a single D-type flip-flop with data (D), clock enable (CE), asynchronous preset (PRE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High clock (C) transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For Spartan-3E devices, the power-on condition can be 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 Spartan-3E symbol.

Inputs					Outputs
CLR	PRE	CE	D	C	Q
1	X	X	X	X	0
0	1	X	X	X	1
0	0	0	X	X	No Change
0	0	1	0	↑	0
0	0	1	1	↑	1

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	1-Bit Binary	1-Bit Binary	1'b0	Sets the initial value of Q output after configuration

VHDL Instantiation Template

```
--    FDCPE    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (FDCPE_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
```

```
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- FDCPE: Single Data Rate D Flip-Flop with Asynchronous Clear, Set and
   --        Clock Enable (posedge clk).  All families.
   -- Xilinx  HDL Libraries Guide version 8.1i

   FDCPE_inst : FDCPE
   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
      PRE => PRE    -- Asynchronous set input
   );

   -- End of FDCPE_inst instantiation
```

Verilog Instantiation Template

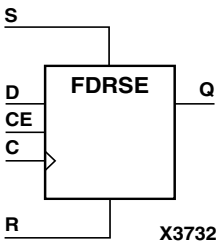
```
//     FDCPE    : In order to incorporate this function into the design,
//    Verilog   : the following instance declaration needs to be placed
//   instance   : in the body of the design code.  The instance name
//  declaration : (FDCPE_inst) and/or the port declarations within the
//     code     : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  Delete or comment
//              : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // FDCPE: Single Data Rate D Flip-Flop with Asynchronous Clear, Set and
   //        Clock Enable (posedge clk).  All families.
   // Xilinx HDL Libraries Guide Version 8.1i

   FDCPE #(
      .INIT(1'b0) // Initial value of register (1'b0 or 1'b1)
   ) FDCPE_inst (
      .Q(Q),       // Data output
      .C(C),       // Clock input
      .CE(CE),     // Clock enable input
      .CLR(CLR),   // Asynchronous clear input
      .D(D),       // Data input
      .PRE(PRE)    // Asynchronous set input
   );

   // End of FDCPE_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

FDRSE

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



FDRSE is a single D-type flip-flop with synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). The reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High clock transition. (Reset has precedence over Set.) When the set (S) input is High and R is Low, the flip-flop is set, output High, during the Low-to-High clock (C) transition. Data on the D input is loaded into the flip-flop when R and S are Low and CE is High during the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, by default, when power is applied or when GSR is active.

Inputs					Outputs
R	S	CE	D	C	Q
1	X	X	X	↑	0
0	1	X	X	↑	1
0	0	0	X	X	No Change
0	0	1	1	↑	1
0	0	1	0	↑	0

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Binary	0, 1	0	Sets the initial value of Q output after configuration and on GSR

VHDL Instantiation Template

```
--    FDRSE    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (FDCRS_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.
```

```
--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- FDRSE: Single Data Rate D Flip-Flop with Synchronous Clear, Set and
   --        Clock Enable (posedge clk).  All families.
   -- Xilinx  HDL Libraries Guide version 8.1i

   FDRSE_inst : FDRSE
   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
      D => D,       -- Data input
      R => R,       -- Synchronous reset input
      S => S        -- Synchronous set input
   );

   -- End of FDRSE_inst instantiation
```

Verilog Instantiation Template

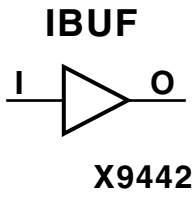
```
//     FDRSE    : In order to incorporate this function into the design,
//    Verilog   : the following instance declaration needs to be placed
//   instance   : in the body of the design code.  The instance name
//  declaration : (FDCRS_inst) and/or the port declarations within the
//     code     : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  Delete or comment
//              : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // FDRSE: Single Data Rate D Flip-Flop with Synchronous Clear, Set and
   //        Clock Enable (posedge clk).  All families.
   // Xilinx HDL Libraries Guide Version 8.1i

   FDRSE #(
      .INIT(1'b0) // Initial value of register (1'b0 or 1'b1)
   ) FDRSE_inst (
      .Q(Q),       // Data output
      .C(C),       // Clock input
      .CE(CE),     // Clock enable input
      .D(D),       // Data input
      .R(R),       // Synchronous reset input
      .S(S)        // Synchronous set input
   );

   // End of FDRSE_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

IBUF

Primitive: Single-Ended Input Buffer with Selectable I/O Standard



Input Buffers are necessary to isolate the internal circuit from the signals coming into the FPGA. IBUFs are contained in input/output blocks (IOB). IBUFs allow the specification of the particular I/O Standard to configure the I/O. In general, an IBUF should be used for all single-ended data input or bidirectional pins.

Inputs (I)	Outputs (O)
0/L	0
1/H	1
U/X/Z	X

Usage

IBUFs are automatically inserted (inferred) to any signal directly connected to a top level input or inout port of the design by the synthesis tool. It is generally recommended to allow the synthesis tool to infer this buffer however if so desired, the IBUF can be instantiated into the design. In order to do so, connect the input port, I, of the component directly to the associated top-level input or in-out port and connect the output port, O, to the FPGA logic to be 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.

Available Attribute

Attribute	Type	Allowed Values	Default	Description
IOSTANDARD	String	"DEFAULT	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.
IBUF_DELAY_VALUE	Integer	0 to 16	0	Specifies the amount of additional delay to add to the non-registered path out of the IOB.
IFD_DELAY_VALUE	String	"AUTO" or 0 to 8	"AUTO"	Specifies the amount of additional delay to add to the registered path within the IOB.

Note: Consult the device user guide or databook for the allowed values and the default value.

VHDL Instantiation Templates

```
--      IBUF       : In order to incorporate this function into the design,
--      VHDL       : the following instance declaration needs to be placed
--    instance     : in the architecture body of the design code.  The
--   declaration   : instance name (IBUF_inst) and/or the port declarations
--      code       : after the "=>" assignment maybe changed to properly
--                 : connect this function to the design.  Delete or comment
--                 : out inputs/outs that are not necessary.
```

```
--     Library     : In addition to adding the instance declaration, a use
--   declaration   : statement for the UNISIM.vcomponents library needs to be
--       for       : added before the entity declaration.  This library
--     Xilinx      : contains the component declarations for all Xilinx
--    primitives   : primitives and points to the models that are used
--                 : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- IBUF: Single-ended Input Buffer
   --       All devices
   -- Xilinx  HDL Libraries Guide version 8.1i

   IBUF_inst : IBUF
   generic map (
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 Templates

```
//      IBUF       : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (IBUF_inst) and/or the port declarations within the
//      code       : parenthesis maybe changed to properly reference and
//                 : connect this function to the design.  Delete or comment
//                 : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // IBUF: Single-ended Input Buffer
   //       All devices
   // Xilinx HDL Libraries Guide 8.1i

   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

Consult the Spartan-3E Data Sheet.

IBUFDS

Primitive: Differential Signaling Input Buffer with Selectable I/O Interface

X9255

IBUFDS 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 and MYNETB).

Inputs		Outputs
I	IB	O
0	0	No Change
0	1	0
1	0	1
1	1	No Change

Usage

This design element is supported for instantiation but not for inference.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	Boolean	FALSE, TRUE	FALSE	Enables the built-in differential termination resistor.
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.

VHDL Instantiation Template

```
--   IBUFDS    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (IBUFDS_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- IBUFDS: Differential Input Buffer
   -- FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i
```

```
   IBUFDS_inst : IBUFDS
   generic map (
      IBUF_DELAY_VALUE => "0", -- Specify the amount of added input delay for buffer, "0"-"16" (Spartan-3E
only)
      IFD_DELAY_VALUE => "AUTO", -- Specify the amount of added delay for input register, "AUTO", "0"-"8"
(Spartan-3E only)
      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

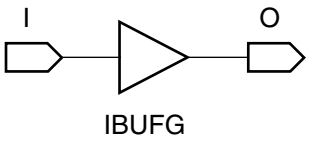
```
//   IBUFDS    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (IBUFDS_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // IBUFDS: Differential Input Buffer
   // FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   IBUFDS #(
      .DIFF_TERM("FALSE"),      // Differential Termination
     .IBUF_DELAY_VALUE("0"),   // Specify the amount of added input delay for the buffer, "0"-"16" (Spartan-
3E only)
      .IFD_DELAY_VALUE("AUTO"), // Specify the amount of added delay for input register, "AUTO", "0"-"8"
(Spartan-3E          .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

Consult the Spartan-3E Data Sheet.

IBUFG

Primitive: Dedicated Input Buffer with Selectable I/O Interface



IBUFG is dedicated to input buffers and is used for connecting to the clock buffer BUFG or DCM_SP. You can attach an IOSTANDARD attribute to an IBUFG instance.

The IBUFG input can only be driven by the global clock pins. The IBUFG output can drive CLKIN of a DCM_SP, BUFG, or user logic. IBUFG can be routed to user logic and does not have to be routed to a DCM_SP.

Attach an IOSTANDARD attribute to an IBUFG and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the input for the I/O standard associated with that value.

Usage

This design element is supported for schematic and instantiation. Synthesis tools usually infer a BUFGP on any clock net. If there are more clock nets than BUFGPs, the synthesis tool usually instantiates BUFGPs for the clocks that are most used. The BUFGP contains both a BUFG and an IBUFG.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.

VHDL Instantiation Template

```
--    IBUFG    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (IBUFG_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- IBUFG: Single-ended global clock input buffer
   --        All FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   IBUFG_inst : IBUFG
   generic map (
      IBUF_DELAY_VALUE => "0", -- Specify the amount of added input delay for buffer, "0"-"16" (Spartan-3E
only)
      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

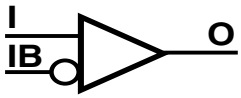
```
//    IBUFG    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (IBUFG_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // IBUFG: Global Clock Buffer (sourced by an external pin)
   //        All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   IBUFG #(
     .IOSTANDARD("DEFAULT")
   ) IBUFG_inst (
      .O(O), // Clock buffer output
      .I(I)  // Clock buffer input (connect directly to top-level port)
   );

   // End of IBUFG_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

IBUFGDS

Primitive: Dedicated Differential Signaling Input Buffer with Selectable I/O Interface



X9255

IBUFGDS is a dedicated differential signaling input buffer for connection to the clock buffer (BUFG) or DCM_SP. 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 and MYNETB).

Inputs		Outputs
I	IB	O
0	0	No Change
0	1	0
1	0	1
1	1	No Change

Usage

This design element is supported for instantiation, but not for inference.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	Boolean	FALSE, TRUE	FALSE	Enables the built-in differential termination resistor.
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.

VHDL Instantiation Template

```
--   IBUFGDS   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (IBUFGDS_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->
```

```
   -- IBUFGDS: Differential Global Clock Input Buffer
   -- FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   IBUFGDS_inst : IBUFGDS
   generic map (
      DIFF_TERM => "FALSE", -- Differential Termination
      IBUF_DELAY_VALUE => "0", -- Specify the amount of added input delay for buffer, "0"-"16" (Spartan-3E
only)
      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 IBUFGDS_inst instantiation
```

Verilog Instantiation Template

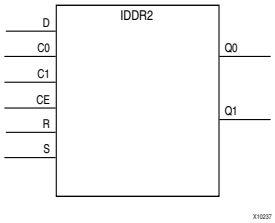
```
//   IBUFGDS   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (IBUFGDS_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // IBUFGDS: Differential Global Clock Buffer (sourced by an external pin)
   // FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   IBUFGDS #(
     .DIFF_TERM("FALSE"),
     .IOSTANDARD("DEFAULT")
   ) IBUFGDS_inst (
      .O(O),  // Clock buffer output
      .I(I),  // Diff_p clock buffer input
      .IB(IB) // Diff_n clock buffer input
   );

   // End of IBUFGDS_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

IDDR2

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



The IDDR2 is an input double data rate (DDR) register useful in capturing double data-rate signals entering the FPGA. 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 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 IDDR2 has an optional alignment feature, which allows both output data ports to the component to be aligned to a single clock.

Usage

The IDDR2 must be instantiated to be incorporated into a design. To change the default behavior of the IDDR2, attributes can be modified via the generic map (VHDL) or named parameter value assignment (Verilog) as a part of the instantiated component. The IDDR2 can be either connected directly to a top-level input port in the design where an appropriate input buffer can be inferred or 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

Attribute	Type	Allowed Values	Default	Description
DDR_ALIGN-MENT	String	"NONE", "C0" or "C1"	"NONE"	Sets output alignment.
INIT_Q0	Integer	0 or 1	0	Sets initial state of the Q0 output to 0 or 1.
INIT_Q1	Integer	0 or 1	0	Sets initial state of the Q1 output to 0 or 1.
SRTYPE	String	"SYNC" or "ASYNC"	"SYNC"	Specifies "SYNC" or "ASYNC" set/reset.

VHDL Instantiation Template

```
--      IDDR2       : In order to incorporate this function into the design,
--     VHDL         : the following instance declaration needs to be placed
--   instance       : in the architecture body of the design code.  The
--  declaration     : instance name (IDDR2_inst) and/or the port declarations
--     code         : after the "=>" assignment maybe changed to properly
--                  : connect this function to the design. All inputs
--                  : and outputs must be connected.

--     Library      : In addition to adding the instance declaration, a use
--  declaration     : statement for the UNISIM.vcomponents library needs to be
--       for        : added before the entity declaration.  This library
--    Xilinx        : contains the component declarations for all Xilinx
--   primitives     : primitives and points to the models that are used
--                  : for simulation.
```

```
--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- IDDR2: Input Double Data Rate Input Register with Set, Reset
   --        and Clock Enable. Spartan-3E
   -- Xilinx  HDL Libraries Guide version 8.1i

   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 Templae

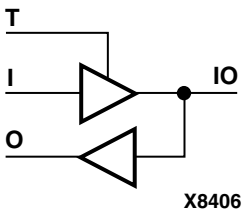
```
//      IDDR2    : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (IDDR2_inst) and/or the port declarations within the
//      code     : parenthesis maybe changed to properly reference and
//               : connect this function to the design.  Delete or comment
//               : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // IDDR2: Input Double Data Rate Input Register with Set, Reset
   //        and Clock Enable. Spartan-3E
   // Xilinx HDL Libraries Guide Version 8.1i

   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
      .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 IDDR2_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

IOBUF

Primitive: bidirectional Buffer with Selectable I/O Interface



For Spartan-3E, IOBUF is a bidirectional buffer whose I/O interface corresponds to an I/O standard. You can attach an IOSTANDARD attribute to an IOBUF instance.

IOBUF components that use the LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33 signaling standards have selectable capacitance drive and slew rates using the capacitance DRIVE and SLEW constraints.

IOBUFs are composites of IBUF and OBUFT elements. The O output is X (unknown) when I/O (input/output) is Z. IOBUFs can be implemented as interconnections of their component elements.

Inputs		Bidirectional	Outputs
T	I	IO	O
1	X	Z	X
0	1	1	1
0	0	0	0

Usage

These design elements are instantiated and inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.

VHDL Instantiation Template

```
--    IOBUF    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (IOBUF_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- IOBUF: Single-ended bidirectional Buffer
   --        All devices
```

```
   -- Xilinx  HDL Libraries Guide version 8.1i

   IOBUF_inst : IOBUF
   generic map (
      DRIVE => 12,
      IBUF_DELAY_VALUE => "0", -- Specify the amount of added input delay for buffer, "0"-"16" (Spartan-3E
only)
      IFD_DELAY_VALUE => "AUTO", -- Specify the amount of added delay for input register, "AUTO", "0"-"8"
(Spartan-3E only)
      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
   );

   -- End of IOBUF_inst instantiation
```

Verilog Instantiation Template

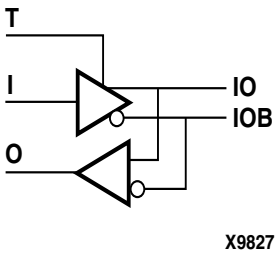
```
//      IOBUF      : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (IOBUF_inst) and/or the port declarations within the
//      code       : parenthesis maybe changed to properly reference and
//                 : connect this function to the design.  Delete or comment
//                 : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // IOBUF: Single-ended bidirectional Buffer
   //        All devices
   // Xilinx HDL Libraries Guide Version 8.1i

   IOBUF #(
      .DRIVE(12), // Specify the output drive strength
     .IBUF_DELAY_VALUE("0"),   // Specify the amount of added input delay for the buffer, "0"-"16" (Spartan-
3E only)
      .IFD_DELAY_VALUE("AUTO"), // Specify the amount of added delay for input register, "AUTO", "0"-"8"
(Spartan-3E only)
      .IOSTANDARD("DEFAULT"), // Specify the I/O standard
      .SLEW("SLOW") // Specify the output slew rate
   ) IOBUF_inst (
      .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
   );

   // End of IOBUF_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

IOBUFDS

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



IOBUFDS is a single 3-state, differential signaling input/output buffer with active Low output enable.

Inputs		Bidirectional		Outputs
I	T	IO	IOB	O
X	1	Z	Z	- *
0	0	0	1	0
1	0	1	0	1

* The dash (-) means No Change.

Usage

This design element is instantiated rather than inferred.

Available Attibutes

Attribute	Type	Allowed Values	Default	Description
DRIVE	Integer	2, 4, 6, 8, 12, 16, 24	12	Selects output drive strength (mA) for the SelectIO buffers that use the LVTTL, LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, or LVCMOS33 interface I/O standard.
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.
SLEW	String	"SLOW" or "FAST"	"SLOW"	Sets the output rise and fall time.

VHDL Instantiation Template

```
--   IOBUFDS   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (IOBUFDS_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
```

```
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- IOBUFDS: Differential bidirectional Buffer
   -- FPGA
   -- Xilinx  HDL Libraries Guide version 8.1i

   IOBUFDS_inst : IOBUFDS
   generic map (
      IBUF_DELAY_VALUE => "0", -- Specify the amount of added input delay for buffer, "0"-"16" (Spartan-3E
only)
      IFD_DELAY_VALUE => "AUTO", -- Specify the amount of added delay for input register, "AUTO", "0"-"8"
(Spartan-3E only)
      IOSTANDARD => "DEFAULT")
   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
   );

   -- End of IOBUFDS_inst instantiation
```

Verilog Instantiation Template

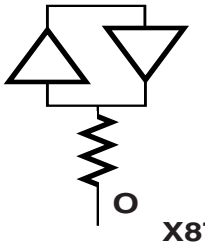
```
//     IOBUFDS   : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (IOBUFDS_inst) and/or the port declarations within the
//      code     : parenthesis maybe changed to properly reference and
//               : connect this function to the design.  Delete or comment
//               : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // IOBUFDS: Differential bidirectional Buffer
   // FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   IOBUFDS #(
     .IBUF_DELAY_VALUE("0"),   // Specify the amount of added input delay for the buffer, "0"-"16" (Spartan-
3E only)
      .IFD_DELAY_VALUE("AUTO"), // Specify the amount of added delay for input register, "AUTO", "0"-"8"
(Spartan-3E only)
      .IOSTANDARD("DEFAULT")    // 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
   );

   // End of IOBUFDS_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

KEEPER

Primitive: KEEPER Symbol



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

Usage

This design element is instantiated rather than inferred.

VHDL Instantiation Template

```
--   KEEPER    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (KEEPER_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- KEEPER: I/O Buffer Weak Keeper
   --         All FPGA
   -- Xilinx  HDL Libraries Guide version 8.1i

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

   -- End of KEEPER_inst instantiation
```

Verilog Instantiation Template

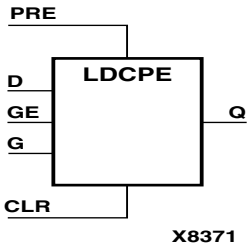
```
//     KEEPER    : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (KEEPER_inst) and/or the port declarations within the
//      code     : parenthesis maybe changed to properly reference and
//               : connect this function to the design.  Delete or comment
//               : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // KEEPER: I/O Buffer Weak Keeper
   //         All FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

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

   // End of KEEPER_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

LDCPE

Primitive: Transparent Data Latch with Asynchronous Clear and Preset and Gate Enable



LDCPE is a transparent data latch with data (D), asynchronous clear (CLR), asynchronous preset (PRE), and gate enable (GE). When CLR is High, it overrides the other inputs and resets the data (Q) output Low. When PRE is High and CLR is Low, it presets the data (Q) output High. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High and CLR and PRE are Low. 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 cleared, output Low, when power is applied, or when global reset is active.

For Spartan-3E 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 Spartan-3E symbol.

Inputs					Outputs
CLR	PRE	GE	G	D	Q
1	X	X	X	X	0
0	1	X	X	X	1
0	0	0	X	X	No Change
0	0	1	1	0	0
0	0	1	1	1	1
0	0	1	0	X	No Change
0	0	1	↓	D	D

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	1-Bit	1 or 0	0	Sets the initial value of Q output after configuration

VHDL Instantiation Template

```
-- LDCPE: Transparent latch with with Asynchronous Reset, Preset and
   --        Gate Enable.  All families.
   -- Xilinx HDL Libraries Guide version 8.1i
```

```
   LDCPE_inst : LDCPE
   generic map (
      INIT => '0') -- Initial value of the latch
   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 LDCPE_inst instantiation
```

Verilog Instantiation Template

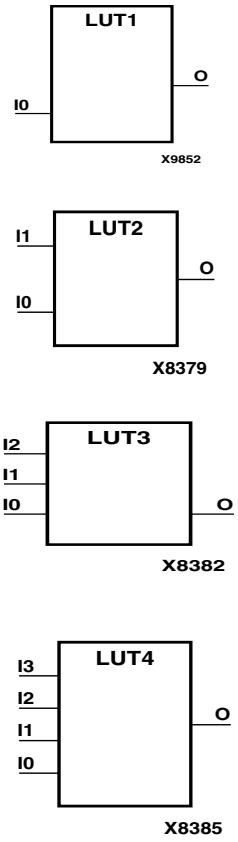
```
//     LDCPE    : In order to incorporate this function into the design,
//    Verilog   : the following instance declaration needs to be placed
//   instance   : in the body of the design code.  The instance name
//  declaration : (LDCPE_inst) and/or the port declarations within the
//     code     : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  Delete or comment
//              : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // LDCPE: Transparent latch with with Asynchronous Reset, Preset and
   //        Gate Enable.  All families.
   // Xilinx HDL Libraries Guide Version 8.1i

   LDCPE #(
      .INIT(1'b0) // Initial value of latch (1'b0 or 1'b1)
   ) LDCPE_inst (
      .Q(Q),      // Data output
      .CLR(CLR),  // Asynchronous clear/reset input
      .D(D),      // Data input
      .G(G),      // Gate input
      .GE(GE),    // Gate enable input
      .PRE(PRE)   // Asynchronous preset/set input
   );

   // End of LDCPE_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

LUT1, 2, 3, 4

Primitive: 1-, 2-, 3-, 4-Bit Look-Up-Table with General Output



LUT1, LUT2, LUT3, and LUT4 are, respectively, 1-, 2-, 3-, and 4-bit look-up-tables (LUTs) 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.

LUT1 provides a look-up-table version of a buffer or inverter.

LUTs are the basic Spartan-3E building blocks. Two LUTs are available in each CLB slice; four LUTs are available in each CLB. The variants, "LUT1_D, LUT2_D, LUT3_D, LUT4_D"and "LUT1_L, LUT2_L, LUT3_L, LUT4_L"provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

LUT3 Function Table

Inputs			Outputs
I2	I1	I0	O
0	0	0	INIT[0]
0	0	1	INIT[1]
0	1	0	INIT[2]
0	1	1	INIT[3]
1	0	0	INIT[4]
1	0	1	INIT[5]
1	1	0	INIT[6]
1	1	1	INIT[7]

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

Usage

LUTs are inferred with the logic portions of the HDL code. Xilinx suggests that you instantiate LUTs only if you have a need to implicitly specify the logic mapping, or if you need to manually place or relationally place the logic.

Available Attributes

LUT1

Attribute	Type	Allowed Values	Default	Description
INIT	2-Bit Hexadecimal	2-Bit Hexadecimal	2'h0	Initializes ROMs, RAMs, registers, and look-up tables.

LUT2

Attribute	Type	Allowed Values	Default	Description
INIT	4-Bit Hexadecimal	4-Bit Hexadecimal	4'h0	Initializes ROMs, RAMs, registers, and look-up tables.

LUT3

Attribute	Type	Allowed Values	Default	Description
INIT	8-Bit Hexadecimal	8-Bit Hexadecimal	8'h00	Initializes ROMs, RAMs, registers, and look-up tables.

LUT4

Attribute	Type	Allowed Values	Default	Description
INIT	16-Bit Hexadecimal	16-Bit Hexadecimal	16'h0000	Initializes ROMs, RAMs, registers, and look-up tables.

VHDL Instantiation Template for LUT1

```
--    LUT1     : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT1_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT1: 1-input Look-Up Table with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 For LUT1

```
//     LUT1    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.
```

```
//  <-----Cut code below this line---->

   // LUT1: 1-input Look-Up Table with general output
   //       For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   LUT1 #(
      .INIT(2'b00)  // Specify LUT Contents
   ) LUT1_inst (
      .O(O),   // LUT general output
      .I0(I0)  // LUT input
   );

   // End of LUT1_inst instantiation
```

VHDL Instantiation Template for LUT2

```
--    LUT2     : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT2_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT2: 2-input Look-Up Table with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 for LUT2

```
//     LUT2    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT2_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT2: 2-input Look-Up Table with general output
   //       For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

VHDL Instantiation Template for LUT3

```
--    LUT3     : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT3_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT3: 3-input Look-Up Table with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 For LUT3

```
//     LUT3    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT3_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT3: 3-input Look-Up Table with general output
   //       For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

VHDL Instantiation Template for LUT4

```
--    LUT4     : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT4_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT4: 4-input Look-Up Table with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 For LUT4

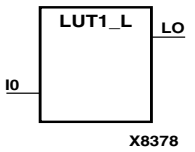
```
//     LUT4    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT4_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT4: 4-input Look-Up Table with general output
   //       For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

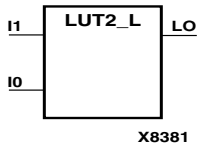
For More Information

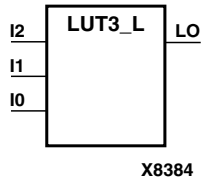
Consult the Spartan-3E Data Sheet.

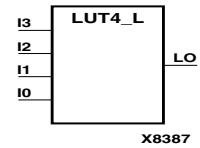
LUT1_L, LUT2_L, LUT3_L, LUT4_L

Primitive: 1-, 2-, 3-, 4-Bit Look-Up-Table with Local Output









LUT1_L, LUT2_L, LUT3_L, and LUT4_L are, respectively, 1-, 2-, 3-, and 4- bit look-up-tables (LUTs) with a local output (LO) that connects 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.

LUT1_L provides a look-up-table version of a buffer or inverter.

See also "LUT1, 2, 3, 4"and "LUT1_D, LUT2_D, LUT3_D, LUT4_D"

LUT3_L Function Table

Inputs			Outputs
I2	I1	I0	LO
0	0	0	INIT[0]
0	0	1	INIT[1]
0	1	0	INIT[2]
0	1	1	INIT[3]
1	0	0	INIT[4]
1	0	1	INIT[5]
1	1	0	INIT[6]
1	1	1	INIT[7]

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

Usage

LUTs are inferred with the logic portions of the HDL code. Xilinx suggests that you instantiate LUTs only if you have a need to implicitly specify the logic mapping, or if you need to manually place or relationally place the logic.

Available Attributes

LUT1_L

Attribute	Type	Allowed Values	Default	Description
INIT	2-Bit Hexadecimal	2-Bit Hexadecimal	2'h0	Initializes ROMs, RAMs, registers, and look-up tables.

LUT2_L

Attribute	Type	Allowed Values	Default	Description
INIT	4-Bit Hexadecimal	4-Bit Hexadecimal	4'h0	Initializes ROMs, RAMs, registers, and look-up tables.

LUT3_L

Attribute	Type	Allowed Values	Default	Description
INIT	8-Bit Hexadecimal	8-Bit Hexadecimal	8'h00	Initializes ROMs, RAMs, registers, and look-up tables.

LUT4_L

Attribute	Type	Allowed Values	Default	Description
INIT	16-Bit Hexadecimal	16-Bit Hexadecimal	16'h0000	Initializes ROMs, RAMs, registers, and look-up tables.

VHDL Instantiation Template for LUT1_L

```
--   LUT1_L     : In order to incorporate this function into the design,
--    VHDL      : the following instance declaration needs to be placed
--  instance    : in the architecture body of the design code.  The
-- declaration  : instance name (LUT1_L_inst) and/or the port declarations
--    code      : after the "=>" assignment maybe changed to properly
--              : reference and connect this function to the design.
--              : All inputs and outputs must be connected.

--   Library    : In addition to adding the instance declaration, a use
-- declaration  : statement for the UNISIM.vcomponents library needs to be
--     for      : added before the entity declaration.  This library
--   Xilinx     : contains the component declarations for all Xilinx
--  primitives  : primitives and points to the models that are used
--              : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT1_L: 1-input Look-Up Table with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 For LUT1_L

```
//   LUT1_L     : In order to incorporate this function into the design,
//   Verilog    : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration  : (LUT1_L_inst) and/or the port declarations within the
//    code      : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  All inputs
//              : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT1_L: 1-input Look-Up Table with local output
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

VHDL Instantiation Template for LUT2_L

```
--   LUT2_L     : In order to incorporate this function into the design,
--    VHDL      : the following instance declaration needs to be placed
--  instance    : in the architecture body of the design code.  The
-- declaration : instance name (LUT2_L_inst) and/or the port declarations
--    code      : after the "=>" assignment maybe changed to properly
--              : reference and connect this function to the design.
--              : All inputs and outputs must be connected.

--   Library    : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for      : added before the entity declaration.  This library
--   Xilinx     : contains the component declarations for all Xilinx
--  primitives  : primitives and points to the models that are used
--              : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT2_L: 2-input Look-Up Table with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 For LUT2_L

```
//   LUT2_L     : In order to incorporate this function into the design,
//   Verilog    : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration : (LUT2_L_inst) and/or the port declarations within the
//    code      : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  All inputs
//              : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT2_L: 2-input Look-Up Table with local output
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

VHDL Instantiation Template for LUT3_L

```
--   LUT3_L    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT3_L_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT3_L: 3-input Look-Up Table with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 For LUT3_L

```
//   LUT3_L    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT3_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT3_L: 3-input Look-Up Table with local output
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

VHDL Instantiation Template for LUT4_L

```
--   LUT4_L    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT4_L_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
```

```
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT4_L: 4-input Look-Up Table with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 For LUT4_L

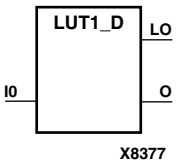
```
//   LUT4_L    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT4_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT4_L: 4-input Look-Up Table with local output
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

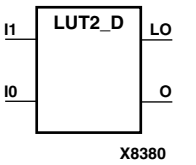
For More Information

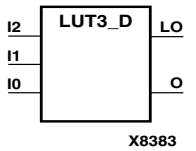
Consult the Spartan-3E Data Sheet.

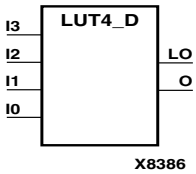
LUT1_D, LUT2_D, LUT3_D, LUT4_D

Primitive: 1-, 2-, 3-, 4-Bit Look-Up-Table with Dual Output









LUT1_D, LUT2_D, LUT3_D, and LUT4_D are, respectively, 1-, 2-, 3-, and 4-bit look-up-tables (LUTs) with two functionally identical outputs, O and LO. The O output is a general interconnect. The LO output is connects 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.

LUT1_D provides a look-up-table version of a buffer or inverter.

See also "LUT1, 2, 3, 4"and"LUT1_L, LUT2_L, LUT3_L, LUT4_L"

LUT3_D Function Table

Inputs			Outputs	
I2	I1	I0	O	LO
0	0	0	INIT[0]	INIT[0]
0	0	1	INIT[1]	INIT[1]
0	1	0	INIT[2]	INIT[2]
0	1	1	INIT[3]	INIT[3]
1	0	0	INIT[4]	INIT[4]
1	0	1	INIT[5]	INIT[5]
1	1	0	INIT[6]	INIT[6]
1	1	1	INIT[7]	INIT[7]

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

Usage

LUTs are inferred with the logic portions of the HDL code. Xilinx suggests that you instantiate LUTs only if you have a need to implicitly specify the logic mapping, or if you need to manually place or relationally place the logic.

Available Attributes

LUT1_D

Attribute	Type	Allowed Values	Default	Description
INIT	2-Bit Hexadecimal	2-Bit Hexadecimal	2'h0	Initializes ROMs, RAMs, registers, and look-up tables.

LUT2_D

Attribute	Type	Allowed Values	Default	Description
INIT	4-Bit Hexadecimal	4-Bit Hexadecimal	4'h0	Initializes ROMs, RAMs, registers, and look-up tables.

LUT3_D

Attribute	Type	Allowed Values	Default	Description
INIT	8-Bit Hexadecimal	8-Bit Hexadecimal	8'h00	Initializes ROMs, RAMs, registers, and look-up tables.

LUT4_D

Attribute	Type	Allowed Values	Default	Description
INIT	16-Bit Hexadecimal	16-Bit Hexadecimal	16'h0000	Initializes ROMs, RAMs, registers, and look-up tables.

VHDL Instantiation Template for LUT1_D

```
--   LUT1_D    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT1_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT1_D: 1-input Look-Up Table with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   LUT1_D_inst : LUT1_D
   generic map (
      INIT => "00")
   port map (
      LO => LO, -- LUT local output
      O => O,   -- LUT general output
      I0 => I0  -- LUT input
   );

   -- End of LUT1_D_inst instantiation
```

Verilog Instantiation Template For LUT1_D

```
//   LUT1_D    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT1_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT1_D: 1-input Look-Up Table with general and local outputs
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

VHDL Instantiation Template for LUT2_D

```
--   LUT2_D    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT2_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT2_D: 2-input Look-Up Table with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   LUT2_D_inst : LUT2_D
   generic map (
      INIT => X"0")
   port map (
      LO => LO, -- LUT local output
      O => O,   -- LUT general output
      I0 => I0, -- LUT input
      I1 => I1  -- LUT input
   );

   -- End of LUT2_D_inst instantiation
```

Verilog Instantiation Template For LUT2_D

```
//   LUT2_D    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT2_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT2_D: 2-input Look-Up Table with general and local outputs
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   LUT2_D #(
      .INIT(4'h0)  // Specify LUT Contents
   ) LUT2_D_inst (
      .LO(LO), // LUT local output
      .O(O),   // LUT general output
      .I0(I0), // LUT input
      .I1(I1)  // LUT input
   );

   // End of LUT2_L_inst instantiation
```

VHDL Instantiation Template for LUT3_D

```
--   LUT3_D    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT3_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT3_D: 3-input Look-Up Table with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   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
```

Verillog Instantiation Template for LUT3_D

```
//   LUT3_D    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT3_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT3_D: 3-input Look-Up Table with general and local outputs
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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
```

VHDL Instantiation Template for LUT4_D

```
--   LUT4_D    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (LUT4_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
```

```
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- LUT4_D: 4-input Look-Up Table with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   LUT4_D_inst : LUT4_D
   generic map (
      INIT => X"0000")
   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
   );

   -- End of LUT4_D_inst instantiation
```

Verilog Instantiation Template For LUT4_D

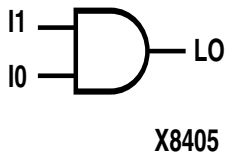
```
//   LUT4_D    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT4_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // LUT4_D: 4-input Look-Up Table with general and local outputs
   //         For use with all FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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

Consult the Spartan-3E Data Sheet.

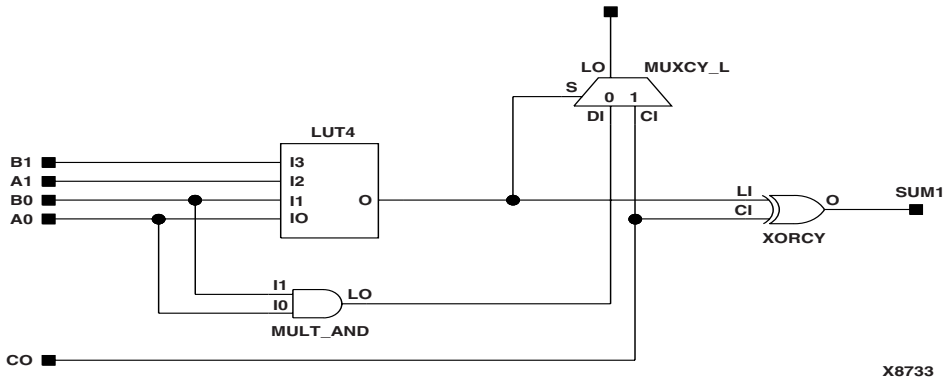
MULT_AND

Primitive: Fast Multiplier AND



MULT_AND is a logical AND gate component that can be used to reduce logic and improve speed when the user is building soft multipliers within the device fabric. It can also be used in some carry-chain operations to reduce the needed LUTs to implement some functions. The I1 and I0 inputs *must* betconnected to the I1 and I0 inputs of the associated LUT. The LO output *must* be connected to the DI input of the associated MUXCY, MUXCY_D, or MUXCY_L.

Inputs		Output
I1	I0	LO
0	0	0
0	1	0
1	0	0
1	1	1



Example Multiplier Using MULT_AND

Usage

This design element can be instantiated and inferred.

VHDL Instantiation Template

```
--  MULT_AND   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MULT_AND_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.
```

```
--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MULT_AND: 2-input AND gate connected to Carry chain
   --           All FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   MULT_AND_inst : MULT_AND
   port map (
      LO => LO,    -- MULT_AND output (connect to MUXCY DI)
      I0 => I0,    -- MULT_AND data[0] input
      I1 => I1     -- MULT_AND data[1] input
   );

   -- End of MULT_AND_inst instantiation
```

Verilog Instantiation Template

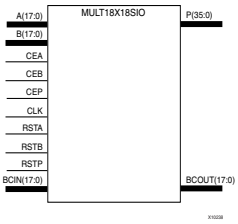
```
//  MULT_AND   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MULT_AND_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MULT_AND: 2-input AND gate connected to Carry chain
   //           For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MULT_AND MULT_AND_inst (
      .LO(LO),    // MULT_AND output (connect to MUXCY DI)
      .I0(I0),    // MULT_AND data[0] input
      .I1(I1)     // MULT_AND data[1] input
   );

   // End of MULT_AND_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MULT18X18SIO

Primitive: 18x18 Cascadable Signed Multiplier with Optional Input and Output registers, Clock Enable, and Synchronous Reset



The MULT18X18SIO is a 36-bit output, 18x18-bit input dedicated signed multiplier. This component can perform asynchronous multiplication operations when the attributes AREG, BREG and PREG are all set to 0. Alternatively, synchronous multiplication operations of different latency and performance characteristics can be performed when any combination of those attributes is set to 1. When using the multiplier in synchronous operation, the MULT18X18SIO features active high clock enables for each set of register banks in the multiplier, CEA, CEB and CEP, as well as synchronous resets, RSTA, RSTB, and RSTP. Multiple MULT18X18SIOs can be cascaded to create larger multiplication functions using the BCIN and BCOUT ports in combination with the B_INPUT attribute.

Usage

The MULT18X18SIO can be inferred by most synthesis tools using standard VHDL or Verilog notation for multiplication. Alternatively, Core Generator™ System and other IP can also create multiplication functions using this component. If preferred, the MULT18X18SIO can be instantiated into the VHDL or Verilog code to give full control over the implementation of the component. To change the default behavior of the MULT18X18SIO, attributes can be modified via the generic map (VHDL) or named parameter value assignment (Verilog) as a part of the instantiated component.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
AREG	Integer	1 or 0	1	Enable the input registers on the A port (1=on, 0=off).
B_INPUT	String	"DIRECT" or "CASCADE"	"DIRECT"	B input from B(17:0) (DIRECT) or from BCIN (17:0) (CASCADE).
BREG	Integer	1 or 0	1	Enable the input registers on the B port (1=on, 0=off).
PREG	Integer	1 or 0	1	Enable the output registers on the P port (1=on, 0=off).

VHDL Instantiation Template

```
--  MULT18X18SIO : In order to incorporate this function into the design,
--    VHDL       : the following instance declaration needs to be placed
--  instance     : in the architecture body of the design code.  The
-- declaration   : instance name (MULT18X18SIO_inst) and/or the port declarations
--    code       : after the "=>" assignment maybe changed to properly
--               : reference and connect this function to the design.
--               : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
```

```
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MULT18X18SIO: 18 x 18 cascadable, signed synchronous/asynchronous multiplier
   --               Spartan-3E
   -- Xilinx  HDL Libraries Guide version 8.1i

   MULT18X18SIO_inst : MULT18X18SIO
   generic map (
      AREG => 1, -- Enable the input registers on the A port (1=on, 0=off)
      BREG => 1, -- Enable the input registers on the B port (1=on, 0=off)
      B_INPUT => "DIRECT", -- B cascade input "DIRECT" or "CASCADE"
      PREG => 1) -- Enable the input registers on the P port (1=on, 0=off)
   port map (
      BCOUT => BCOUT, -- 18-bit cascade output
      P => P,     -- 36-bit multiplier output
      A => A,     -- 18-bit multiplier input
      B => B,     -- 18-bit multiplier input
      BCIN => BCIN, -- 18-bit cascade input
      CEA => CEA, -- Clock enable input for the A port
      CEB => CEB, -- Clock enable input for the B port
      CEP => CEP, -- Clock enable input for the P port
      CLK => CLK,     -- Clock input
      RSTA => RSTA, -- Synchronous reset input for the A port
      RSTB => RSTB, -- Synchronous reset input for the B port
      RSTP => RSTP, -- Synchronous reset input for the P port
   );

   -- End of MULT18X18SIO_inst instantiation
```

Verilog Instantiation Template

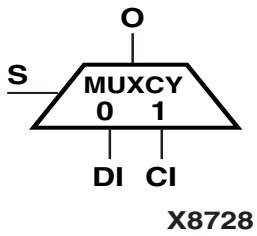
```
//  MULT18X18SIO : In order to incorporate this function into the design,
//    Verilog    : the following instance declaration needs to be placed
//   instance    : in the body of the design code.  The instance name
//  declaration  : (MULT18X18SIO_inst) and/or the port declarations within the
//     code      : parenthesis maybe changed to properly reference and
//               : connect this function to the design.  All inputs
//               : and outputs must be connected.

//  <-----Cut code below this line---->

   // MULT18X18SIO: 18 x 18 cascadable, signed synchronous/asynchronous multiplier
   //               Spartan-3E
   // Xilinx HDL Libraries Guide Version 8.1i

   MULT18X18SIO #(
      .AREG(1), // Enable the input registers on the A port (1=on, 0=off)
      .BREG(1), // Enable the input registers on the B port (1=on, 0=off)
      .B_INPUT("DIRECT"), // B cascade input "DIRECT" or "CASCADE"
      .PREG(1)  // Enable the input registers on the P port (1=on, 0=off)
   ) MULT18X18SIO_inst (
      .BCOUT(BCOUT), // 18-bit cascade output
      .P(P),     // 36-bit multiplier output
      .A(A),     // 18-bit multiplier input
      .B(B),     // 18-bit multiplier input
      .BCIN(BCIN), // 18-bit cascade input
      .CEA(CEA), // Clock enable input for the A port
      .CEB(CEB), // Clock enable input for the B port
      .CEP(CEP), // Clock enable input for the P port
      .CLK(CLK), // Clock input
      .RSTA(RSTA), // Synchronous reset input for the A port
      .RSTB(RSTB), // Synchronous reset input for the B port
      .RSTP(RSTP)  // Synchronous reset input for the P port
   );

   // End of MULT18X18SIO_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXCY

Primitive: 2-to-1 Multiplexer for Carry Logic with General Output



MUXCY implements a 1-bit high-speed carry propagate function. One such function can be implemented per logic cell (LC), for a total of 8 bits per configurable logic block (CLB) for Spartan-3E.

The direct input (DI) of a slice is connected to the DI input of the MUXCY. The carry in (CI) input of an LC is connected to the CI input of the MUXCY. The select input (S) of the MUXCY is driven by the output of the lookup table (LUT) and configured as a MUX function. The carry out (O) of the MUXCY reflects the state of the selected input and implements the carry out function of each LC. When Low, S selects DI; when set to High, S selects CI.

The variants, "MUXCY_D"and "MUXCY_L"provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

Inputs			Outputs
S	DI	CI	O
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Usage

This design element can be instantiated and inferred.

VHDL Instantiation Template

```
--    MUXCY    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXCY_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXCY: Carry-Chain MUX with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXCY_inst : MUXCY
   port map (
      O => O,    -- Carry output signal
```

```
      CI => CI, -- Carry input signal
      DI => DI, -- Data input signal
      S => S    -- MUX select, tie to '1' or LUT4 out
   );

   -- End of MUXCY_inst instantiation
```

Verilog Instantiation Template

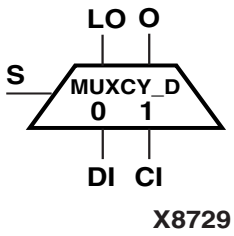
```
//    MUXCY    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXCY_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design. All inputs and
//             : and outputs of this primtive should be connected.

//  <-----Cut code below this line---->

   // MUXCY: Carry-Chain MUX with general output
   //        For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXCY MUXCY_inst (
      .O(O),   // Carry output signal
      .CI(CI), // Carry input signal
      .DI(DI), // Data input signal
      .S(S)    // MUX select, tie to '1' or LUT4 out
   );

   // End of MUXCY_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXCY_D

Primitive: 2-to-1 Multiplexer for Carry Logic with Dual Output



MUXCY_D implements a 1-bit high-speed carry propagate function. One such function can be implemented per logic cell (LC), for a total of 4 bits per configurable logic block (CLB). The direct input (DI) of an LC is connected to the DI input of the MUXCY_D. The carry in (CI) input of an LC is connected to the CI input of the MUXCY_D. The select input (S) of the MUX is driven by the output of the lookup table (LUT) and configured as an XOR function. The carry out (O and LO) of the MUXCY_D reflects the state of the selected input and implements the carry out function of each LC. When Low, S selects DI; when High, S selects CI.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO outputs connect to other inputs within the same slice.

See also "MUXCY" and "MUXCY_L"

Inputs			Outputs	
S	DI	CI	O	LO
0	1	X	1	1
0	0	X	0	0
1	X	1	1	1
1	X	0	0	0

Usage

This design element can only be instantiated. Synthesis tools use the MUXCY primitive, then MAP uses the MUXCY_D.

VHDL Instantiation Template

```
--   MUXCY_D   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXCY_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXCY_D: Carry-Chain MUX with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXCY_D_inst : MUXCY_D
   port map (
      LO => LO, -- Carry local output signal
```

```
      O => O,   -- Carry general output signal
      CI => CI, -- Carry input signal
      DI => DI, -- Data input signal
      S => S    -- MUX select, tie to '1' or LUT4 out
   );

   -- End of MUXCY_D_inst instantiation
```

Verilog Instantiation Template

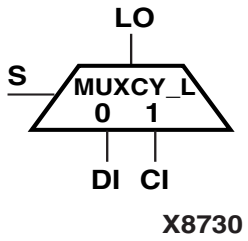
```
//   MUXCY_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXCY_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXCY_D: Carry-Chain MUX with general and local outputs
   //          For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXCY_D MUXCY_D_inst (
      .LO(LO), // Carry local output signal
      .O(O),   // Carry general output signal
      .CI(CI), // Carry input signal
      .DI(DI), // Data input signal
      .S(S)    // MUX select, tie to '1' or LUT4 out
   );

   // End of MUXCY_D_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXCY_L

Primitive: 2-to-1 Multiplexer for Carry Logic with Local Output



MUXCY_L implements a 1-bit high-speed carry propagate function. One such function can be implemented per slice, for a total of 4 bits per configurable logic block (CLB). The direct input (DI) of an LC is connected to the DI input of the MUXCY_L. The carry in (CI) input of an LC is connected to the CI input of the MUXCY_L. The select input (S) of the MUXCY_L is driven by the output of the lookup table (LUT) and configured as an XOR function. The carry out (LO) of the MUXCY_L reflects the state of the selected input and implements the carry out function of each slice. When Low, S selects DI; when High, S selects CI.

See also "MUXCY"and "MUXCY_D"

Inputs			Outputs
S	DI	CI	LO
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Usage

This design element can only be instantiated. Synthesis tools use the MUXCY primitive, then MAP uses the MUXCY_L.

VHDL Instantiation Template

```
--   MUXCY_L   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--   instance  : in the architecture body of the design code.  The
-- declaration : instance name (MUXCY_L_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXCY_L: Carry-Chain MUX with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXCY_L_inst : MUXCY_L
   port map (
      LO => LO, -- Carry local output signal
      CI => CI, -- Carry input signal
      DI => DI, -- Data input signal
      S => S    -- MUX select, tie to '1' or LUT4 out
   );
```

```
   -- End of MUXCY_L_inst instantiation
```

Verilog Instantiation Template

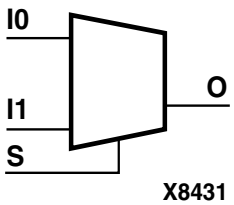
```
//   MUXCY_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXCY_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXCY_L: Carry-Chain MUX with local output
   //         For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXCY_L MUXCY_L_inst (
      .LO(LO), // Carry local output signal
      .CI(CI), // Carry input signal
      .DI(DI), // Data input signal
      .S(S)    // MUX select, tie to '1' or LUT4 out
   );

   // End of MUXCY_L_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF5

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



MUXF5 provides a multiplexer function in a CLB slice for creating a function-of-5 lookup table or a 4-to-1 multiplexer in combination with the associated lookup tables. The local outputs (LO) from the two lookup tables are connected to the I0 and I1 inputs of the MUXF5. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The variants, "MUXF5_D"and "MUXF5_L", provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

Inputs			Outputs
S	**I0**	**I1**	**O**
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Usage

This design element can be instantiated and inferred.

VHDL Instantiation Template

```
--    MUXF5    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF5_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF5: Slice MUX to tie two LUT4's together with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF5_inst : MUXF5
   port map (
      O => O,     -- Output of MUX to general routing
      I0 => I0,   -- Input (tie directly to the output of LUT4)
      I1 => I1,   -- Input (tie directoy to the output of LUT4)
      S => S      -- Input select to MUX
   );

   -- End of MUXF5_inst instantiation
```

Verilog Instantiation Template

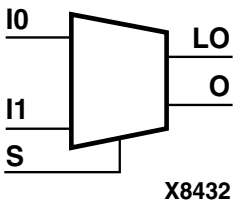
```
//    MUXF5    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF5_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF5: Slice MUX to tie two LUT4's together with general output
   //        For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF5 MUXF5_inst (
      .O(O),    // Output of MUX to general routing
      .I0(I0),  // Input (tie directly to the output of LUT4)
      .I1(I1),  // Input (tie directoy to the output of LUT4)
      .S(S)     // Input select to MUX
   );

   // End of MUXF5_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF5_D

Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output



MUXF5_D provides a multiplexer function in a CLB slice for creating a function-of-5 lookup table or a 4-to-1 multiplexer in combination with the associated lookup tables. The local outputs (LO) from the two lookup tables are connected to the I0 and I1 inputs of the MUXF5. 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 is used to connect to other inputs within the same CLB slice.

See also "MUXF5"and "MUXF5_L"

Inputs			Outputs	
S	I0	I1	O	LO
0	1	X	1	1
0	0	X	0	0
1	X	1	1	1
1	X	0	0	0

Usage

This design element can only be instantiated. Synthesis tools use the MUXF5, then MAP uses the MUXF5_D.

VHDL Instantiation Template

```
--   MUXF5_D   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--   instance  : in the architecture body of the design code.  The
-- declaration : instance name (MUXF5_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF5_D: Slice MUX to tie two LUT4's together with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF5_D_inst : MUXF5_D
   port map (
      LO => LO,  -- Ouptut of MUX to local routing
      O => O,    -- Output of MUX to general routing
      I0 => I0,  -- Input (tie directly to the output of LUT4)
      I1 => I1,  -- Input (tie directoy to the output of LUT4)
      S => S     -- Input select to MUX
   );
```

```
   -- End of MUXF5_D_inst instantiation
```

Verilog Instantiation Template

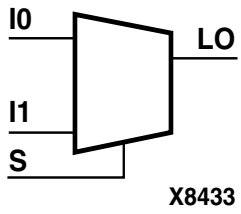
```
//   MUXF5_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF5_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF5_D: Slice MUX to tie two LUT4's together with general and local outputs
   //          For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF5_D MUXF5_D_inst (
      .LO(LO),  // Ouptut of MUX to local routing
      .O(O),    // Output of MUX to general routing
      .I0(I0),  // Input (tie directly to the output of LUT4)
      .I1(I1),  // Input (tie directoy to the output of LUT4)
      .S(S)     // Input select to MUX
   );

   // End of MUXF5_D_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF5_L

Primitive: 2-to-1 Look-Up Table Multiplexer with Local Output



MUXF5_L provides a multiplexer function in a CLB slice for creating a function-of-5 lookup table or a 4-to-1 multiplexer in combination with the associated lookup tables. The local outputs (LO) from the two lookup tables are connected to the I0 and I1 inputs of the MUXF5. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The LO output is used to connect to other inputs within the same CLB slice.

See also "MUXF5"and "MUXF5_D".

Inputs			Output
S	I0	I1	LO
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Usage

This design element can only be instantiated. Synthesis tools use the MUXF5 primitive, then MAP uses the MUXF5_L.

VHDL Instantiation Template

```
--   MUXF5_L   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF5_L_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF5_L: Slice MUX to tie two LUT4's together with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF5_L_inst : MUXF5_L
   port map (
      LO => LO,   -- Output of MUX to local routing
      I0 => I0,   -- Input (tie directly to the output of LUT4)
      I1 => I1,   -- Input (tie directoy to the output of LUT4)
      S => S      -- Input select to MUX
   );

   -- End of MUXF5_L_inst instantiation
```

Verilog Instantiation Template

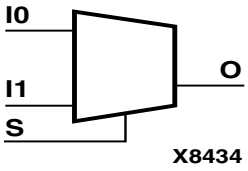
```
//   MUXF5_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF5_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF5_L: Slice MUX to tie two LUT4's together with local output
   //          For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF5_L MUXF5_L_inst (
      .LO(LO),  // Output of MUX to local routing
      .I0(I0),  // Input (tie directly to the output of LUT4)
      .I1(I1),  // Input (tie directoy to the output of LUT4)
      .S(S)     // Input select to MUX
   );

   // End of MUXF5_L_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF6

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



MUXF6 provides a multiplexer function in one half of a Spartan-3E CLB (two slices) for creating a function-of-6 lookup table or an 8-to-1 multiplexer in combination with the associated four lookup tables and two MUXF5s. The local outputs (LO) from the two MUXF5s in the CLB are connected to the I0 and I1 inputs of the MUXF6. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The variants, "MUXF6_D" and "MUXF6_L", provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

Inputs			Outputs
S	**I0**	**I1**	**O**
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--    MUXF6    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF6_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF6: CLB MUX to tie two MUXF5's together with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF6_inst : MUXF6
   port map (
      O => O,     -- Output of MUX to general routing
      I0 => I0,   -- Input (tie to MUXF5 LO out)
      I1 => I1,   -- Input (tie to MUXF5 LO out)
      S => S      -- Input select to MUX
   );

   -- End of MUXF6_inst instantiation
```

Verilog Instantiation Template

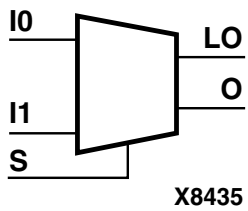
```
//    MUXF6    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF6_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF6: CLB MUX to tie two MUXF5's together with general output
   //        For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF6 MUXF6_inst (
      .O(O),    // Output of MUX to general routing
      .I0(I0),  // Input (tie to MUXF5 LO out)
      .I1(I1),  // Input (tie to MUXF5 LO out)
      .S(S)     // Input select to MUX
   );

   // End of MUXF6_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF6_D

Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output



MUXF6_D provides a multiplexer function in two slices for creating a function-of-6 lookup table or an 8-to-1 multiplexer in combination with the associated four lookup tables and two MUXF5s. The local outputs (LO) from the two MUXF5s in the CLB are connected to the I0 and I1 inputs of the MUXF6. 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 is used to connect to other inputs within the same CLB slice.

See also "MUXF6"and "MUXF6_L"

Inputs			Outputs	
S	I0	I1	O	LO
0	1	X	1	1
0	0	X	0	0
1	X	1	1	1
1	X	0	0	0

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--   MUXF6_D   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF6_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF6_D: CLB MUX to tie two MUXF5's together with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF6_D_inst : MUXF6_D
   port map (
      LO => LO,   -- Ouptut of MUX to local routing
      O => O,     -- Output of MUX to general routing
      I0 => I0,   -- Input (tie to MUXF5 LO out)
      I1 => I1,   -- Input (tie to MUXF5 LO out)
      S => S      -- Input select to MUX
   );
```

```
   -- End of MUXF6_D_inst instantiation
```

Verilog Instantiation Template

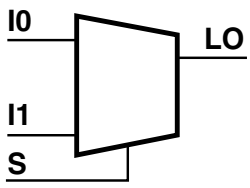
```
//   MUXF6_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF6_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF6_D: CLB MUX to tie two MUXF5's together with general and local outputs
   //          For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF6_D MUXF6_D_inst (
      .LO(LO),  // Ouptut of MUX to local routing
      .O(O),    // Output of MUX to general routing
      .I0(I0),  // Input (tie to MUXF5 LO out)
      .I1(I1),  // Input (tie to MUXF5 LO out)
      .S(S)     // Input select to MUX
   );

   // End of MUXF6_D_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF6_L

Primitive: 2-to-1 Look-Up Table Multiplexer with Local Output



MUXF6_L provides a multiplexer function in half of a Spartan-3E CLB (two slices) for creating a function-of-6 lookup table or an 8-to-1 multiplexer in combination with the associated four lookup tables and two MUXF5s. The local outputs (LO) from the two MUXF5s in the CLB are connected to the I0 and I1 inputs of the MUXF6. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The LO output is used to connect to other inputs within the same CLB slice.

See also "MUXF6"and "MUXF6_D".

Inputs			Output
S	I0	I1	LO
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--   MUXF6_L   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF6_L_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF6_L: CLB MUX to tie two MUXF5's together with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF6_L_inst : MUXF6_L
   port map (
      LO => LO,   -- Output of MUX to local routing
      I0 => I0,   -- Input (tie to MUXF5 LO out)
      I1 => I1,   -- Input (tie to MUXF5 LO out)
      S => S      -- Input select to MUX
   );

   -- End of MUXF6_L_inst instantiation
```

Verilog Instantiation Template

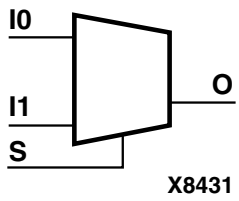
```
//   MUXF6_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF6_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF6_L: CLB MUX to tie two MUXF5's together with local output
   //          For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF6_L MUXF6_L_inst (
      .LO(LO),  // Output of MUX to local routing
      .I0(I0),  // Input (tie to MUXF5 LO out)
      .I1(I1),  // Input (tie to MUXF5 LO out)
      .S(S)     // Input select to MUX
   );

   // End of MUXF6_L_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF7

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



MUXF7 provides a multiplexer function in a full Spartan-3E CLB for creating a function-of-7 lookup table or a 16-to-1 multiplexer in combination with the associated lookup 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.

Inputs			Outputs
S	**I0**	**I1**	**O**
0	I0	X	I0
1	X	I1	I1
X	0	0	0
X	1	1	1

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--    MUXF7    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF7_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF7: CLB MUX to tie two MUXF6's together with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF7_inst : MUXF7
   port map (
      O => O,     -- Output of MUX to general routing
      I0 => I0,   -- Input (tie to MUXF6 LO out)
      I1 => I1,   -- Input (tie to MUXF6 LO out)
      S => S      -- Input select to MUX
   );

   -- End of MUXF7_inst instantiation
```

Verilog Instantiation Template

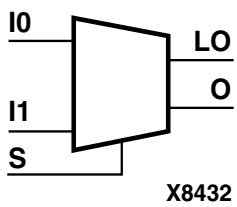
```
//    MUXF7    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF7_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF7: CLB MUX to tie two MUXF6's together with general output
   //        For use with FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF7 MUXF7_inst (
      .O(O),    // Output of MUX to general routing
      .I0(I0),  // Input (tie to MUXF6 LO out)
      .I1(I1),  // Input (tie to MUXF6 LO out)
      .S(S)     // Input select to MUX
   );

   // End of MUXF7_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF7_D

Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output



MUXF7_D provides a multiplexer function in a full Spartan-3E CLB (four slices) for creating a function-of-7 lookup table or a 16-to-1 multiplexer in combination with the associated lookup 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 within the same CLB slice.

See also "MUXF7"and "MUXF7_L".

Inputs			Outputs	
S	**I0**	**I1**	**O**	**LO**
0	I0	X	I0	I0
1	X	I1	I1	I1
X	0	0	0	0
X	1	1	1	1

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--   MUXF7_D   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF7_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF7_D: CLB MUX to tie two MUXF6's together with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF7_D_inst : MUXF7_D
   port map (
      LO => LO,   -- Ouptut of MUX to local routing
      O => O,     -- Output of MUX to general routing
      I0 => I0,   -- Input (tie to MUXF6 LO out)
      I1 => I1,   -- Input (tie to MUXF6 LO out)
      S => S      -- Input select to MUX
   );
```

```
   -- End of MUXF7_D_inst instantiation
```

Verilog Instantiation Template

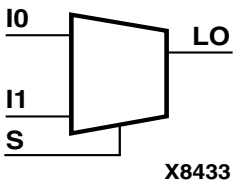
```
//   MUXF7_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF7_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF7_D: CLB MUX to tie two MUXF6's together with general and local outputs
   //          For use with FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF7_D MUXF7_D_inst (
      .LO(LO),  // Ouptut of MUX to local routing
      .O(O),    // Output of MUX to general routing
      .I0(I0),  // Input (tie to MUXF6 LO out)
      .I1(I1),  // Input (tie to MUXF6 LO out)
      .S(S)     // Input select to MUX
   );

   // End of MUXF7_D_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF7_L

Primitive: 2-to-1 Look-Up Table Multiplexer with Local Output



MUXF7_L provides a multiplexer function in a full Spartan-3E CLB (four slices) for creating a function-of-7 lookup table or a 16-to-1 multiplexer in combination with the associated lookup 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 is used to connect to other inputs within the same CLB slice.

See also "MUXF7"and "MUXF7_D".

Inputs			Output
S	**I0**	**I1**	**LO**
0	I0	X	I0
1	X	I1	I1
X	0	0	0
X	1	1	1

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--   MUXF7_L   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF7_L_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF7_L: CLB MUX to tie two MUXF6's together with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF7_L_inst : MUXF7_L
   port map (
      LO => LO,   -- Output of MUX to local routing
      I0 => I0,   -- Input (tie to MUXF6 LO out)
      I1 => I1,   -- Input (tie to MUXF6 LO out)
      S => S      -- Input select to MUX
   );

   -- End of MUXF7_L_inst instantiation
```

Verilog Instantiation Template

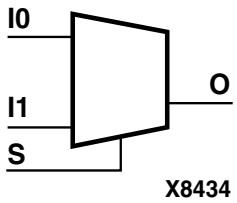
```
//   MUXF7_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF7_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF7_L: CLB MUX to tie two MUXF6's together with local output
   //          For use with FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF7_L MUXF7_L_inst (
      .LO(LO),  // Output of MUX to local routing
      .I0(I0),  // Input (tie to MUXF6 LO out)
      .I1(I1),  // Input (tie to MUXF6 LO out)
      .S(S)     // Input select to MUX
   );

   // End of MUXF7_L_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

MUXF8

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



MUXF8 provides a multiplexer function in full Spartan-3E CLBs for creating a function-of-8 lookup table or a 32-to-1 multiplexer in combination with the associated lookup 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.

See also "MUXF8_D"and "MUXF8_L".

Inputs			Outputs
S	I0	I1	O
0	I0	X	I0
1	X	I1	I1
X	0	0	0
X	1	1	1

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--    MUXF8    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF8_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF8: CLB MUX to tie two MUXF7's together with general output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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

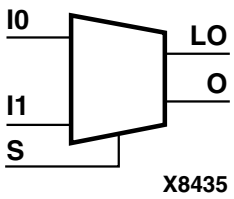
```
//    MUXF8    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF8_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF8: CLB MUX to tie two MUXF7's together with general output
   //        For use with FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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

Consult the Spartan-3E Data Sheet.

MUXF8_D

Primitive: 2-to-1 Look-Up Table Multiplexer with Dual Output



MUXF8_D provides a multiplexer function in two full Spartan-3E CLBs for creating a function-of-8 lookup table or a 32-to-1 multiplexer in combination with the associated four lookup 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 is used to connect to other inputs within the same CLB slice.

See also "MUXF8"and "MUXF8_L".

Inputs			Outputs	
S	I0	I1	O	LO
0	I0	X	I0	I0
1	X	I1	I1	I1
X	0	0	0	0
X	1	1	1	1

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--   MUXF8_D   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (MUXF8_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF8_D: CLB MUX to tie two MUXF7's together with general and local outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   MUXF8_D_inst : MUXF8_D
   port map (
      LO => LO,   -- Ouptut 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

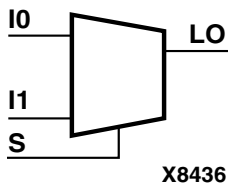
```
//   MUXF8_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF8_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF8_D: CLB MUX to tie two MUXF7's together with general and local outputs
   //          For use with FPGAa,
   // Xilinx HDL Libraries Guide Version 8.1i

   MUXF8_D MUXF8_D_inst (
      .LO(LO),  // Ouptut 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

Consult the Spartan-3E Data Sheet.

MUXF8_L

Primitive: 2-to-1 Look-Up Table Multiplexer with Local Output



MUXF8_L provides a multiplexer function in two full Spartan-3E CLBs for creating a function-of-8 lookup table or a 32-to-1 multiplexer in combination with the associated four lookup 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 within the same CLB slice.

See also "MUXF8"and "MUXF8_D".

Inputs			Output
S	**I0**	**I1**	**LO**
0	I0	X	I0
1	X	I1	I1
X	0	0	0
X	1	1	1

Usage

This design element can only be instantiated.

VHDL Instantiation Template

```
--   MUXF8_L   : In order to incorporate this function into the design,
--     VHDL     : the following instance declaration needs to be placed
--  instance    : in the architecture body of the design code.  The
-- declaration : instance name (MUXF8_L_inst) and/or the port declarations
--    code      : after the "=>" assignment maybe changed to properly
--              : reference and connect this function to the design.
--              : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for      : added before the entity declaration.  This library
--   Xilinx     : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--              : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- MUXF8_L: CLB MUX to tie two MUXF7's together with local output
   -- Xilinx HDL Libraries Guide Version 8.1i

   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

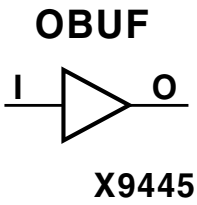
```
//   MUXF8_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF8_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // MUXF8_L: CLB MUX to tie two MUXF7's together with local output
   //          For use with FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   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

Consult the Spartan-3E Data Sheet.

OBUF

Primitive: Single-ended Output Buffers



Output buffers are necessary for all output signals because they isolate the internal circuit and provide drive current for signals leaving a chip. The OBUF is a constantly enabled output buffer that specifies a single-ended output when a 3-state is not necessary for the output. The output (O) of an OBUF should be connected directly to the top-level ouput port in the design.

Usage

OBUFs are optional for use in schematics because they are automatically inserted into a design, if necessary. To manually add this component, however, the component should be placed in the top-level schematic connecting the output directly to an output port marker.

OBUFs are available in bundles of 4, 8, or 16 to make it easier for you to incorporate them into your design without having to apply multiples of them one at a time. (The bundles are identified as OBUF4, OBUF8, and OBUF16.)

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	Integer	2, 4, 6, 8, 12, 16, 24	12	Sets the output drive in mA.
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.
SLEW	String	"SLOW", "FAST", and "QUIETIO"	"SLOW"	Sets the output rise and fall time.

VHDL Instantiation Template

```
--    OBUF     : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (OBUF_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- OBUF: Single-ended Output Buffer
   --       All devices
   -- Xilinx  HDL Libraries Guide version 8.1i
```

```
   OBUF_inst : OBUF
   generic map (
      DRIVE => 12,
      IOSTANDARD => "DEFAULT",
      SLEW => "SLOW")
   port map (
      O => O,     -- Buffer output (connect directly to top-level port)
      I => I      -- Buffer input
   );

   -- End of OBUF_inst instantiation
```

Verilog Instantiation Template

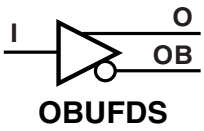
```
//      OBUF       : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (OBUF_inst) and/or the port declarations within the
//      code       : parenthesis maybe changed to properly reference and
//                 : connect this function to the design.  Delete or comment
//                 : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // OBUF: Single-ended Output Buffer
   //       All devices
   // Xilinx HDL Libraries Guide Version 8.1i

   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

Consult the *Spartan-3E Data Sheet*.

OBUFDS

Primitive: Differential Signaling Output Buffer with Selectable I/O Interface



X9259

OBUFDS is a single output buffer that supports low-voltage, differential signaling (1.8v 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).

Inputs	Outputs	
I	O	OB
0	0	1
1	1	0

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.

VHDL Instantiation Template

```
--   OBUFDS     : In order to incorporate this function into the design,
--    VHDL      : the following instance declaration needs to be placed
--  instance    : in the architecture body of the design code.  The
-- declaration  : instance name (OBUFDS_inst) and/or the port declarations
--    code      : after the "=>" assignment maybe changed to properly
--              : connect this function to the design.  Delete or comment
--              : out inputs/outs that are not necessary.

--   Library    : In addition to adding the instance declaration, a use
-- declaration  : statement for the UNISIM.vcomponents library needs to be
--     for      : added before the entity declaration.  This library
--   Xilinx     : contains the component declarations for all Xilinx
--  primitives  : primitives and points to the models that are used
--              : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- OBUFDS: Differential Output Buffer
   -- FPGAs.
   -- Xilinx  HDL Libraries Guide version 8.1i

   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

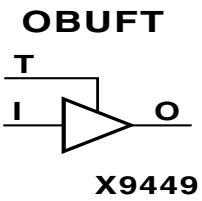
```
//     OBUFDS      : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (OBUFDS_inst) and/or the port declarations within the
//      code       : parenthesis maybe changed to properly reference and
//                 : connect this function to the design.  Delete or comment
//                 : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // OBUFDS: Differential Output Buffer
   // Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   OBUFDS #(
      .IOSTANDARD("DEFAULT") // Specify the output I/O standard
   ) OBUFDS_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
   );

   // End of OBUFDS_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

OBUFT

Primitive: 3-State Output Buffer with Active-Low Output Enable



Output buffers are necessary for all output signals because they isolate the internal circuit and provide drive current for signals leaving a chip. The OBUFT is a 3-state output buffer with input I, output O, and active-Low output enables (T). 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).

An OBUFT output should be connected directly to the top-level output or inout port. 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.

Inputs		Outputs
T	**I**	**O**
1	X	Z
0	1	1
0	0	0

Usage

OBUFTs are generally inferred by the synthesis when an output port is specified to have a high impedance, Z, as well as drive an output. It is generally suggested to infer this element however if more control of the usage of this component is necessary, it can be instantiated.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	Integer	2, 4, 6, 8, 12, 16, 24	12	Selects output drive strength (mA) for the SelectIO buffers that use the LVTTL, LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, or LVCMOS33 interface I/O standard.
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.
SLEW	String	"SLOW" , "FAST", and "QUIETIO"	"SLOW"	Sets the output rise and fall time.

VHDL Instantiation Template

```
--    OBUFT     : In order to incorporate this function into the design,
--    VHDL      : the following instance declaration needs to be placed
--  instance    : in the architecture body of the design code.  The
-- declaration : instance name (OBUFT_inst) and/or the port declarations
--    code      : after the "=>" assignment maybe changed to properly
--              : connect this function to the design.  Delete or comment
--              : out inputs/outs that are not necessary.

--   Library    : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
```

```
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- OBUFT: Single-ended 3-state Output Buffer
   --        All devices
   -- Xilinx  HDL Language Template version 7.1i

   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

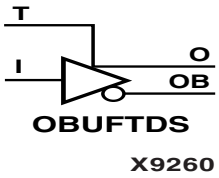
```
//      OBUFT      : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (OBUFT_inst) and/or the port declarations within the
//      code       : parenthesis maybe changed to properly reference and
//                 : connect this function to the design.  Delete or comment
//                 : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // OBUFT: Single-ended 3-state Output Buffer
   //        All devices
   // Xilinx HDL Language Template version 7.1i

   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

Consult the Spartan-3E Data Sheet.

OBUFTDS

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



OBUFTDS is a single 3-state, differential signaling output buffer with active Low enable and a Select I/O interface.

When T is Low, data on the input of the buffer is transferred to the output (O) and inverted output (OB). When T is High, both outputs are high impedance (off or Z state).

Inputs		Outputs	
I	T	O	OB
X	1	Z	Z
0	0	0	1
1	0	1	0

Usage

This design element is available for instantiation only.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	Integer	2, 4, 6, 8, 12, 16, 24	12	Selects output drive strength (mA) for the SelectIO buffers that use the LVTTL, LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, or LVCMOS33 interface I/O standard.
IOSTANDARD	String	"DEFAULT"	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.
SLEW	String	"SLOW" or "FAST"	"SLOW"	Sets the output rise and fall time.

VHDL Instantiation Template

```
-- OBUFTDS: Differential 3-state Output Buffer
   --                          All FPGA
   -- Xilinx  HDL Libraries Guide version 8.1i

   OBUFTDS_inst : OBUFTDS
   -- Edit the following generics to specify the I/O standard, drive and slew rate.
   generic map (
      DRIVE => 12,
      IOSTANDARD => "LVDS_25",
      SLEW => "SLOW")
   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

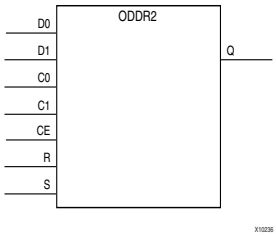
```
// OBUFTDS: Differential bidirectional Buffer
//          All FPGAs
// Xilinx HDL Libraries Guide version 8.1i OBUFTDS OBUFTDS_inst ( .O(O),
// Buffer output .IO(IO), // Diff_p inout (connect directly to
// top-level port) .IOB(IOB),
// Diff_n inout (connect directly to
// top-level port) .I(I),
// Buffer input .T(T)
// 3-state enable input );
// Edit the following defparams to specify the I/O standard, drive and
// slew rate. If the instance name is change, that change needs to be
// reflecting the this defparam. defparam OBUFTDS_inst.DRIVE = 12; defparam OBUFTDS_inst.IOSTANDARD =
"LVDS_25"; defparam OBUFTDS_inst.SLEW = "SLOW";
// End of OBUFTDS_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

ODDR2

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



The ODDR2 is an output double data rate (DDR) register useful in producing double data-rate signals exiting the FPGA. The ODDR2 requires two clocks to be connected to the component, C0 and C1, so that data is provided at the positive edge of both C0 and C1 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 yet clocked out by two clocks.

Usage

The ODDR2 must be instantiated to be incorporated into a design. To change the default behavior of the ODDR2, attributes can be modified via the generic map (VHDL) or named parameter value assignment (Verilog) as a part of the instantiated component. The ODDR2 can be either connected directly to a top-level output port in the design where an appropriate output buffer can be inferred or to an instantiated OBUF, IOBUF, OBUFDS, OBUFTDS or IOBUFDS. All inputs and outputs of this component should either be connected or properly tied off.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DDR_ ALIGNMENT	String	"NONE", "C0" or "C1"	"NONE"	Sets output alignment to "NONE", "C0" or "C1."
INIT	Integer	0 or 1	0	Sets initial state of the Q0 output to 0 or 1.
SRTYPE	String	"SYNC" or "ASYNC"	"SYNC"	Specifies "SYNC" or "ASYNC" set/reset.

VHDL Instantiation Template

```
--      ODDR2       : In order to incorporate this function into the design,
--      VHDL        : the following instance declaration needs to be placed
--    instance      : in the architecture body of the design code.  The
--   declaration    : instance name (ODDR2_inst) and/or the port declarations
--      code        : after the "=>" assignment maybe changed to properly
--                  : connect this function to the design. All inputs
--                  : and outputs must be connected.

--     Library      : In addition to adding the instance declaration, a use
--   declaration    : statement for the UNISIM.vcomponents library needs to be
--       for        : added before the entity declaration.  This library
--     Xilinx       : contains the component declarations for all Xilinx
--    primitives    : primitives and points to the models that are used
--                  : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->
```

```
   -- ODDR2: Output Double Data Rate Output Register with Set, Reset
   --       and Clock Enable. Spartan-3E
   -- Xilinx  HDL Libraries Guide version 8.1i

   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

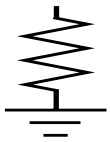
```
//      ODDR2     : In order to incorporate this function into the design,
//     Verilog    : the following instance declaration needs to be placed
//    instance    : in the body of the design code.  The instance name
//   declaration  : (ODDR2_inst) and/or the port declarations within the
//      code      : parenthesis maybe changed to properly reference and
//                : connect this function to the design.  Delete or comment
//                : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // ODDR2: Output Double Data Rate Output Register with Set, Reset
   //       and Clock Enable. Spartan-3E
   // Xilinx HDL Libraries Guide Version 8.1i

   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)
      .R(R),   // 1-bit reset input
      .S(S)    // 1-bit set input
);

   // End of ODDR2_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

PULLDOWN

Primitive: Resistor to GND



X3860

PULLDOWN resistor elements are connected to output, or bidirectional pads to guarantee a logic Low level for nodes that might Float.

Usage

This design element is instantiated rather than inferred.

VHDL Instantiation Template

```
--  PULLDOWN   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (PULLDOWN_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- PULLDOWN: I/O Buffer Weak Pull-down
   --           All FPGA
   -- Xilinx  HDL Libraries Guide version 8.1i

   PULLDOWN_inst : PULLDOWN
   port map (
      O => O     -- Pulldown output (connect directly to top-level port)
   );

   -- End of PULLDOWN_inst instantiation
```

Verilog Instantiation Template

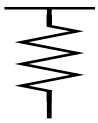
```
//    PULLDOWN   : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (PULLDOWN_inst) and/or the port declarations within the
//      code     : parenthesis maybe changed to properly reference and
//               : connect this function to the design.  Delete or comment
//               : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // PULLDOWN: I/O Buffer Weak Pull-down
   //           All FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   PULLDOWN PULLDOWN_inst (
      .O(O),     // Pulldown output (connect directly to top-level port)
   );

   // End of PULLDOWN_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

PULLUP

Primitive: Resistor to VCC, Open-Drain, and 3-State Outputs



X3861

The pull-up elements establish a High logic level for open-drain elements and macros when all the drivers are off.

Usage

This design element is instantiated rather than inferred.

VHDL Instantiation Template

```
--   PULLUP    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (PULLUP_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Delete or comment
--             : out inputs/outs that are not necessary.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- PULLUP: I/O Buffer Weak Pull-up
   --         All FPGA
   -- Xilinx  HDL Libraries Guide version 8.1i

   PULLUP_inst : PULLUP
   port map (
      O => O     -- Pullup output (connect directly to top-level port)
   );

   -- End of PULLUP_inst instantiation
```

Verilog Instantiation Template

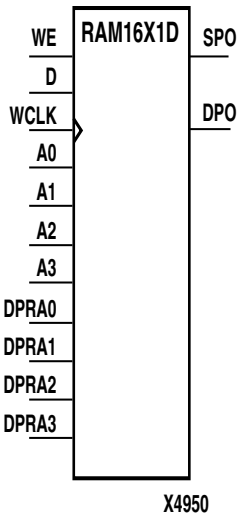
```
//     PULLUP    : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (PULLUP_inst) and/or the port declarations within the
//      code     : parenthesis maybe changed to properly reference and
//               : connect this function to the design.  Delete or comment
//               : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // PULLUP: I/O Buffer Weak Pull-up
   //         All FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   PULLUP PULLUP_inst (
      .O(O),     // Pullup output (connect directly to top-level port)
   );

   // End of PULLUP_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

RAM16X1D

Primitive: 16-Deep by 1-Wide Static Dual Port Synchronous RAM



RAM16X1D is a 16-word by 1-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA3 – DPRA0) and the write address (A3 – 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 4-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.

Mode selection is shown in the following truth table.

Inputs			Outputs	
WE (mode)	**WCLK**	**D**	**SPO**	**DPO**
0 (read)	X	X	data_a	data_d
1 (read)	0	X	data_a	data_d
1 (read)	1	X	data_a	data_d
1 (write)	↑	D	D	data_d
1 (read)	↓	X	data_a	data_d

data_a = word addressed by bits A3-A0

data_d = word addressed by bits DPRA3-DPRA0

The SPO output reflects the data in the memory cell addressed by A3 – A0. The DPO output reflects the data in the memory cell addressed by DPRA3 – DPRA0.

Note: The write process is not affected by the address on the read address port.

Specifying Initial Contents of a RAM

You can use the INIT attribute to specify an initial value directly on the symbol if the RAM is 1 bit wide and 16, 32, 64, or 128 bits deep. The value must be a hexadecimal number, for example, INIT=ABAC. If the INIT attribute is not specified, the RAM is initialized with zero.

See the "INIT" section of the *Constraints Guide* for more information on the INIT attribute.

For Spartan-3E wide RAMs (2, 4, and 8-bit wide single port synchronous RAMs with a WCLK) can also be initialized. These RAMs, however, require INIT_xx attributes.

Usage

This design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the *XST User Guide*.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	16-Bit Hexadecimal	16-Bit Hexadecimal	All zeros	Initializes ROMs, RAMs, registers, and look-up tables.

VHDL Instantiation Template

```
--   RAM16X1D  : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--   instance  : in the architecture body of the design code.  The
-- declaration : instance name (RAM16X1D_inst) and/or the port declarations
--     code    : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- RAM16X1D: 16 x 1 positive edge write, asynchronous read dual-port distributed RAM
   --           All FPGAs
   -- Xilinx  HDL Libraries Guide version 8.1i

   RAM16X1D_inst : RAM16X1D
   generic map (
      INIT => X"0000")
   port map (
      DPO => DPO,     -- Port A 1-bit data output
      SPO => SPO,     -- Port B 1-bit data output
      A0 => A0,       -- Port A address[0] input bit
      A1 => A1,       -- Port A address[1] input bit
      A2 => A2,       -- Port A address[2] input bit
      A3 => A3,       -- Port A address[3] input bit
      D => D,         -- Port A 1-bit data input
      DPRA0 => DPRA0, -- Port B address[0] input bit
      DPRA1 => DPRA1, -- Port B address[1] input bit
      DPRA2 => DPRA2, -- Port B address[2] input bit
      DPRA3 => DPRA3, -- Port B address[3] input bit
      WCLK => WCLK,   -- Port A write clock input
      WE => WE        -- Port A write enable input
   );

   -- End of RAM16X1D_inst instantiation
```

Verilog Instantiation Template

```
//   RAM16X1D  : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//   instance  : in the body of the design code.  The instance name
// declaration : (RAM16X1D_inst) and/or the port declarations within the
//     code    : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // RAM16X1D: 16 x 1 positive edge write, asynchronous read dual-port distributed RAM
   //           All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i
```

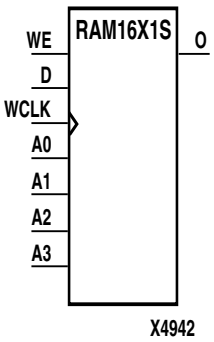
```
RAM16X1D #(
   .INIT(16'h0000) // Initial contents of RAM
) RAM16X1D_inst (
   .DPO(DPO),     // Port A 1-bit data output
   .SPO(SPO),     // Port B 1-bit data output
   .A0(A0),       // Port A address[0] input bit
   .A1(A1),       // Port A address[1] input bit
   .A2(A2),       // Port A address[2] input bit
   .A3(A3),       // Port A address[3] input bit
   .D(D),         // Port A 1-bit data input
   .DPRA0(DPRA0), // Port B address[0] input bit
   .DPRA1(DPRA1), // Port B address[1] input bit
   .DPRA2(DPRA2), // Port B address[2] input bit
   .DPRA3(DPRA3), // Port B address[3] input bit
   .WCLK(WCLK),   // Port A write clock input
   .WE(WE)        // Port A write enable input
);

// End of RAM16X1D_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

RAM16X1S

Primitive: 16-Deep by 1-Wide Static Synchronous RAM



RAM16X1S is a 16-word by 1-bit static random access memory with synchronous write capability. When the write enable (WE) 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 4-bit address (A3 – 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 RAM16X1S during configuration using the INIT attribute. See "Specifying Initial Contents of a RAM" in the RAM16X1D section.

Mode selection is shown in the following truth table.

Inputs			Outputs
WE(mode)	WCLK	D	O
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D	D
1 (read)	↓	X	Data

Data = word addressed by bits A3 – A0

Usage

This design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the *XST User Guide*.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Hexadecimal	Any 16-bit value.	All zeros	Specifies initial contents of the RAM.

VHDL Instantiation Template

```
--  RAM16X1S   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (RAM16X1S_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
```

```
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- RAM16X1S: 16 x 1 posedge write distributed  => LUT RAM
   --           All FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   RAM16X1S_inst : RAM16X1S
   generic map (
      INIT => X"0000")
   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
      D => D,        -- RAM data input
      WCLK => WCLK, -- Write clock input
      WE => WE       -- Write enable input
   );

   -- End of RAM16X1S_inst instantiation
```

Verilog Instantiation Template

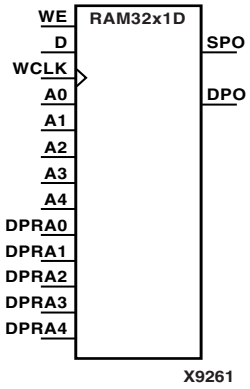
```
//  RAM16X1S   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (RAM16X1S_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // RAM16X1S: 16 x 1 posedge write distributed (LUT) RAM
   //           All FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   RAM16X1S #(
      .INIT(16'h0000) // Initial contents of RAM
   ) RAM16X1S_inst (
      .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
      .D(D),       // RAM data input
      .WCLK(WCLK), // Write clock input
      .WE(WE)      // Write enable input
   );

   // End of RAM16X1S_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

RAM32X1D

Primitive: 32-Deep by 1-Wide Static Dual Static Port Synchronous RAM



RAM32X1D 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. See "Specifying Initial Contents of a RAM" in the RAM16X1D section.

Mode selection is shown in the following truth table.

Inputs			Outputs	
WE (mode)	WCLK	D	SPO	DPO
0 (read)	X	X	data_a	data_d
1 (read)	0	X	data_a	data_d
1 (read)	1	X	data_a	data_d
1 (write)	↑	D	D	data_d
1 (read)	↓	X	data_a	data_d

data_a = word addressed by bits A4-A0

data_d = word addressed by bits DPRA4-DPRA0

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.

Note: The write process is not affected by the address on the read address port.

Usage

This design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the *XST User Guide*.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	32-Bit Hexadecimal	32-Bit Hexadecimal	All zeros	Initializes ROMs, RAMs, registers, and look-up tables.

VHDL Instantiation Template

```
--   RAM32X1D  : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--   instance  : in the architecture body of the design code.  The
-- declaration : instance name (RAM32X1D_inst) and/or the port declarations
--     code    : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- RAM32X1D: 32 x 1 positive edge write, asynchronous read dual-port distributed RAM
   --           Viretx-II/II-Pro
   -- Xilinx  HDL Libraries Guide version 8.1i

   RAM32X1D_inst : RAM32X1D
   generic map (
      INIT => X"00000000")
   port map (
      DPO => DPO,     -- Port A 1-bit data output
      SPO => SPO,     -- Port B 1-bit data output
      A0 => A0,       -- Port A address[0] input bit
      A1 => A1,       -- Port A address[1] input bit
      A2 => A2,       -- Port A address[2] input bit
      A3 => A3,       -- Port A address[3] input bit
      A4 => A4,       -- Port A address[4] input bit
      D => D,         -- Port A 1-bit data input
      DPRA0 => DPRA0, -- Port B address[0] input bit
      DPRA1 => DPRA1, -- Port B address[1] input bit
      DPRA2 => DPRA2, -- Port B address[2] input bit
      DPRA3 => DPRA3, -- Port B address[3] input bit
      DPRA4 => DPRA4, -- Port B address[4] input bit
      WCLK => WCLK,   -- Port A write clock input
      WE => WE        -- Port A write enable input
   );

End of RAM32X1D_inst instantiation
```

Verilog Instantiation Template

```
//   RAM32X1D  : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//   instance  : in the body of the design code.  The instance name
// declaration : (RAM32X1D_inst) and/or the port declarations within the
//     code    : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // RAM32X1D: 32 x 1 positive edge write, asynchronous read dual-port distributed RAM
   //           Viretx-II/II-Pro
   // Xilinx HDL Libraries Guide Version 8.1i

   RAM32X1D #(
      .INIT(32'h00000000) // Initial contents of RAM
   ) RAM32X1D_inst (
      .DPO(DPO),     // Port A 1-bit data output
      .SPO(SPO),     // Port B 1-bit data output
      .A0(A0),       // Port A address[0] input bit
      .A1(A1),       // Port A address[1] input bit
      .A2(A2),       // Port A address[2] input bit
      .A3(A3),       // Port A address[3] input bit
      .A4(A4),       // Port A address[4] input bit
```

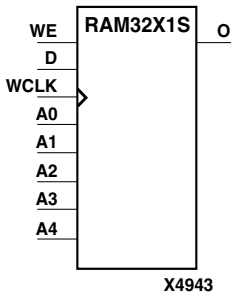
```
    .D(D),          // Port A 1-bit data input
    .DPRA0(DPRA0), // Port B address[0] input bit
    .DPRA1(DPRA1), // Port B address[1] input bit
    .DPRA2(DPRA2), // Port B address[2] input bit
    .DPRA3(DPRA3), // Port B address[3] input bit
    .DPRA4(DPRA4), // Port B address[4] input bit
    .WCLK(WCLK),    // Port A write clock input
    .WE(WE)         // Port A write enable input
);

// End of RAM32X1D_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

RAM32X1S

Primitive: 32-Deep by 1-Wide Static Synchronous RAM



RAM32X1S is a 32-word by 1-bit static random access memory 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 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. See "Specifying Initial Contents of a RAM" in the RAM16X1D section.

Mode selection is shown in the following truth table.

Inputs			Outputs
WE (mode)	WCLK	D	O
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D	D
1 (read)	↓	X	Data

Data = word addressed by bits A4 – A0

Usage

This design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the *XST User Guide*.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Hexa-decimal	Any 32-bit value.	All zeros	Specifies initial contents of the RAM.

VHDL Instantiation Template

```
--  RAM32X1S   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (RAM32X1S_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
```

```
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- RAM32X1S: 32 x 1 posedge write distributed  => LUT RAM
   --           All FPGA
   -- Xilinx HDL Libraries Guide Version 8.1i

   RAM32X1S_inst : RAM32X1S
   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_inst instantiation
```

Verilog Instantiation Template

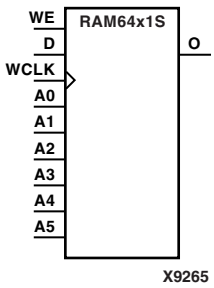
```
//  RAM32X1S   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (RAM32X1S_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // RAM32X1S: 32 x 1 posedge write distributed (LUT) RAM
   //           All FPGA
   // Xilinx HDL Libraries Guide Version 8.1i

   RAM32X1S #(
      .INIT(32'h00000000)  // Initial contents of RAM
   ) RAM32X1S_inst (
      .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_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

RAM64X1S

Primitive: 64-Deep by 1-Wide Static Synchronous RAM



RAM64X1S 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). 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 RAM64X1S during configuration using the INIT attribute. See "Specifying Initial Contents of a RAM" in the RAM16X1D section.

Mode selection is shown in the following truth table.

Inputs			Outputs
WE (mode)	**WCLK**	**D**	**O**
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D	D
1 (read)	↓	X	Data

Data = word addressed by bits A5 – A0

Usage

This design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the *XST User Guide*.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	64-Bit Hexa-decimal	64-Bit Hexadecimal	All zeros	Initializes ROMs, RAMs, registers, and look-up tables.

VHDL Instantiation Template

```
--  RAM64X1S   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (RAM64X1S_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
```

```
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- RAM64X1S: 64 x 1 positive edge write, asynchronous read single-port distributed RAM
   -- FPGAs
   -- Xilinx  HDL Libraries Guide version 8.1i

   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

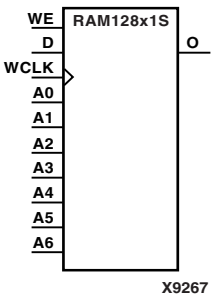
```
//  RAM64X1S   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (RAM64X1S_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // RAM64X1S: 64 x 1 positive edge write, asynchronous read single-port distributed RAM
   // FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   RAM64X1S #(
      .INIT(64'h0000000000000000) // Initial contents of RAM
   ) RAM64X1S_inst (
      .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
```

For More Information

Consult the Spartan-3E Data Sheet.

RAM128X1S

Primitive: 128-Deep by 1-Wide Static Synchronous RAM



RAM128X1S is a 128-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 7-bit address (A6 – 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 RAM128X1S during configuration using the INIT attribute. See "Specifying Initial Contents of a RAM" in the RAM16X1D section.

Mode selection is shown in the following truth table.

Inputs			Outputs
WE (mode)	**WCLK**	**D**	**O**
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D	D
1 (read)	↓	X	Data

Data = word addressed by bits A6 – A0

Usage

Below are example templates for instantiating this component into a design. These templates can be cut and pasteddirectly into the user's source code.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	128-Bit Hexadecimal	128-Bit Hexadecimal	All zeros	Initializes ROMs, RAMs, registers, and look-up tables.

VHDL Instantiation Template

```
--  RAM128X1S  : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (RAM128X1S_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
```

```
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- RAM128X1S: 128 x 1 positive edge write, asynchronous read single-port distributed RAM
   -- FPGAs.
   -- Xilinx  HDL Libraries Guide version 8.1i

   RAM128X1S_inst : RAM128X1S
   generic map (
      INIT => X"00000000000000000000000000000000")
   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
      A6 => A6,       -- Address[6] input bit
      D => D,         -- 1-bit data input
      WCLK => WCLK,   -- Write clock input
      WE => WE        -- Write enable input
   );

   -- End of RAM128X1S_inst instantiation
```

Verilog Instantiation Template

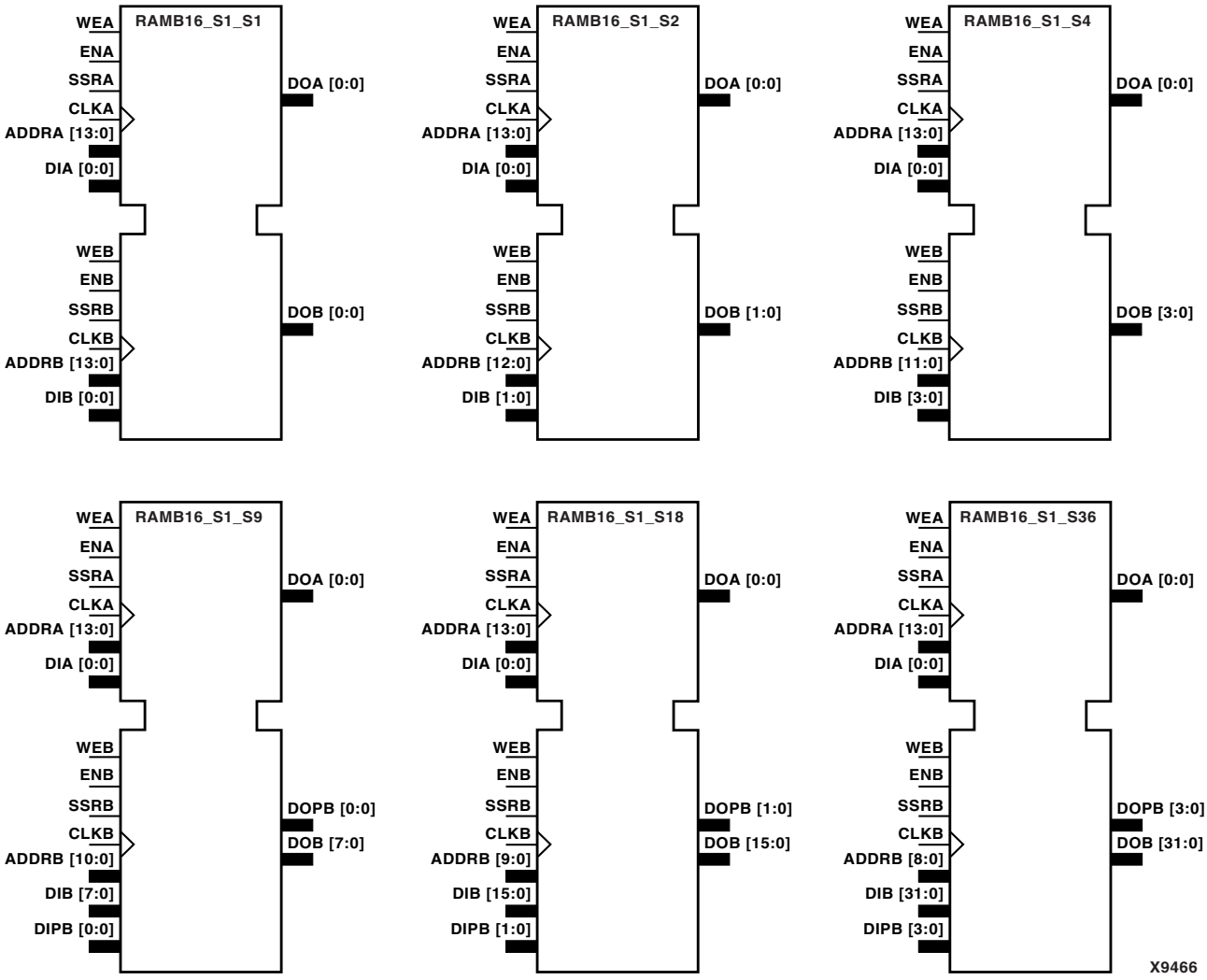
```
//  RAM128X1S  : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (RAM128X1S_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // RAM128X1S: 128 x 1 positive edge write, asynchronous read single-port distributed RAM
   // FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   RAM128X1S #(
      .INIT(128'h00000000000000000000000000000000) // Initial contents of RAM
   ) RAM128X1S_inst (
      .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
      .A6(A6),       // Address[6] input bit
      .D(D),         // 1-bit data input
      .WCLK(WCLK),   // Write clock input
      .WE(WE)        // Write enable input
   );

   // End of RAM128X1S_inst instantiation
```

For More Information

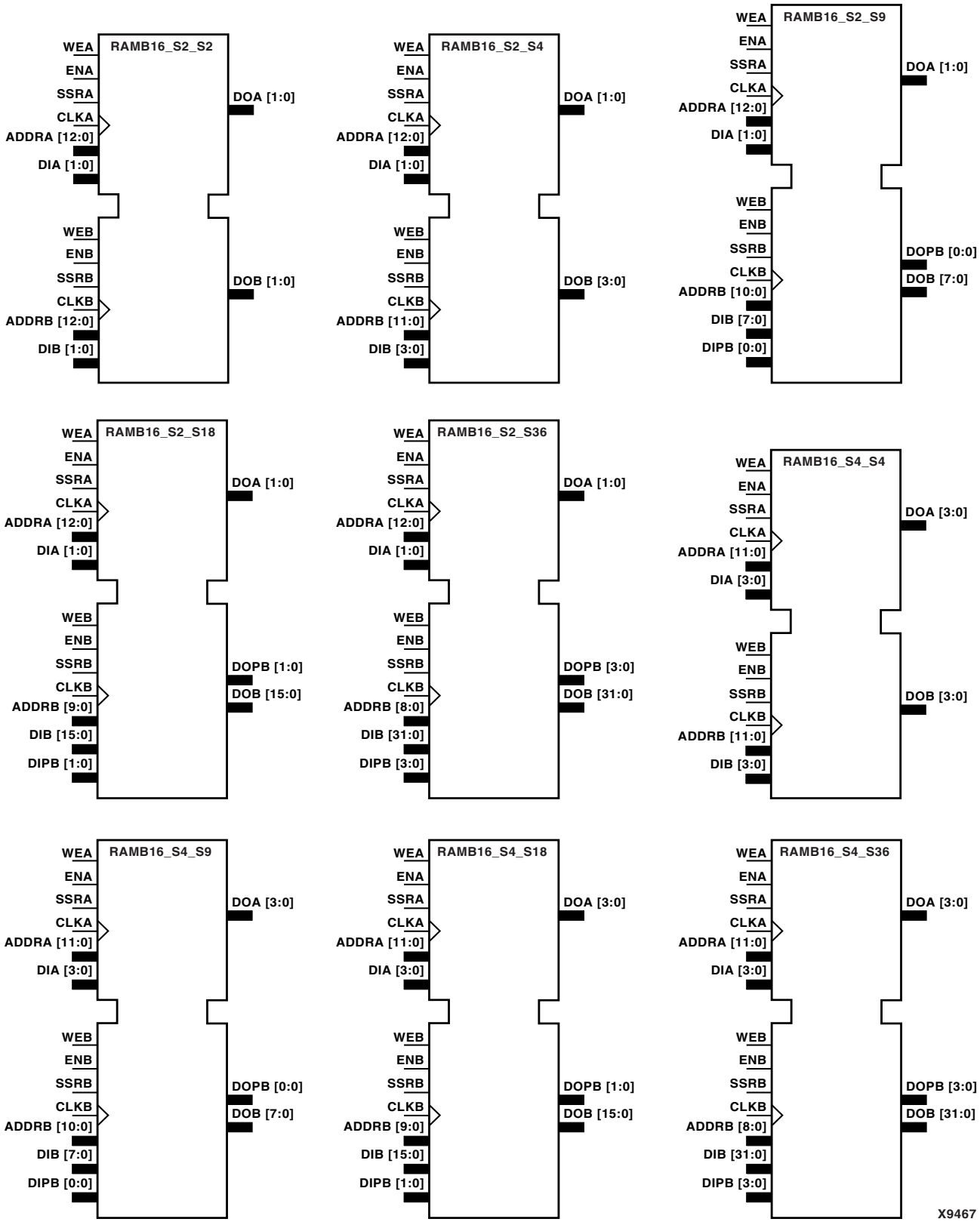
Consult the Spartan-3E Data Sheet.

RAMB16_S*m*_S*n*

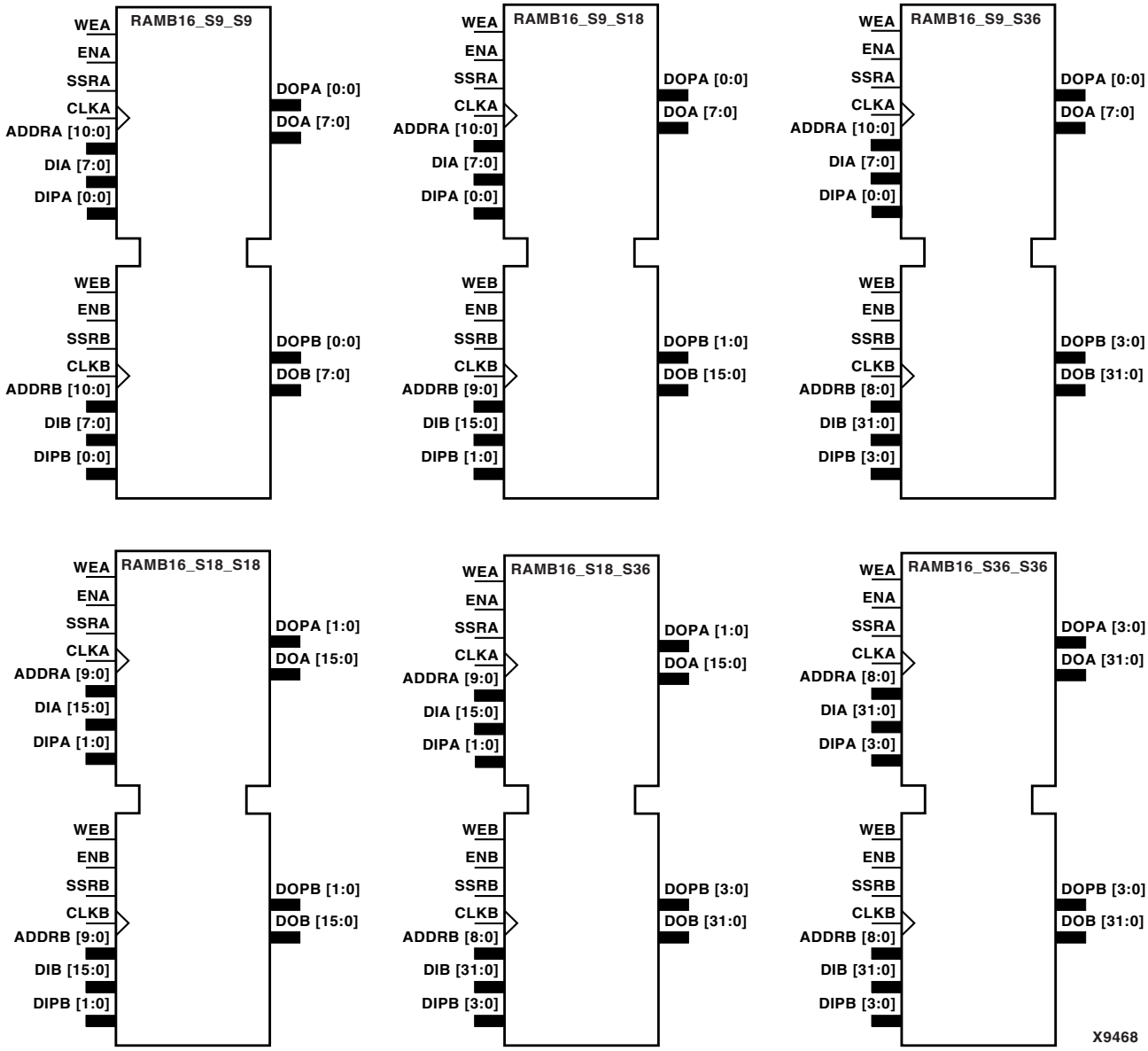
Primitive: 16384-Bit Data Memory and 2048-Bit Parity Memory, Dual-Port Synchronous Block RAM with Port Width (m or n) Configured to 1, 2, 4, 9, 18, or 36 Bits



RAMB16_S1_S1 through RAMB16_S1_S36 Representations



RAMB16_S2_S2 through RAMB16_S4_S36 Representations



RAMB16_S9_S9 through RAMB16_S36_S36 Representations

The RAMB16_S*m*_S*n* components listed in the following table are dual-ported dedicated random access memory blocks with synchronous write capability. Each block RAM port has 16384 bits of data memory. Ports configured as 9, 18, or 36-bits wide have an additional 2048 bits of parity memory. Each port is independent of the other while accessing the same set of 16384 data memory cells. Each port is

independently configured to a specific data width. The possible port and cell configurations are listed in the following table.

Component	Port A					Port B				
	Data Cells[a]	Parity Cells[a]	Address Bus	Data Bus	Parity Bus	Data Cells[a]	Parity Cells[a]	Address Bus	Data Bus	Parity Bus
RAMB16_S1_S1	16384 x 1	-	(13:0)	(0:0)	-	16384 x 1	-	(13:0)	(0:0)	-
RAMB16_S1_S2	16384 x 1	-	(13:0)	(0:0)	-	8192 x 2	-	(12:0)	(1:0)	-
RAMB16_S1_S4	16384 x 1	-	(13:0)	(0:0)	-	4096 x 4	-	(11:0)	(3:0)	-
RAMB16_S1_S9	16384 x 1	-	(13:0)	(0:0)	-	2048 x 8	2048 x 1	(10:0)	(7:0)	(0:0)
RAMB16_S1_S18	16384 x 1	-	(13:0)	(0:0)	-	1024 x 16	1024 x 2	(9:0)	(15:0)	(1:0)
RAMB16_S1_S36	16384 x 1	-	(13:0)	(0:0)	-	512 x 32	512 x 4	(8:0)	(31:0)	(3:0)
RAMB16_S2_S2	8192 x 2	-	(12:0)	(1:0)	-	8192 x 2	-	(12:0)	(1:0)	-
RAMB16_S2_S4	8192 x 2	-	(12:0)	(1:0)	-	4096 x 4	-	(11:0)	(3:0)	-
RAMB16_S2_S9	8192 x 2	-	(12:0)	(1:0)	-	2048 x 8	2048 x 1	(10:0)	(7:0)	(0:0)
RAMB16_S2_S18	8192 x 2	-	(12:0)	(1:0)	-	1024 x 16	1024 x 2	(9:0)	(15:0)	(1:0)
RAMB16_S2_S36	8192 x 2	-	(12:0)	(1:0)	-	512 x 32	512 x 4	(8:0)	(31:0)	(3:0)
RAMB16_S4_S4	4096 x 4	-	(11:0)	(3:0)	-	4096 x 4	-	(11:0)	(3:0)	-
RAMB16_S4_S9	4096 x 4	-	(11:0)	(3:0)	-	2048 x 8	2048 x 1	(10:0)	(7:0)	(0:0)
RAMB16_S4_S18	4096 x 4	-	(11:0)	(3:0)	-	1024 x 16	1024 x 2	(9:0)	(15:0)	(1:0)
RAMB16_S4_S36	4096 x 4	-	(11:0)	(3:0)	-	512 x 32	512 x 4	(8:0)	(31:0)	(3:0)
RAMB16_S9_S9	2048 x 8	2048 x 1	(10:0)	(7:0)	(0:0)	2048 x 8	2048 x 1	(10:0)	(7:0)	(0:0)
RAMB16_S9_S18	2048 x 8	2048 x 1	(10:0)	(7:0)	(0:0)	1024 x 16	1024 x 2	(9:0)	(15:0)	(1:0)
RAMB16_S9_S36	2048 x 8	2048 x 1	(10:0)	(7:0)	(0:0)	512 x 32	512 x 4	(8:0)	(31:0)	(3:0)
RAMB16_S18_S18	1024 x 16	1024 x 2	(9:0)	(15:0)	(1:0)	1024 x 16	1024 x 2	(9:0)	(15:0)	(1:0)
RAMB16_S18_S36	1024 x 16	1024 x 2	(9:0)	(15:0)	(1:0)	512 x 32	512 x 4	(8:0)	(31:0)	(3:0)

	Port A					Port B				
RAMB16_S36_S36	512 x 32	512 x 4	(8:0)	(31:0)	(3:0)	512 x 32	512 x 4	(8:0)	(31:0)	(3:0)

[a]Depth x Width

Each port is fully synchronous with independent clock pins. All port A input pins have setup time referenced to the CLKA pin and its data output bus DOA has a clock-to-out time referenced to the CLKA. All port B input pins have setup time referenced to the CLKB pin and its data output bus DOB has a clock-to-out time referenced to the CLKB.

The enable ENA pin controls read, write, and reset for port A. When ENA is Low, no data is written and the outputs (DOA and DOPA) retain the last state. When ENA is High and reset (SSRA) is High, DOA and DOPA are set to SRVAL_A during the Low-to-High clock (CLKA) transition; if write enable (WEA) is High, the memory contents reflect the data at DIA and DIPA. When ENA is High and WEA is Low, the data stored in the RAM address (ADDRA) is read during the Low-to-High clock transition. By default, WRITE_MODE_A=WRITE_FIRST, when ENA and WEA are High, the data on the data inputs (DIA and DIPA) is loaded into the word selected by the write address (ADDRA) during the Low-to-High clock transition and the data outputs (DOA and DOPA) reflect the selected (addressed) word.

The enable ENB pin controls read, write, and reset for port B. When ENB is Low, no data is written and the outputs (DOB and DOPB) retain the last state. When ENB is High and reset (SSRB) is High, DOB and DOPB are set to SRVAL_B during the Low-to-High clock (CLKB) transition; if write enable (WEB) is High, the memory contents reflect the data at DIB and DIPB. When ENB is High and WEB is Low, the data stored in the RAM address (ADDRB) is read during the Low-to-High clock transition. By default, WRITE_MODE_B=WRITE_FIRST, when ENB and WEB are High, the data on the data inputs (DIB and PB) are loaded into the word selected by the write address (ADDRB) during the Low-to-High clock transition and the data outputs (DOB and DOPB) reflect the selected (addressed) word.

The above descriptions assume active High control pins (ENA, WEA, SSRA, CLKA, ENB, WEB, SSRB, and CLKB). However, the active level can be changed by placing an inverter on the port. Any inverter placed on a RAMB16 port is absorbed into the block and does not use a CLB resource.

Port A Truth Table

Inputs								Outputs			
GSR	ENA	SSRA	WEA	CLKA	ADDRA	DIA	DIPA	DOA	DOPA	RAM Contents	
										Data RAM	Parity RAM
1	X	X	X	X	X	X	X	INIT_A	INIT_A	No Chg	No Chg
0	0	X	X	X	X	X	X	No Chg	No Chg	No Chg	No Chg
0	1	1	0	↑	X	X	X	SRVAL_A	SRVAL_A	No Chg	No Chg
0	1	1	1	↑	addr	data	pdata	SRVAL_A	SRVAL_A	RAM(addr) =>data	RAM(addr) =>pdata
0	1	0	0	↑	addr	X	X	RAM(addr)	RAM(addr)	No Chg	No Chg

Port A Truth Table

Inputs								Outputs			
GSR	ENA	SSRA	WEA	CLKA	ADDRA	DIA	DIPA	DOA	DOPA	RAM Contents	
0	1	0	1	↑	addr	data	pdata	No Chg[1] RAM (addr)[2] data[3]	No Chg[1] RAM(addr)[2] pdata[3]	RAM(addr) =>data	RAM(addr) =>pdata

GSR=Global Set Reset

INIT_A=Value specified by the INIT_A attribute for output register. Default is all zeros.
SRVAL_A=register value

addr=RAM address
RAM(addr)=RAM contents at address ADDR
data=RAM input data
pdata=RAM parity data

[1]WRITE_MODE_A=NO_CHANGE
[2]WRITE_MODE_A=READ_FIRST
[3]WRITE_MODE_A=WRITE_FIRST

Port B Truth Table

Inputs								Outputs			
GSR	ENB	SSRB	WEB	CLKB	ADDRB	DIB	DIPB	DOB	DOPB	RAM Contents	
										Data RAM	Parity RAM
1	X	X	X	X	X	X	X	INIT_B	INIT_B	No Chg	No Chg
0	0	X	X	X	X	X	X	No Chg	No Chg	No Chg	No Chg
0	1	1	0	↑	X	X	X	SRVAL_B	SRVAL_B	No Chg	No Chg
0	1	1	1	↑	addr	data	pdata	SRVAL_B	SRVAL_B	RAM(addr) =>data	RAM(addr) =>pdata
0	1	0	0	↑	addr	X	X	RAM(addr)	RAM(addr)	No Chg	No Chg

Port B Truth Table

Inputs								Outputs			
GSR	ENB	SSRB	WEB	CLKB	ADDRB	DIB	DIPB	DOB	DOPB	RAM Contents	
0	1	0	1	↑	addr	data	pdata	No Chg[1] RAM (addr)[2] data[3]	No Chg[1] RAM(addr)[2] pdata[3]	RAM(addr) =>data	RAM(addr) =>pdata

GSR=Global Set Reset

INIT_B=Value specified by the INIT_B attribute for output registers. Default is all zeros.
SRVAL_B=register value

addr=RAM address
RAM(addr)=RAM contents at address ADDR
data=RAM input data
pdata=RAM parity data

[1]WRITE_MODE_B=NO_CHANGE
[2]WRITE_MODE_B=READ_FIRST
[3]WRITE_MODE_B=WRITE_FIRST

Address Mapping

Each port accesses the same set of 18432 memory cells using an addressing scheme that is dependent on the width of the port. For all port widths, 16384 memory cells are available for data as shown in the "Port Address Mapping for Data" table below. For 9-, 18-, and 36-bit wide ports, 2408 parity memory cells are also available as shown in "Port Address Mapping for Parity" table below. The physical RAM location that is addressed for a particular width is determined from the following formula.

```
Start=((ADDR port+1)*(Widthport)) -1
End=(ADDRport)*(Widthport)
```

The following tables shows address mapping for each port width.

Port Address Mapping for Data

Data Width	Port Data Addresses																															
1	16384	<--	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
2	8192	<--	15		14		13		12		11		10		09		08		07		06		05		04		03		02		01		00	
4	4096	<--	07				06				05				04				03				02				01				00	
8	2048	<--	03								02								01								00					
16	1024	<--	01																00													
32	512	<--	00																													

Port Address Mapping for Parity

Parity Width	Port Parity Addresses						
1	2048	<-----	03	02	01	00	
2	1024	<-----	01		00		
4	512	<-----	00				

Initializing Memory Contents of a Dual-Port RAMB16

You can use the INIT_xx attributes to specify an initialization value for the memory contents of a RAMB16 during device configuration. The initialization of each RAMB16_Sm_Sn is set by 64 initialization attributes (INIT_00 through INIT_3F) of 64 hex values for a total of 16384 bits.

You can use the INITP_xx attributes to specify an initial value for the parity memory during device configuration or assertion. The initialization of the parity memory for ports configured for 9, 18, or 36 bits is set by 8 initialization attributes (INITP_00 through INITP_07) of 64 hex values for a total of 2048 bits.

If any INIT_xx or INITP_xx attribute is not specified, it is configured as zeros. Partial strings are padded with zeros to the left.

See the *Constraints Guide* for more information on these attributes.

Initializing the Output Register of a Dual-Port RAMB16

In Spartan-3E, each bit in an output register can be initialized at power on (when GSR is high) to either a 0 or 1. In addition, the initial state specified for power on can be different than the state that results from assertion of a set/reset. Four properties control initialization of the output register for a dual-port RAMB16: INIT_A, INIT_B, SRVAL_A, and SRVAL_B. The INIT_A attribute specifies the output register value at power on for port A and the INIT_B attribute specifies the value for port B. You can use the SRVAL_A attribute to define the state resulting from assertion of the SSR (set/reset) input on port A. You can use the SRVAL_B attribute to define the state resulting from assertion of the SSR input on port B.

The INIT_A, INIT_B, SRVAL_A, and SRVAL_B attributes specify the initialization value as a hexadecimal String. The value is dependent upon the port width. For example, for a RAMB16_S1_S4 with port A width equal to 1 and port B width equal to 4, the port A output register contains 1 bit and the port B output register contains 4 bits. Therefore, the INIT_A or SRVAL_A value can only be specified as a 1 or 0. For port B, the output register contains 4 bits. In this case, you can use INIT_B or SRVAL_B to specify a hexadecimal value from 0 through F to initialize the 4 bits of the output register.

For those ports that include parity bits, the parity portion of the output register is specified in the high order bit position of the INIT_A, INIT_B, SRVAL_A, or SRVAL_B value.

The INIT and SRVAL attributes default to zero if they are not set by the user.

See the *Constraints Guide* for more information on these attributes.

Write Mode Selection

The WRITE_MODE_A attribute controls the memory and output contents of port A for a dual-port RAMB16. The WRITE_MODE_B attribute does the same for port B. By default, both WRITE_MODE_A and WRITE_MODE_B are set to WRITE_FIRST. This means that input is read, written to memory, and then passed to output. You can set the write mode for port A and/or port B to READ_FIRST to read the memory contents, pass the memory contents to the outputs, and then write the input to memory. Or, you can set the write mode to NO_CHANGE to have the input written to memory without changing the output. The "Port A and Port B Conflict Resolution" section describes how read/write conflicts are resolved when both port A and port B are attempting to read/write to the same memory cells.

Port A and Port B Conflict Resolution

Spartan-3E block SelectRAM is True Dual-Port RAM that allows both ports to simultaneously access the same memory cell. When one port writes to a given memory cell, the other port must not address that memory cell (for a write or a read) within the clock-to-clock setup window.

The following tables summarize the collision detection behavior of the dual-port RAMB16 based on the WRITE_MODE_A and WRITE_MODE_B settings.

WRITE_MODE_A=NO_CHANGE and WRITE_MODE_B=NO_CHANGE

WEA	WEB	CLKA	CLKB	DIA	DIB	DIPA	DIPB	DOA	DOB	DOPA	DOPB	Data RAM	Parity Ram
0	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	No Chg	No Chg
1	0	↑	↑	DIA	DIB	DIPA	DIPB	No Chg	X	No Chg	X	DIA	DIPA
0	1	↑	↑	DIA	DIB	DIPA	DIPB	X	No Chg	X	No Chg	DIB	DIPB
1	1	↑	↑	DIA	DIB	DIPA	DIPB	No Chg	No Chg	No Chg	No Chg	X	X

WRITE_MODE_A=READ_FIRST and WRITE_MODE_B=READ_FIRST

WEA	WEB	CLKA	CLKB	DIA	DIB	DIPA	DIPB	DOA	DOB	DOPA	DOPB	Data RAM	Parity Ram
0	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	No Chg	No Chg
1	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	DIA	DIPA
0	1	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	DIB	DIPB
1	1	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	X	X

WRITE_MODE_A= WRITE_FIRST and WRITE_MODE_B=WRITE_FIRST

WEA	WEB	CLKA	CLKB	DIA	DIB	DIPA	DIPB	DOA	DOB	DOPA	DOPB	Data RAM	Parity Ram
0	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	No Chg	No Chg
1	0	↑	↑	DIA	DIB	DIPA	DIPB	DIA	X	DIPA	X	DIA	DIPA

WRITE_MODE_A= WRITE_FIRST and WRITE_MODE_B=WRITE_FIRST

WEA	WEB	CLKA	CLKB	DIA	DIB	DIPA	DIPB	DOA	DOB	DOPA	DOPB	Data RAM	Parity Ram
0	1	↑	↑	DIA	DIB	DIPA	DIPB	X	DIB	X	DIPB	DIB	DIPB
1	1	↑	↑	DIA	DIB	DIPA	DIPB	X	X	X	X	X	X

WRITE_MODE_A=NO_CHANGE and WRITE_MODE_B=READ_FIRST

WEA	WEB	CLKA	CLKB	DIA	DIB	DIPA	DIPB	DOA	DOB	DOPA	DOPB	Data RAM	Parity Ram
0	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	No Chg	No Chg
1	0	↑	↑	DIA	DIB	DIPA	DIPB	No Chg	X	No Chg	X	DIA	DIPA
0	1	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	DIB	DIPB
1	1	↑	↑	DIA	DIB	DIPA	DIPB	No Chg	X	No Chg	X	DIB	DIPB

WRITE_MODE_A=NO_CHANGE and WRITE_MODE_B=WRITE_FIRST

WEA	WEB	CLKA	CLKB	DIA	DIB	DIPA	DIPB	DOA	DOB	DOPA	DOPB	Data RAM	Parity Ram
0	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	No Chg	No Chg
1	0	↑	↑	DIA	DIB	DIPA	DIPB	No Chg	X	No Chg	X	DIA	DIPA
0	1	↑	↑	DIA	DIB	DIPA	DIPB	X	DIB	X	DIPB	DIB	DIPB
1	1	↑	↑	DIA	DIB	DIPA	DIPB	No Chg	X	No Chg	X	X	X

WRITE_MODE_A=READ_FIRST and WRITE_MODE_B=WRITE_FIRST

WEA	WEB	CLKA	CLKB	DIA	DIB	DIPA	DIPB	DOA	DOB	DOPA	DOPB	Data RAM	Parity Ram
0	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	No Chg	No Chg
1	0	↑	↑	DIA	DIB	DIPA	DIPB	RAM	RAM	RAM	RAM	DIA	DIPA
0	1	↑	↑	DIA	DIB	DIPA	DIPB	X	DIB	X	DIPB	DIB	DIPB
1	1	↑	↑	DIA	DIB	DIPA	DIPB	X	DIB	X	DIPB	DIA	DIPA

Usage

These design elements can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the *XST User Guide*.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT_00 To INIT_3F	Binary/Hexadecimal	Any	All zeros	Specifies the initial contents of the data portion of the RAM array.
INIT_A	Binary/Hexadecimal	Any	All zeros	Identifies the initial value of the DOA/DOB output port after completing configuration. For Type, the bit width is dependent on the width of the A or B port of the RAM.
INIT_B	Binary/Hexadecimal	Any	All zeros	Identifies the initial value of the DOA/DOB output port after completing configuration. For Type, the bit width is dependent on the width of the A or B port of the RAM.
INITP_00 To INITP_07	Binary/Hexadecimal	Any	All zeros	Specifies the initial contents of the parity portion of the RAM array.
SIM_COLLISION_CHECK	String	¨ALL”, ¨NONE”, ¨WARNING”, or ¨GENERATE_X_ONLY”	¨ALL”	Specifies the behavior during simulation in the event of a data collision (data being read or written to the same address from both ports of the Ram simultaneously. "ALL" issues a warning to simulator console and generate an X or all unknown data due to the collision. This is the recommended setting. "WARNING" generates a warning only and "GENERATE_X_ONLY" generates an X for unknown data but won’t output the occurrence to the simulation console. "NONE" completely ignores the error. It is suggested to only change this attribute if you can ensure the data generated during a collision is discarded.
SRVAL_A	Binary/Hexadecimal	Any	All zeros	Allows the individual selection of whether the DOA/DOB output port sets (go to a one) or reset (go to a zero) upon the assertion of the SSRA/SSRB pin. For Type, the bit width is dependent on the width of the A or B port of the RAM.

Attribute	Type	Allowed Values	Default	Description
SRVAL_B	Binary/Hex-adecimal	Any	All zeros	Allows the individual selection of whether the DOA/DOB output port sets (go to a one) or reset (go to a zero) upon the assertion of the SSRA/SSRB pin. For Type, the bit width is dependent on the width of the A or B port of the RAM.
WRITE_MODE_A	String	"WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"	"WRITE_FIRST"	Specifies the behavior of the DOA/DOB port upon a write command to the respected port. If set to "WRITE_FIRST", the same port that is written to displays the contents of the written data to the outputs upon completion of the operation. "READ_FIRST" displays the prior contents of the RAM to the output port prior to writing the new data. "NO_CHANGE" keeps the previous value on the output port and won't update the output port upon a write command. This is the suggested mode if not using the read data from a particular port of the RAM.
WRITE_MODE_B	String	"WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"	"WRITE_FIRST"	Specifies the behavior of the DOA/DOB port upon a write command to the respected port. If set to "WRITE_FIRST", the same port that is written to displays the contents of the written data to the outputs upon completion of the operation. "READ_FIRST" displays the prior contents of the RAM to the output port prior to writing the new data. "NO_CHANGE" keeps the previous value on the output port and won't update the output port upon a write command. This is the suggested mode if not using the read data from a particular port of the RAM.

VHDL and Verilog Instantiation

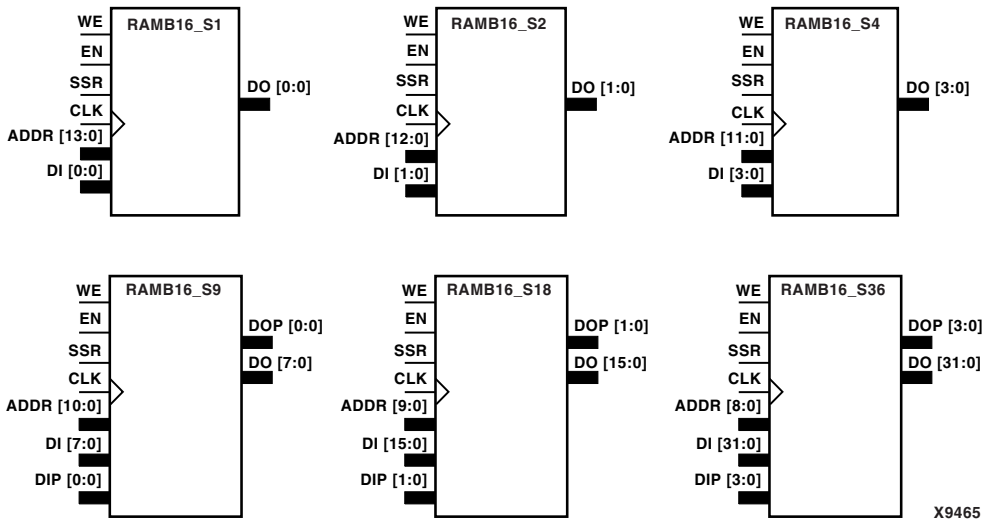
For VHDL and Verilog coding examples for each configuration of this RAM, refer to the ISE HDL Language Templates in the ISE Project Navigator software.

For More Information

Consult the Spartan-3E Data Sheet.

RAMB16_S*n*

Primitive: 16384-Bit Data Memory and 2048-Bit Parity Memory, Single-Port Synchronous Block RAM with Port Width (n) Configured to 1, 2, 4, 9, 18, or 36 Bits



RAMB16_S1 through RAMB16_S36 Representations

RAMB16_S1, RAMB16_S2, RAMB16_S4, RAMB16_S9, RAMB16_S18, and RAMB16_S36 are dedicated random access memory blocks with synchronous write capability. The block RAM port has 16384 bits of data memory. RAMB16_S9, RAMB16_S18, and RAMB16_S36 have an additional 2048 bits of parity memory. The RAMB16_Sn cell configurations are listed in the following table.

The enable (EN) pin controls read, write, and reset. When EN is Low, no data is

Component	Data Cells		Parity Cells		Address Bus	Data Bus	Parity Bus
	Depth	Width	Depth	Width			
RAMB16_S1	16384	1	-	-	(13:0)	(0:0)	-
RAMB16_S2	8192	2	-	-	(12:0)	(1:0)	-
RAMB16_S4	4096	4	-	-	(11:0)	(3:0)	-
RAMB16_S9	2048	8	2048	1	(10:0)	(7:0)	(0:0)
RAMB16_S18	1024	16	1024	2	(9:0)	(15:0)	(1:0)
RAMB16_S36	512	32	512	4	(8:0)	(31:0)	(3:0)

written and the outputs (DO and DOP) retain the last state. When EN is High and reset (SSR) is High, DO and DOP are set to SRVAL during the Low-to-High clock (CLK) transition; if write enable (WE) is High, the memory contents reflect the data at DI and DIP. When SSR is Low, EN is High, and WE is Low, the data stored in the RAM address (ADDR) is read during the Low-to-High clock transition. The output value depends on the mode. By default WRITE_MODE=WRITE_FIRST, when EN and WE

are High and SSR is Low, the data on the data inputs (DI and DIP) is loaded into the word selected by the write address (ADDR) during the Low-to-High clock transition. See "Write Mode Selection" for information on setting the WRITE_MODE.

The above description assumes an active High EN, WE, SSR, and CLK. However, the active level can be changed by placing an inverter on the port. Any inverter placed on a RAMB16 port is absorbed into the block and does not use a CLB resource.

Inputs								Outputs			
GSR	EN	SSR	WE	CLK	ADDR	DI	DIP	DO	DOP	RAM Contents	
										Data RAM	Parity RAM
1	X	X	X	X	X	X	X	INIT	INIT	No Chg	No Chg
0	0	X	X	X	X	X	X	No Chg	No Chg	No Chg	No Chg
0	1	1	0	↑	X	X	X	SRVAL	SRVAL	No Chg	No Chg
0	1	1	1	↑	addr	data	pdata	SRVAL	SRVAL	RAM(addr) =>data	RAM(addr) =>pdata
0	1	0	0	↑	addr	X	X	RAM(addr)	RAM(addr)	No Chg	No Chg
0	1	0	1	↑	addr	data	pdata	No Chg[a] RAM (addr)[b] data[c]	No Chg[a] RAM(addr) [b] pdata[c]	RAM(addr) =>data	RAM(addr) =>pdata

GSR=Global Set Reset signal

INIT=Value specified by the INIT attribute for data memory. Default is all zeros.
SRVAL=Value after assertion of SSR as specified by the SRVAL attribute.

addr=RAM address
RAM(addr)=RAM contents at address ADDR
data=RAM input data
pdata=RAM parity data

[a]WRITE_MODE=NO_CHANGE
[b]WRITE_MODE=READ_FIRST
[c]WRITE_MODE=WRITE_FIRST

Initializing Memory Contents of a Single-Port RAMB16

You can use the INIT_xx attributes to specify an initialization value for the memory contents of a RAMB16 during device configuration. The initialization of each RAMB16_Sn is set by 64 initialization attributes (INIT_00 through INIT_3F) of 64 hex values for a total of 16384 bits.

You can use the INITP_xx attributes to specify an initial value for the parity memory during device configuration or assertion. The initialization of the parity memory for ports configured for 9, 18, or 36 bits is set by 8 initialization attributes (INITP_00 through INITP_07) of 64 hex values for a total of 2048 bits.

If any INIT_xx or INITP_xx attribute is not specified, it is configured as zeros. Partial strings are padded with zeros to the left.

See the *Constraints Guide* for more information on these attributes.

Initializing the Output Register of a Single-Port RAMB16

In Spartan-3E, each bit in the output register can be initialized at power on to either a 0 or 1. In addition, the initial state specified for power on can be different than the state that results from assertion of a set/reset. Two types of properties control initialization of the output register for a single-port RAMB16: INIT and SRVAL. The INIT attribute specifies the output register value at power on. You can use the SRVAL attribute to define the state resulting from assertion of the SSR (set/reset) input.

The INIT and SRVAL attributes specify the initialization value as a hexadecimal String. The value is dependent upon the port width. For example, for a RAMB16_S1 with port width equal to 1, the output register contains 1 bit. Therefore, the INIT or SRVAL value can only be specified as a 1 or 0. For RAMB16_S4 with port width equal to 4, the output register contains 4 bits. In this case, you can specify a hexadecimal value from 0 through F to initialize the 4 bits of the output register.

For those ports that include parity bits, the parity portion of the output register is specified in the high order bit position of the INIT or SRVAL value.

The INIT and SRVAL attributes default to zero if they are not set by the user.

Write Mode Selection

The WRITE_MODE attribute controls RAMB16 memory and output contents. By default, the WRITE_MODE is set to WRITE_FIRST. This means that input is read, written to memory, and then passed to output. You can set the WRITE_MODE to READ_FIRST to read the memory contents, pass the memory contents to the outputs, and then write the input to memory. Or, you can set the WRITE_MODE to NO_CHANGE to have the input written to memory without changing the output.

Usage

This design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the *XST User Guide*.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Binary/Hexa-decimal	Any	All zeros	Identifies the initial value of the DO output port after completing configuration. The bit width is dependent on the width of the A or B port of the RAM.
INIT_00 ? INIT_3F	Binary/Hexa-decimal	Any	All zeros	Specifies the initial contents of the data portion of the RAM array.
INITP_00 ? INITP_07	Binary/Hexa-decimal	Any	All zeros	Specifies the initial contents of the parity portion of the RAM array.

Attribute	Type	Allowed Values	Default	Description
SRVAL	Binary/Hexa-decimal	Any	All zeros	Allows the individual selection of whether the DO output port sets (go to a one) or reset (go to a zero) upon the assertion of the SSR pin. The bit width is dependent on the width of the A or B port of the RAM.
WRITE_ MODE	String	"WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"	"WRITE_ FIRST"	Specifies the behavior of the DO port upon a write command to the respected port. If set to "WRITE_FIRST", the same port that is written to displays the contents of the written data to the outputs upon completion of the operation. "READ_FIRST" displays the prior contents of the RAM to the output port prior to writing the new data. "NO_CHANGE" keeps the previous value on the output port and won't update the output port upon a write command. This is the suggested mode if not using the read data from a particular port of the RAM.

VHDL and Verilog Instantiation

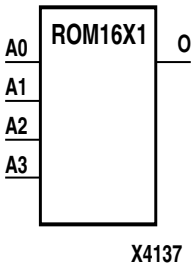
For VHDL and Verilog coding examples for each configuration of this RAM, refer to the ISE HDL Language Templates in the ISE Project Navigator software.

For More Information

Consult the Spartan-3E Data Sheet.

ROM16X1

Primitive: 16-Deep by 1-Wide ROM



ROM16X1 is a 16-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 4-bit address (A3 – A0). The ROM is initialized with the INIT = value parameter during configuration. The value consists of four hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H. For example, the INIT=10A7 parameter produces the data stream:

```
0001 0000 1010 0111
```

An error occurs if the INIT=*value* is not specified.

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Hexa-decimal	Any 16-bit value.	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

```
--   ROM16X1   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (ROM16X1_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- ROM16X1: 16 x 1 Asynchronous Distributed  => LUT ROM
   -- Xilinx HDL Libraries Guide Version 8.1i

   ROM16X1_inst : ROM16X1
   generic map (
      INIT => X"0000")
   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]
   );

   -- End of ROM16X1_inst instantiation
```

Verilog Instantiation Code

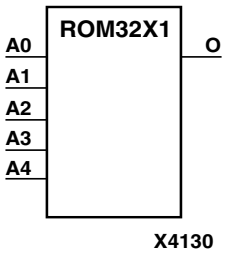
```
//   ROM16X1   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (ROM16X1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // ROM16X1: 16 x 1 Asynchronous Distributed (LUT) ROM
   //          All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   ROM16X1 #(
      .INIT(16'h0000) // Contents of ROM
   ) ROM16X1_inst (
      .O(O),   // ROM output
      .A0(A0), // ROM address[0]
      .A1(A1), // ROM address[1]
      .A2(A2), // ROM address[2]
      .A3(A3)  // ROM address[3]
   );

   // End of ROM16X1_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

ROM32X1

Primitive: 32-Deep by 1-Wide ROM



ROM32X1 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 with the INIT = value parameter during configuration. 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.

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Hexa-decimal	Any 32-bit value.	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

```
--   ROM32X1   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (ROM32X1_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- ROM32X1: 32 x 1 Asynchronous Distributed  => LUT ROM
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 Code

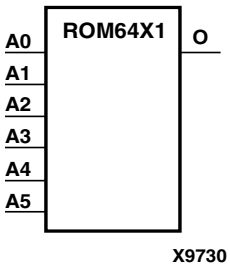
```
//   ROM32X1   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (ROM32X1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // ROM32X1: 32 x 1 Asynchronous Distributed (LUT) ROM
   //          All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   ROM32X1 #(
      .INIT(32'h00000000) // Contents of ROM
   ) ROM32X1_inst (
      .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
```

For More Information

Consult the Spartan-3E Data Sheet.

ROM64X1

Primitive: 64-Deep by 1-Wide ROM



ROM64X1 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 with an INIT = value parameter during configuration. 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.

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Hexa-decimal	Any 64-bit value.	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

```
--   ROM64X1   : In order to incorporate this function into the design,
--     VHDL    : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (ROM64X1_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- ROM64X1: 64 x 1 Asynchronous Distributed  => LUT ROM
   -- FPGAs.
   -- Xilinx HDL Libraries Guide Version 8.1i

   ROM64X1_inst : ROM64X1
   generic map (
      INIT => X"0000000000000000")
   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]
   );

   -- End of ROM64X1_inst instantiation
```

Verilog Instantiation Code

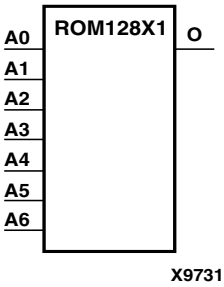
```
//   ROM64X1   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (ROM64X1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

//  <-----Cut code below this line---->

   // ROM64X1: 64 x 1 Asynchronous Distributed (LUT) ROM
   // FPGAs.
// Xilinx HDL Libraries Guide Version 8.1i

   ROM64X1 #(
      .INIT(64'h0000000000000000) // Contents of ROM
   ) ROM64X1_inst (
      .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]
   );

   // End of ROM64X1_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

ROM128X1

Primitive: 128-Deep by 1-Wide ROM



ROM128X1 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 with an INIT = value parameter during configuration. 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.

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Hexa-decimal	Any 128-bit value.	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

```
--  ROM128X1   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (ROM128X1_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- ROM128X1: 128 x 1 Asynchronous Distributed  => LUT ROM
   -- FPGAs.
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 Code

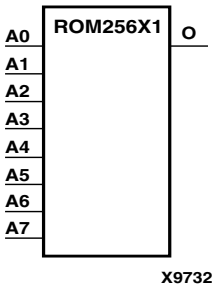
```
//   ROM128X1   : In order to incorporate this function into the design,
//   Verilog    : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration : (ROM128X1_inst) and/or the port declarations within the
//    code      : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  All inputs
//              : and outputs must be connect.

//  <-----Cut code below this line---->

   // ROM128X1: 128 x 1 Asynchronous Distributed (LUT) ROM
   // FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   ROM128X1 #(
      .INIT(128'h00000000000000000000000000000000) // Contents of ROM
   ) ROM128X1_inst (
      .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
```

For More Information

Consult the Spartan-3E Data Sheet.

ROM256X1

Primitive: 256-Deep by 1-Wide ROM



ROM256X1 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 with an INIT=value parameter during configuration. 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.

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	Hexa-decimal	Any 256-bit value.	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

```
--   ROM256X1   : In order to incorporate this function into the design,
--     VHDL     : the following instance declaration needs to be placed
--   instance   : in the architecture body of the design code.  The
-- declaration : instance name (ROM256X1_inst) and/or the port declarations
--    code      : after the "=>" assignment maybe changed to properly
--              : reference and connect this function to the design.
--              : All inputs and outputs must be connected.

--   Library    : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for      : added before the entity declaration.  This library
--   Xilinx     : contains the component declarations for all Xilinx
--  primitives  : primitives and points to the models that are used
--              : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- ROM256X1: 256 x 1 Asynchronous Distributed  => LUT ROM
   -- FPGAs.
   -- Xilinx HDL Libraries Guide Version 8.1i

   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 Code

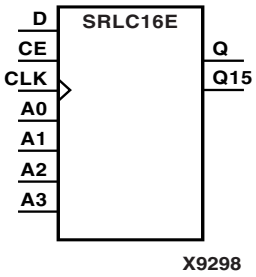
```
//   ROM256X1   : In order to incorporate this function into the design,
//   Verilog    : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration : (ROM256X1_inst) and/or the port declarations within the
//    code      : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  All inputs
//              : and outputs must be connect.

//  <-----Cut code below this line---->

   // ROM256X1: 256 x 1 Asynchronous Distributed (LUT) ROM
   // FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   ROM256X1 #(
      .INIT(256'h0000000000000000000000000000000000000000000000000000000000000000) // Contents of ROM
   ) ROM256X1_inst (
      .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
```

For More Information

Consult the Spartan-3E Data Sheet.

SRLC16E

Primitive: 16-Bit Shift Register Look-Up-Table (LUT) with Carry and Clock Enable



SRLC16E is a shift register look up table (LUT) with carry and clock enable. 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.

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, that value defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the Low-to-High clock (CLK) transition. When CE is High, during subsequent Low-to-High clock transitions, data is shifted 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.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

Inputs				Output	
Am	**CLK**	**CE**	**D**	**Q**	**Q15**
Am	X	0	X	Q(Am)	Q(15)
Am	X	1	X	Q(Am)	Q(15)
Am	↑	1	D	Q(Am - 1)	Q15
m= 0, 1, 2, 3					

Usage

This design element can be inferred or instantiated.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	16-Bit Hexa-decimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of content and output of shift register after configuration

VHDL Instantiation Template

```
--  SRLC16E    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (SRL16E_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
```

```
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- SRLC16E: 16-bit cascable shift register LUT with clock enable operating on posedge of clock
   -- FPGAs.
   -- Xilinx  HDL Libraries Guide version 8.1i

   SRLC16E_inst : SRLC16E
   generic map (
      INIT => X"0000")
   port map (
      Q => Q,       -- SRL data output
      Q15 => Q15,   -- Carry output (connect to next SRL)
      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 SRLC16E_inst instantiation
```

Verilog Instantiation Code

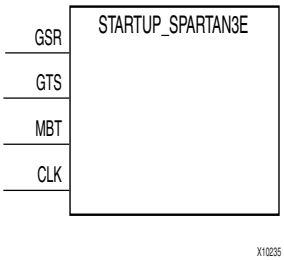
```
//  SRLC16E     : In order to incorporate this function into the design,
//  Verilog     : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration : (SRL16E_inst) and/or the port declarations within the
//    code      : parenthesis maybe changed to properly reference and
//              : connect this function to the design.  All inputs
//              : and outputs must be connected.

//  <-----Cut code below this line---->

   // SRLC16E: 16-bit cascable shift register LUT with clock enable operating on posedge of clock
   // FPGAs.
   // Xilinx HDL Libraries Guide Version 8.1i

   SRLC16E #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRLC16E_inst (
      .Q(Q),       // SRL data output
      .Q15(Q15),   // Carry output (connect to next SRL)
      .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 SRLC16E_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

STARTUP_SPARTAN3E

Primitive: Spartan-3E User Interface to the GSR, GTS, Configuration Startup Sequence and Multi-Boot Trigger Circuitry



The STARTUP_SPARTAN3E component allows the connection of ports, or user circuitry, to control certain dedicated circuitry and routes within the FPGA. Signals connected to the GSR port of this component can control the global set/reset (referred to as GSR) of the device. The GSR net connects to all registers in the device and places the registers into their initial value state. Connecting a signal to the GTS port connects that port to the dedicated route controlling the 3-state outputs of every pin in the device. Connecting a clock signal to the CLK input allows the startup sequence after configuration to be synchronized to a user defined clock. The MBT (Multi-Boot Trigger) pin allows the triggering of a new configuration.

Usage

The STARTUP_SPARTAN3E component must be instantiated to be incorporated into a design. Do not connect any input not needed for the design.

VHDL Instantiation Template

```
-- STARTUP_SPARTAN3E : In order to incorporate this function into the design,
--      VHDL        : the following instance declaration needs to be placed
--    instance      : in the architecture body of the design code.  The
--   declaration    : instance name (STARTUP_SPARTAN3E_inst) and/or the port declarations
--      code        : after the "=>" assignment maybe changed to properly
--                  : connect this function to the design.  Delete or comment
--                  : out inputs/outs that are not necessary.

--     Library      : In addition to adding the instance declaration, a use
--   declaration    : statement for the UNISIM.vcomponents library needs to be
--       for        : added before the entity declaration.  This library
--     Xilinx       : contains the component declarations for all Xilinx
--    primitives    : primitives and points to the models that are used
--                  : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- STARTUP_SPARTAN3E: Startup primitive for GSR, GTS, startup sequence
   --                   control and Multi-Boot Configuration. Spartan-3E
   -- Xilinx  HDL Libraries Guide version 8.1i

   STARTUP_SPARTAN3E_inst : STARTUP_SPARTAN3E
   port map (
      CLK => CLK,      -- Clock input for start-up sequence
      GSR => GSR_PORT, -- Global Set/Reset input (GSR cannot be used for the port name)
      GTS => GTS_PORT  -- Global 3-state input (GTS cannot be used for the port name)
      MBT => MBT -- Multi-Boot Trigger input
   );

   -- End of STARTUP_SPARTAN3E_inst instantiation
```

Verilog Instantiation Template

```
// STARTUP_SPARTAN3E : In order to incorporate this function into the design,
//     Verilog       : the following instance declaration needs to be placed
//    instance       : in the body of the design code.  The instance name
//   declaration     : (STARTUP_SPARTAN3E_inst) and/or the port declarations within the
//      code         : parenthesis maybe changed to properly reference and
```

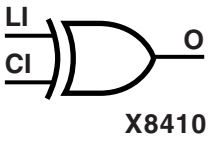
```
//                   : connect this function to the design.  Delete or comment
//                   : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // STARTUP_SPARTAN3E: Startup primitive for GSR, GTS, startup sequence control
   //                    and Multi-Boot Configuration Trigger. Spartan-3E
   // Xilinx HDL Libraries Guide Version 8.1i

   STARTUP_SPARTAN3E STARTUP_SPARTAN3E_inst (
      .CLK(CLK),       // Clock input for start-up sequence
      .GSR(GSR_PORT), // Global Set/Reset input (GSR can not be used as a port name)
      .GTS(GTS_PORT), // Global 3-state input (GTS can not be used as a port name)
      .MBT(MBT)        // Multi-Boot Trigger input
   );

   // End of STARTUP_SPARTAN3E_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet. Also see the Synthesis and Verification Design Guide for more information on using the GSR and GTS signals of the STARTUP_SPARTAN3E component.

XORCY

Primitive: XOR for Carry Logic with General Output



XORCY is a special XOR with general O output used for generating faster and smaller arithmetic functions.

Usage

Its O output is a general interconnect. See also "XORCY_D"and "XORCY_L".

VHDL Instantiation Code

```
--    XORCY    : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (XORCY_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- XORCY: Carry-Chain XOR-gate with general output
   -- Xilinx  HDL Libraries Guide version 8.1i

   XORCY_inst : XORCY
   port map (
      O => O,   -- XOR output signal
      CI => CI, -- Carry input signal
      LI => LI  -- LUT4 input signal
   );

   -- End of XORCY_inst instantiation
```

Verilog Instantiation Code

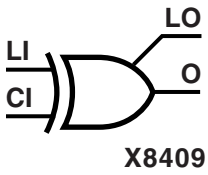
```
//    XORCY    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (XORCY_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // XORCY: Carry-Chain XOR-gate with general output
   //        For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   XORCY XORCY_inst (
      .O(O),   // XOR output signal
      .CI(CI), // Carry input signal
      .LI(LI)  // LUT4 input signal
   );

   // End of XORCY_inst instantiation
```

For More Information

Consult the Spartan-3E Data Sheet.

XORCY_D

Primitive: XOR for Carry Logic with Dual Output



XORCY_D is a special XOR used for generating faster and smaller arithmetic functions.

Usage

XORCY_D has two, functionally identical outputs: O and LO. The O output is a general interconnect. The LO output connects to another output within the same CLB slice.

VHDL Instantiation Code

```
--   XORCY_D   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (XORCY_D_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- XORCY_D: Carry-Chain XOR-gate with local and general outputs
   -- Xilinx HDL Libraries Guide Version 8.1i

   XORCY_D_inst : XORCY_D
   port map (
      LO => LO, -- XOR local output signal
      O => O,   -- XOR general output signal
      CI => CI, -- Carry input signal
      LI => LI  -- LUT4 input signal
   );

   -- End of XORCY_D_inst instantiation
```

Verilog Instantiation Code

```
//   XORCY_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (XORCY_D_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // XORCY_D: Carry-Chain XOR-gate with local and general outputs
   //          For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   XORCY_D XORCY_D_inst (
      .LO(LO), // XOR local output signal
      .O(O),   // XOR general output signal
```

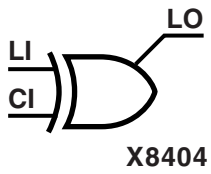
```
   .CI(CI), // Carry input signal
   .LI(LI)  // LUT4 input signal
);

// End of XORCY_D_inst instantiation.
```

For More Information

Consult the Spartan-3E Data Sheet.

XORCY_L

Primitive: XOR for Carry Logic with Local Output



XORCY_L is a special XOR with local LO output used for generating faster and smaller arithmetic functions.

Usage

The LO output connects to another output within the same CLB slice.

VHDL Instantiation Code

```
--   XORCY_L   : In order to incorporate this function into the design,
--    VHDL     : the following instance declaration needs to be placed
--  instance   : in the architecture body of the design code.  The
-- declaration : instance name (XORCY_L_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : reference and connect this function to the design.
--             : All inputs and outputs must be connected.

--   Library   : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--     for     : added before the entity declaration.  This library
--   Xilinx    : contains the component declarations for all Xilinx
--  primitives : primitives and points to the models that are used
--             : for simulation.

--  Copy the following two statements and paste them before the
--  Entity declaration, unless they already exist.

Library UNISIM;
use UNISIM.vcomponents.all;

--  <-----Cut code below this line and paste into the architecture body---->

   -- XORCY_L: Carry-Chain XOR-gate with local  => direct-connect ouput
   -- Xilinx HDL Libraries Guide Version 8.1i

   XORCY_L_inst : XORCY_L
   port map (
      LO => LO, -- XOR local output signal
      CI => CI, -- Carry input signal
      LI => LI  -- LUT4 input signal
   );

   -- End of XORCY_L_inst instantiation
```

Verilog Instantiation Code

```
//   XORCY_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (XORCY_L_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

//  <-----Cut code below this line---->

   // XORCY_L: Carry-Chain XOR-gate with local (direct-connect) ouput
   //          For use with All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   XORCY_L XORCY_L_inst (
      .LO(LO), // XOR local output signal
      .CI(CI), // Carry input signal
      .LI(LI)  // LUT4 input signal
   );
```

```
// End of XORCY_L_inst instantiation.
```

For More Information

Consult the Spartan-3E Data Sheet.