Virtex-4 Libraries Guide for HDL Designs

ISE 8.1i

XILINX®

About this Guide

The *Virtex-4™ 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 *Virtex-4™ 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 Virtex-4 primitives.
- A listing of the primitives and macros that are supported under the Virtex-4 architecture, organized by functional categories.
- Individual sections for each of the primitive design elements, including VHDL and Verilog instantiation and inference code examples.

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 design elements that comprise the Xilinx Unified Libraries for the Virtex-4 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 different device architectures. New functional elements are assembled with each release of development system software. In addition to a comprehensive Unified Library containing all design elements, beginning in 2003, Xilinx developed a separate library for each architecture. This Virtex-4 guide is one in a series of architecture-specific libraries.

This guide describes the primitive elements available for Xilinx Virtex-4 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 may 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

Functional Categories

This section categorizes, by function, the design elements that are described in detail later in this guide. The design elements are listed in alphanumeric order under each functional category.

Arithmetic Functions

Design Element	Description
DSP48	Primitive: 18x18 Signed Multiplier Followed by a Three-Input Adder with Optional Pipeline Registers

Clock Components

Design Element	Description
BUFG	Primitive: Global Clock Buffer
BUFGCE	Primitive: Global Clock MUX with Clock Enable and Output State 0
BUFGCE_1	Primitive: Global Clock MUX Buffer with Clock Enable and Output State 1
BUFGCTRL	Primitive: Global Clock MUX Buffer
BUFGMUX	Primitive: Global Clock MUX Buffer with Output State 0
BUFGMUX_1	Primitive: Global Clock MUX with Output State 1
BUFGMUX_VIRTEX4	Primitive: Global Clock MUX Buffer
BUFIO	Primitive: Local Clock Buffer for I/O
BUFR	Primitive: Local Clock Buffer for I/O and CLB
DCM_ADV	Primitive: Digital Clock Manager with Advanced Features
DCM_BASE	Primitive: Digital Clock Manager with Basic Features
DCM_PS	Primitive: Digital Clock Manager with Basic and Phase-Shift Features
PMCD	Primitive: Phase-Matched Clock Divider

Config/BSCAN Components

Design Element	Description
BSCAN_VIRTEX4	Primitive: Provides Access to the BSCAN Sites on Virtex-4 Devices
CAPTURE_VIRTEX4	Primitive: Virtex-4 Boundary Scan Logic Control Circuit
FRAME_ECC_VIRTEX4	Primitive: Reads a Single, Virtex-4 Configuration Frame and Computes a Hamming, Single-Error Correction, Double-Error Detection "Syndrome"
ICAP_VIRTEX4	Primitive: Virtex-4 Internal Configuration Access Port

STARTUP_VIRTEX4	Primitive: Virtex-4 User Interface to Configuration Clock, Global Reset, Global 3-State Controls, and Other Configuration Signals
USR_ACCESS_VIRTEX4	Primitive: 32-Bit Register with a 32-Bit DATA Bus and a DATAVALID Port

Gigabit Transceivers

Design Element	Description
GT11_CUSTOM	Primitive: RocketIO MGTs with 622 Mb/s to 11.1 Gb/s data rates, 8 to 24 transceivers per FPGA, and 2.5 GHz – 5.55 GHz VCO, less than 1ns RMS jitter
GT11_DUAL	Primitive: RocketIO MGT Tile (contains 2 GT11_CUSTOM) with 622 Mb/s to 11.1 Gb/s data rates, 8 to 24 transceivers per FPGA, and 2.5 GHz – 5.55 GHz VCO, less than 1ns RMS jitter
GT11CLK	Primitive: A MUX That Can Select Fom Differential Package Input Clock, refclk From the Fabric, or rxbclk to Drive the Two Vertical Reference Clock Buses for the Column of MGTs
GT11CLK_MGT	Primitive: Allows Differential Package Input to Drive the Two Vertical Reference Clock Buses for the Column of MGTs

I/O Components

Design Element	Description
BUFIO	Primitive: Local Clock Buffer for I/O
DCIRESET	Primitive: DCI State Machine Reset (After Configuration Has Been Completed)
IBUF	Primitive: Single-Ended Input Buffer with Selectable I/O Standard and Capacitance
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
IDDR	Primitive: A Dedicated Input Register to Receive External Dual Data Rate (DDR) Signals into Virtex-4 FPGAs
IDELAY	Primitive: Dedicated input variable-tap delay chain
IDELAYCTRL	Primitive: IDELAY tap delay value control
IOBUF	Primitive: Bi-Directional Buffer with Selectable I/0 Interface (multiple primitives)
IOBUFDS	Primitive: 3-State Differential Signaling I/O Buffer with Active Low Output Enable
ISERDES	Primitive: Dedicated I/O Buffer Input Deserializer
KEEPER	Primitive: KEEPER Symbol
OBUF	Primitive: Single-ended Output Buffer
OBUFT	Primitive : 3-State Output Buffer with Active Low Output Enable and with Selectable I/O Interface
OBUFDS	Primitive: Differential Signaling Output Buffer with Selectable I/O Interface
OBUFTDS	Primitive: 3-State Output Buffer with Differential Signaling, Active-Low Output Enable, and Selectable I/O Interface
ODDR	Primitive: A dedicated output registers to transmit dual data rate (DDR) signals from Virtex-4 FPGAs
OSERDES	Primitive: Provides a way for the user to easily implement source synchronous interface by using the OSERDES module
PULLDOWN	Primitive: Resistor to GND for Input Pads
PULLUP	Primitive: Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs

Processor Components

Design Element	Description
EMAC	Primitive: Fully integrated 10/100/1000 Mb/s Ethernet Media Access Controller (Ethernet MAC)
PPC405_ADV	Primitive: Primitive for the Power PC Core

RAM/ROM

Design Element	Description
FIFO16	Primitive: Virtex-4 Block RAM based built-in FIFO
RAM16X1D	Primitive: 16-Deep by 1-Wide Static Dual Port Synchronous RAM
RAM16X1S	Primitive: 16-Deep by 1-Wide Static Synchronous RAM
RAM32X1S	Primitive: 32-Deep by 1-Wide Static Synchronous RAM
RAM64X1S	Primitive: 64-Deep by 1-Wide Static Synchronous RAM
RAMB16	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
RAMB32_S64_ECC	Primitive: 512 Deep by 64-Bit Wide Synchronous, Two-Port, Block RAM with Built-In Error Correction
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
SRL16	Primitive: 16-Bit Shift Register Look-Up Table (LUT)
SRL16_1	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Negative-Edge Clock
SRL16E	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Clock Enable
SRL16E_1	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Negative-Edge Clock and Clock Enable
SRLC16	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Carry
SRLC16_1	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Carry and Negative-Edge Clock
SRLC16E	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Carry and Clock Enable
SRLC16E_1	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Negative-Edge Clock and Clock Enable

Slice/CLB Primitives

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

Design Element	Description
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 Lookup Table Multiplexer with General Output
MUXF5_D	Primitive: 2-to-1 Lookup Table Multiplexer with Dual Output
MUXF5_L	Primitive: 2-to-1 Lookup Table Multiplexer with Local Output
MUXF6	Primitive: 2-to-1 Lookup Table Multiplexer with General Output
MUXF6_D	Primitive: 2-to-1 Lookup Table Multiplexer with Dual Output
MUXF6_L	Primitive: 2-to-1 Lookup Table Multiplexer with Local Output
MUXF7	Primitive: 2-to-1 Lookup Table Multiplexer with General Output
MUXF7_D	Primitive: 2-to-1 Lookup Table Multiplexer with Dual Output
MUXF7_L	Primitive: 2-to-1 Lookup Table Multiplexer with Local Output
MUXF8	Primitive: 2-to-1 Lookup Table Multiplexer with General Output
MUXF8_D	Primitive: 2-to-1 Lookup Table Multiplexer with Dual Output
MUXF8_L	Primitive: 2-to-1 Lookup 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 Virtex-4 Design Elements

The remaining sections in this book describe each primitive design element that can be used under the Virtex-4 architecture.

The design elements are organized in alphanumeric order, with all numeric suffixes in ascending order. For example, FDR precedes FDRS, and ADD4 precedes ADD8, which precedes ADD16.

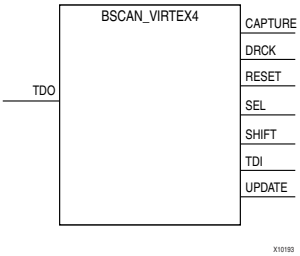
The following information is provided for each library element, where applicable:

- 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 *Virtex-4 Libraries Guide for Designers Using Schematics*.

BSCAN_VIRTEX4

Primitive: Provides Access to the BSCAN Sites on Virtex-4 Devices



When the JTAG USER1/2/3/4 instruction is loaded, BSCAN_VIRTEX4 allows users to monitor dedicated JTAG pins TCK, TMS, and TDI. Users are also granted the ability to drive the TDO pin with user-specified data.

Name	Type	Width	Function
CAPTURE	Output	1	Active upon the loading of the USER instruction. Asserts High when the JTAG TAP controller is in the CAPTURE-DR state.
DRCK	Output	1	A mirror of the TCK pin when the JTAG USER instruction is loaded and the JTAG TAP controller is in the SHIFT-DR state.
RESET	Output	1	Active upon the loading of the USER instruction. It asserts High when the JTAG TAP controller is in the TEST-LOGIC-RESET state.
SEL	Output	1	Indicates when the USER1 instruction has been loaded into the JTAG Instruction Register. Becomes active in the UPDATE-IR state, and stays active until a new instruction is loaded.
SHIFT	Output	1	Active upon the loading of the USER instruction. It asserts High when the JTAG TAP controller is in the SHIFT-DR state.
TDI	Output	1	A mirror of the TDI pin.
UPDATE	Output	1	Active upon the loading of the USER1 or USER2 instruction. It asserts High when the JTAG TAP controller is in the UPDATE-DR state.
TDO	Input	1	Active upon the loading of the USER1 or USER2 instruction. External JTAG TDO pin will reflect data input to the macro's TDO1 pin.

Usage

Virtex-4 has four available BSCAN_VIRTEX4 primitives. Use the appropriate attributes to target the desired primitive. To access these primitives, the JTAG USER instruction must be loaded.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
JTAG_CHAIN	INTEGER	1, 2, 3, or 4	1	Used to set the BSCAN site in the device.

VHDL Instantiation Template

```
--  BSCAN_VIRETX4  : 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 instance name
--   declaration   : (BSCAN_VIRTEX4_inst) and/or the port declarations after the
--      code       : "=>" assignment maybe changed to properly reference and 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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- BSCAN_VIRETX4: Boundary Scan primitve for connecting internal logic to
   --                JTAG interface. Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   BSCAN_VIRTEX4_inst : BSCAN_VIRTEX4
   generic map (
      JTAG_CHAIN => 1) -- Value to set BSCAN site of device. Possible values: (1,2,3 or 4)
   port map (
      CAPTURE => CAPTURE, -- CAPTURE output from TAP controller
      DRCK => DRCK,       -- Data register output for USER functions
      RESET => RESET,     -- Reset output from TAP controller
      SEL => SEL,         -- USER active output
      SHIFT => SHIFT,     -- SHIFT output from TAP controller
      TDI => TDI,         -- TDI output from TAP controller
      UPDATE => UPDATE,   -- UPDATE output from TAP controller
      TDO => TDO          -- Data input for USER function
   );

   -- End of BSCAN_VIRETX4_inst instantiation
```

Verilog Instantiation Template

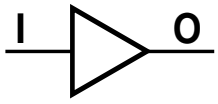
```
//  BSCAN_VIRETX4  : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (BSCAN_VIRTEX4_inst) and/or the port declarations within 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_VIRETX4: Boundary Scan primitive for connecting internal logic to
   //                JTAG interface. Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   BSCAN_VIRETX4 #(
      .JTAG_CHAIN(1)     // Possible values: 1, 2, 3, or 4
   ) BSCAN_VIRETX4_inst (
      .CAPTURE(CAPTURE), // CAPTURE output from TAP controller
      .DRCK(DRCK),       // Data register output for USER function
      .RESET(RESET),     // Reset output from TAP controller
      .SEL(SEL),         // USER active output
      .SHIFT(SHIFT),     // SHIFT output from TAP controller
      .TDI(TDI),         // TDI output from TAP controller
      .UPDATE(UPDATE),   // UPDATE output from TAP controller
      .TDO(TDO)          // Data input for USER function
   );

   // End of BSCAN_VIRETX4_inst instantiation
```

For More Information

Consult the *Virtex-4 Configuration Guide.*

BUFCF

Primitive: Fast Connect Buffer



X9444

BUFCF is a fast connect buffer used to communicate to the software tools the Slice packing of logic. This buffer does not indicate any functionality for the design, but it can be used to tell the software to place both the sourcing logic to the buffer and the destination logic in the same Slice in order to minimize the routing delays for that path.

Usage

The BUFCF must be instantiated. To connect this element to the design, connect the input of the buffer to the output of Slice logic (such as a LUT4) and connect the output to another piece of Slice logic (such as another LUT4). This will indicate to the tools that both components connected to the logic should be placed into the same Slice. It is generally suggested to use this component with instantiated logic since connecting it to inferred logic may disrupt the optimization opportunities for the tools.

VHDL Instantiation Template

```
--     BUFCF       : 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 (BUFCF_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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- BUFCF: Fast connect buffer used to connect the outputs of the LUTs
   --        and some dedicated logic directly to the input of another LUT.
   --        For use with all FPGAs.
   -- Xilinx HDL Langauge Template version 8.1i

   BUFCF_inst: BUFCF (
   port map (
      O => O, -- Connect to the output of a LUT
      I => I  -- Connect to the input of a LUT
   );

   -- End of BUFCF_inst instantiation
```

Verilog Instantiation Template

```
//    BUFCF     : 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 : (BUFCF_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---->
```

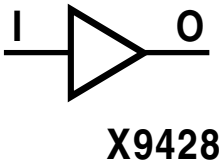
```
// BUFCF: Fast connect buffer used to connect the outputs of the LUTs
//        and some dedicated logic directly to the input of another LUT.
//        For use with all FPGAs.
// Xilinx HDL Libraries Guide Version 8.1i

BUFCF BUFCF_inst (
   .O(O), // Connect to the output of a LUT
   .I(I)  // Connect to the input of a LUT
);

// End of BUFCF_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

BUFG

Primitive: Global Clock Buffer



BUFG, an architecture-independent global buffer, distributes high fan-out clock signals throughout a PLD device. The Xilinx implementation software converts each BUFG to an appropriate type of global buffer for the target PLD device.

Usage

This design element is supported for both schematics 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 utilized. The BUFGP contains both a BUFG and an IBUFG.

To use a BUFG in a schematic, connect the input of the BUFG symbol to the clock source. The clock source can be an external PAD symbol, an IBUF symbol, or internal logic. For a negative-edge clock input, insert an INV (inverter) symbol between the BUFG output and the clock input. The inversion is implemented at the Configurable Logic Block (CLB) or Input/Output Block (IOB) clock pin.

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 will be used
--             : for simulation.

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

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.
```

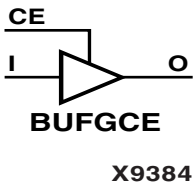
```
//  <-----Cut code below this line---->

   // 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 *Virtex-4 User Guide*.

BUFGCE

Primitive: Global Clock Buffer with Clock Enable and Output State 0



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 schematics and 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 will be used
--              : for simulation.

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

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)
   --         Virtex-II/II-Pro, Spartan-3/3E
   -- 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.
```

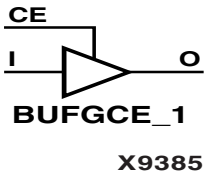
```
//  <-----Cut code below this line---->

   // BUFGCE: Global Clock Buffer with Clock Enable (active high)
   //         Virtex-II/II-Pro/4, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

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 will be used
--             : for simulation.

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

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)
   --         Virtex-II/II-Pro, Spartan-3/3E
   -- 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
```

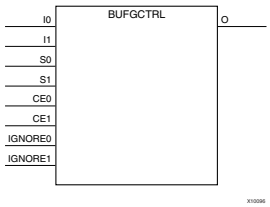
```
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

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

   // BUFGCE_1: Global Clock Buffer with Clock Enable (active low)
   //         Virtex-II/II-Pro, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

BUFGCTRL

Primitive: Global Clock Mux Buffer



BUFGCTRL primitive is a Virtex-4 global clock buffer that is designed as a synchronous/asynchronous "glitch free" 2:1 multiplexer with two clock inputs. Unlike global clock buffers that are found in previous generations of FPGAs, the BUFGCTRL is designed with additional control pins to provide a wider range of functionality and more robust input switching. BUFGCTRL is *not* limited to clocking applications.

BUFGCTRL Ports (Detailed Description)

O – Clock Output Pin

The O pin represents the clock output pin.

I0 – Clock Input Pin

I1 - Clock Input Pin

The I pin represents the clock input pin.

CE0 – CE1 – Clock Enable Pins

The CE pins represent the clock enable pin for each clock input. It is also used to select the clock inputs. When using the CE pin as input select, there is a setup/hold time requirement. Failure to meet this requirement may result in a clock glitch.

S0 – S1 – Clock Select Pin

The S pins represent the clock select pin for each clock input. When using the S pin select, there is a setup/hold time requirement. Unlike CE pins, failure to meet this requirement will not result in a clock glitch. However, it may cause the output clock to appear one clock cycle later.

IGNORE0 – IGNORE1 – Ignore Pin

The IGNORE pin is used whenever you want to bypass the switching algorithm executed by the BUFGCTRL.

Name	Type	Width	Function
O	Output	1	Clock Output
I0 - I1	Input	1 (each)	Clock Input
CE0 – CE1	Input	1 (each)	Clock Enable Input
S0 – S1	Input	1 (each)	Clock Select Input
IGNORE0 – IGNORE1	Input	1 (each)	Clock Ignore Input

Usage

In order to properly select a BUFGCTRL input, you must assert both the S and CE pins of the desired input. Failure to do so may cause the output to not switch with the desired input or output signal toggling.

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

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT_OUT	INTEGER	0 or 1	0	Initializes the BUFGCTRL output to the specified value after configuration.
PRESELECT_I0	BOOLEAN	FALSE, TRUE	FALSE	If TRUE, BUFGCTRL output will use I0 input after configuration.
PRESELECT_I1	BOOLEAN	FALSE, TRUE	FALSE	If TRUE, BUFGCTRL output will use I1 input after configuration.

Note: Both PRESELECT attributes might not be TRUE at the same time.

VHDL Instantiation Template

```
--     BUFGCTRL    : In order to incorporate this function into the design,
--       VHDL      : the following instance declaration needs to be placed
--     instance    : in the body of the design code.  The instance name
--   declaration   : (BUFGCTRL_inst) and/or the port declarations after the
--       code      : "=>" assignemnt 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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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


   -- BUFGCTRL: Advanced Clock Primtive
   --           Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   BUFGCTRL_inst : BUFGCTRL
   generic map (
      INIT_OUT => 0,         -- Inital value of 0 or 1 after configuration
      PRESELECT_I0 => FALSE, -- TRUE/FALSE set the I0 input after configuration
      PRESELECT_I1 => FALSE) -- TRUE/FALSE set the I1 input after configuration
   port map (
      O => O,              -- Clock MUX output
      CE0 => CE0,          -- Clock enable0 input
      CE1 => CE1,          -- Clock enable1 input
      I0 => I0,            -- Clock0 input
      I1 => I1,            -- Clock1 input
      IGNORE0 => IGNORE0,  -- Ignore clock select0 input
      IGNORE1 => IGNORE1,  -- Ignore clock select1 input
      S0 => S0,            -- Clock select0 input
      S1 => S1             -- Clock select1 input
   );

   -- End of BUFGCTRL_inst instantiation
```

Verilog Instantiation Template

```
//  BUFGCTRL  : 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 : (BUFGCTRL_inst) and/or the port declarations within the
```

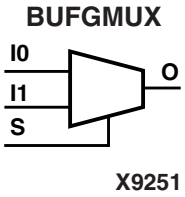
```
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connect.

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

   // BUFGCTRL: Advanced Clock MUX Primtive
   //           Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFGCTRL #(
      .INIT_OUT(0),  // Inital value of 0 or 1 after configuration
      .PRESELECT_I0("FALSE"), // "TRUE" or "FALSE" set the I0 input after configuration
      .PRESELECT_I1("FALSE")  // "TRUE" or "FALSE" set the I1 input after configuration
   ) BUFGCTRL_inst (
      .O(O),     // 1-bit output
      .CE0(CE0), // 1-bit clock enable 0
      .CE1(CE1), // 1-bit clock enable 1
      .I0(I0),   // 1-bit clock 0 input
      .I1(I1),   // 1-bit clock 1 input
      .IGNORE0(IGNORE0), // 1-bit ignore 0 input
      .IGNORE1(IGNORE1), // 1-bit ignore 1 input
      .S0(S0),   // 1-bit select 0 input
      .S1(S1)    // 1-bit select 1 input
   );

   // End of BUFGCTRL_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

BUFGMUX

Primitive: Global Clock MUX Buffer with Output State 0



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 which state the output assumes when it switches between clocks in response to a change in its select input. BUGFMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Note: BUFGMUX guarantees that when S is toggled, the state of the output will remain 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 will be used
--             : for simulation.

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

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
   --          Virtex-II/II-Pro, Spartan-3/3E
   -- 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

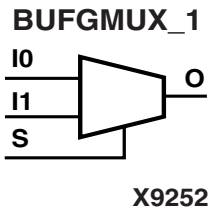
```
//   BUFGMUX   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (BUFGMUX_inst) and/or the port declarations within the
//    code     : 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
   //          Virtex-II/II-Pro/4, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

BUFGMUX_1

Primitive: Global Clock MUX Buffer with Output State 1



X9252

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 which state the output assumes when it switches between clocks in response to a change in its select 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 will be used
--             : for simulation.

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

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)
   --            Virtex-II/II-Pro, Spartan-3/3E
   -- 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

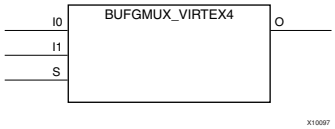
```
//  BUFGMUX_1  : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (BUFGMUX_1_inst) and/or the port declarations within the
//    code     : 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)
   //            Virtex-II/II-Pro, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

BUFGMUX_VIRTEX4

Primitive: Global Clock MUX Buffer



BUFGMUX_VIRTEX4 is a global clock buffer with two clock inputs, one clock output, and a select line. This primitive is based on BUFGCTRL, with some pins connected to logic High or Low.

This element uses the S pins as select pins. S can switch anytime without causing a glitch. The Setup/Hold time on S is for determining whether the output wil pass an extra pulse of the previously selected clock before switching to the new clock. If S changes prior to the setup time T_{BCCCK_S}, and before I/O transitions from High to Low, then the output will not pass an extra pulse of I/O. If S changes following the hold time for S, then the output will pass an extra pulse, but it will not glitch. In any case the output will change to the new clock within three clock cycles of the slower clock.

The Setup/Hold requirements for S0 and S1 are with respect to the falling clock edge (assuming INIT_OUT = 0), not the rising edge, as for CE0 and CE1.

Switching conditions for BUFGMUX_VIRTEX4 are the same as the S pin of BUFGCTRL.

BUFGMUX_VIRTEX4 Ports

O – Clock Output Pin

The O pin represents the clock output pin.

I0 – Clock Input Pin

I1 - Clock Input Pin

The I pin represents the clock input pin.

Clock Select Pin

The S pin represents the clock select pin.

The port list and definitions for this element are as follows:

Name	Type	Width	Function
O	Output	1	Clock Output
I1 - I0	Input	1	Clock Input
S0 – S1	Input	1	Clock Select Input

VHDL Instantiation Template

```
-- BUFGMUX_VIRTEX4  : In order to incorporate this function into the design,
--      VHDL        : the following instance declaration needs to be placed
--    instance      : in the body of the design code.  The instance name
--   declaration    : (BUFGMUX_VIRTEX4_inst) and/or the port declarations after the
--      code        : "=>" assignment maybe changed to properly reference and
--                  : connect this function to the design.  All inputs
--                  : and outputs must be connect.

--     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 will be used
--                  : for simulation.
```

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- BUFGMUX_VIRTEX4: Global Clock Buffer 2-to-1 MUX
   --                  Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   BUFGMUX_VIRTEX4_inst : BUFGMUX_VIRTEX4
   generic map (
      INIT_OUT => 0,         -- Inital value of 0 or 1 after configuration
      PRESELECT_I0 => FALSE, -- TRUE/FALSE set the I0 input after configuration
      PRESELECT_I1 => FALSE) -- TRUE/FALSE set the I1 input after configuration
   port map (
      O => O,     -- Clock MUX output
      I0 => I0,   -- Clock0 input
      I1 => I1,   -- Clock1 input
      S => S      -- Clock select input
   );

   -- End of BUFGMUX_VIRTEX4_inst instantiation
```

Verilog Instantiation Template

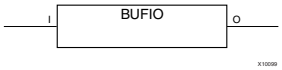
```
// BUFGMUX_VIRTEX4 : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (BUFGMUX_VIRTEX4_inst) and/or the port declarations within 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_VIRTEX4: Global Clock Buffer 2-to-1 MUX
   //                  Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFGMUX_VIRTEX4 BUFGMUX_VIRTEX4_inst (
      .O(O),     // Clock MUX output
      .I0(I0),   // Clock0 input
      .I1(I1),   // Clock1 input
      .S(S)      // Clock select input
   );

   // End of BUFGMUX_VIRTEX4_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

BUFIO

Primitive: Local Clock Buffer for I/O



The BUFIO is a clock buffer available in Virtex-4 devices. It is simply a clock-in, clock-out buffer. The BUFIO drives a dedicated clock net within the I/O column, independent of the global clock resources. Thus, BUFIOs are ideally suited for source-synchronous data capture (forwarded/receiver clock distribution). BUFIOs can only be driven by clock capable I/Os located in the same clock region. They drive the two adjacent I/O clock nets (for a total of up to three clock regions), as well as the regional clock buffers (BUFR). BUFIOs cannot drive logic resources (CLB, block RAM, etc.) because the I/O clock network only reaches the I/O column.

BUFIO Ports (Detailed Description)

Name	Type	Width	Function
O	Output	1	Clock output port
I	Input	1	Clock input port

Usage

BUFIOs work in conjunction with I/O capable clocks, and represent an ideal solution for source synchronous applications that require clock recovery.

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

VHDL Instantiation Template

```
--      BUFIO    : In order to incorporate this function into the design,
--      VHDL     : the following instance declaration needs to be placed
--    instance   : in the body of the design code.  The instance name
--   declaration : (BUFIO_inst) and/or the port declarations after the
--      code     : "=>" 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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- BUFIO: Clock in, clock out buffer
   --        Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

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

   -- End of BUFIO_inst instantiation
```

Verilog Instantiation Template

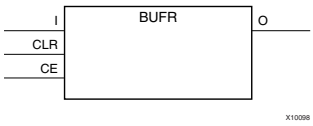
```
//    BUFIO    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (BUFIO_inst) and/or the port declarations within the
//    code     : 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---->

   // BUFIO: Local Clock Buffer
   //       Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFIO BUFIO_inst (
      .O(O),      // Clock buffer output
      .I(I)       // Clock buffer input
   );

   // End of BUFIO_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

BUFR

Primitive: Regional Clock Buffer for I/O and Logic Resources



The BUFR is a clock buffer available in Virtex-4 devices. BUFRs drive clock signals to a dedicated clock net within a clock region, independent from the global clock tree. Each BUFR can drive the two regional clock nets in the region in which it is located, and the two clock nets in the adjacent clock regions (up to three clock regions). Unlike BUFIOs, BUFRs can drive the I/O logic *and* logic resources (CLB, block RAM, etc.) in the existing and adjacent clock regions. BUFRs can be driven by either the output from BUFIOs or local interconnect. In addition, BUFRs are capable of generating divided clock outputs with respect to the clock input. The divide values are an integer between one and eight. BUFRs are ideal for source-synchronous applications requiring clock domain crossing or serial-to-parallel conversion. There are two BUFRs in a typical clock region (two regional clock networks). The center column does not have BUFRs.

BUFR Ports (Detailed Description)

O – Clock Output Port

This port drives the clock tracks in the clock region of the BUFR and the two adjacent clock regions. This port drives FPGA fabric and IOBs.

CE – Clock Enable Port

CLR – Counter Reset for Divided Clock Output

When asserted HIGH, this port resets the counter used to produce the divided clock output.

I – Clock Input Port

This port is the clock source port for BUFR. It may be driven by BUFIO output or local interconnect.

The port list and definitions for this element are as follows:

Name	Type	Width	Function
O	Output	1	Clock output port
CE	Input	1	Clock enable port. Cannot be used in BYPASS mode.
CLR	Input	1	Asynchronous clear for the divide logic, and sets the output Low. Cannot be used in BYPASS mode.
I	Input	1	Clock input port

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BUFR_DIVIDE	STRING	"BYPASS", "1", "2", "3", "4", "5", "6", "7", "8	"BYPASS"	Defines whether the output clock is a divided version of input clock.

Usage

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

VHDL Instantiation Template

```
--      BUFR     : In order to incorporate this function into the design,
--      VHDL     : the following instance declaration needs to be placed
--    instance   : in the body of the design code.  The instance name
--   declaration : (BUFR_inst) and/or the port declarations after the
--      code     : "=>" 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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- BUFR: Regional (Local) Clock Buffer /w Enable, Clear and Division Capabilities
   --       Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   BUFR_inst : BUFR
   generic map (
      BUFR_DIVIDE => "BYPASS") -- "BYPASS", "1", "2", "3", "4", "5", "6", "7", "8",
   port map (
      O => O,     -- Clock buffer output
      CE => CE,   -- Clock enable input
      CLR => CLR, -- Clock buffer reset input
      I => I      -- Clock buffer input
   );

   -- End of BUFR_inst instantiation
```

Verilog Instantiation Template

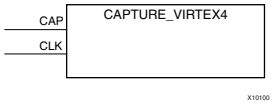
```
//    BUFR     : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (BUFR_inst) and/or the port declarations within the
//    code     : 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---->

   // BUFR: Regioanl Clock Buffer /w Enable, Clear and Division Capabilities
   //       Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   BUFR #(
      .BUFR_DIVIDE("BYPASS") // "BYPASS", "1", "2", "3", "4", "5", "6", "7", "8"
   ) BUFR_inst (
      .O(O),     // Clock buffer output
      .CE(CE),   // Clock enable input
      .CLR(CLR), // Clock buffer reset input
      .I(I)      // Clock buffer input
   );

   // End of BUFR_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

CAPTURE_VIRTEX4

Primitive: Virtex-4 Boundary Scan Logic Control Circuit



CAPTURE_VIRTEX4 provides user control over when to capture register (flip-flop and latch) information for readback. Virtex-4 devices provide the readback function through dedicated configuration port instructions.

CAPTURE_VIRTEX4 Ports (Detailed Description)

CAP – Input

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).

CLK – Input

Clock input pin

The port list and definitions for this element are as follows:

Name	Type	Width	Function
CAP	Input	1	Capture trigger pin
CLK	Input	1	Clock input pin

Usage

Using the CAPTURE_VIRTEX4 primitive is optional. Without this primitive, readback of the DFFs will be the initial set/preset value instead of the current value that DFFs hold.

Virtex-4 devices allow for capturing register (flip-flop and latch) states only. LUTRAM, SRL, and block RAM always have their current states readback once the GLUTMASK bit is disabled. Refer to the *Virtex-4 Configuration User Guide* for more information about readback.

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

Available Attributes

Name	Description	Possible Values
ONESHOT	Limits the readback operation to a single data capture	TRUE (default), FALSE

VHDL Instantiation Template

```
-- CAPTURE_VIRTEX4  : In order to incorporate this function into the design,
--      VHDL        : the following instance declaration needs to be placed
--    instance      : in the body of the design code.  The instance name
--   declaration    : (CAPTURE_VIRTEX4_inst) and/or the port declarations after the
--      code        : "=>" assignment maybe changed to properly reference and
--                  : connect this function to the design.  Delete or comment
--                  : out inputs/outputs 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 will be used
```

```
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- CAPTURE_VIRTEX4: Register State Capture for Bitstream Readback
   --                  Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   CAPTURE_VIRTEX4_inst : CAPTURE_VIRTEX4
   generic map (
      ONESHOT => TRUE) -- TRUE or FALSE
   port map (
      CAP => CAP,     -- Capture input
      CLK => CLK      -- Clock input
   );
   -- End of CAPTURE_VIRTEX4_inst instantiation
```

Verilog Template

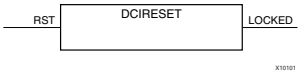
```
// CAPTURE_VIRTEX4 : In order to incorporate this function into the design,
//     Verilog      : the following instance declaration needs to be placed
//    instance      : in the body of the design code.  The instance name
//   declaration    : (CAPTURE_VIRTEX4_inst) and/or the port declarations 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_VIRTEX4: Register State Capture for Bitstream Readback
   //                  Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   CAPTURE_VIRTEX4 #(
      .ONESHOT("TRUE") // "TRUE" or "FALSE"
   ) CAPTURE_VIRTEX4_inst (
      .CAP(CAP),       // Capture input
      .CLK(CLK)        // Clock input
   );

   // End of CAPTURE_VIRTEX4_inst instantiation
```

For More Information

Consult the *Virtex-4 Configuration Guide*.

DCIRESET

Primitive: DCI State Machine Reset (After Configuration Has Been Completed)



The port list and definitions for this primitive are as follows:

Name	Type	Width	Function
RST	Input	1	Invokes the DCI state machine to start from initial state
LOCKED	Output	1	Indicates that DCI state machine has achieved a stable state after reset

Usage

The DCIRESET primitive is used to reset the DCI state machine after configuration has been completed. This design element is supported for schematics and instantiations, but not for inference.

VHDL Template

```
--    DCIRESET     : 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 (DCIRESET_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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- DCIRESET: DCI reset component
   --           Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   DCIRESET_inst : DCIRESET
   port map (
      LOCKED => LOCKED,     -- DCIRESET LOCK status output
      RST => RST            -- DCIRESET asynchronous reset input
   );

   -- End of DCIRESET_inst instantiation
```

Verilog Template

```
//    DCIRESET     : 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   : (DCIRESET_inst) and/or the port declarations within the
//      code       : parenthesis maybe changed to properly reference and
//                 : connect this function to the design.

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

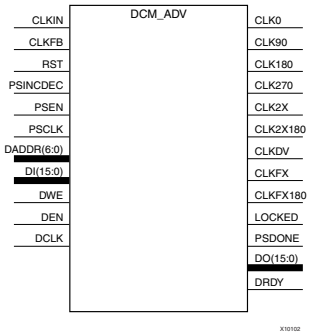
```
// DCIRESET: Digital Controlled Impedance (DCI) Reset Component
//           Virtex-4
// Xilinx HDL Libraries Guide Version 8.1i

DCIRESET DCIRESET_inst (
   .LOCKED(LOCKED),
   .RST(RST)
);

// End of DCIRESET_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

DCM_ADV

Digital Clock Manager with Advanced Features



The Digital Clock Managers (DCM) provide a wide range of powerful clock management features. In the case of DCM_ADV, these include:

- *Phase Shifting*

  The three outputs driving the same frequency as CLK0 are delayed by a fourth, a half, and then three-fourths of a clock period. An additional control signal optionally shifts all of the nine clock outputs by a fixed fraction of the input clock period (defined during configuration and described in multiples of the clock period divided by 256).

  The user can also dynamically and repetitively move the phase forwards or backwards by one unit of the clock period divided by 256. Any phase shift is always invoked as a specific fraction of the clock period, and is always implemented by moving delay taps with a resolution of DCM_TAP.

- *Dynamic Reconfiguration*

  The DADDR[6:0], DI[15:0], DWE, DEN, CCLK inputs and DO[15:0] and DRDY outputs are available to dynamically reconfigure select DCM functions. With dynamic reconfiguration, DCM attributes are changeable to select a different phase shift, frequency, or frequency-mode setting from the currently configured settings.

Port Descriptions

There are four types of DCM ports available in the Virtex-4 architecture:

1. Clock Input Ports
2. Control and Data Input Ports
3. Clock Output Ports
4. Status and Data Output Ports

Available Ports

Available Ports	Port Names
Clock Input	CLKIN, CLKFB, PSCLK, DCLK
Control and Data Input	RST, PSINCDEC, PSEN, DADDR[6:0], DI[15:0], DWE, DEN
Clock Output	CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKDV, CLKFX, CLKFX180
Status and Data Output	LOCKED, PSDONE, DO[15:0], DRDY

Clock Input Ports

Source Clock Input - CLKIN

The source clock (CLKIN) input pin provides the source clock to the DCM. The CLKIN frequency must fall in the ranges specified in the Virtex-4 Data Sheet. The clock input signal comes from one of the following buffers:

1. IBUFG – Global Clock Input Buffer

   The DCM compensates for the clock input path with an IBUFG on the same edge (top or bottom) of the device as the DCM is used.

2. BUFGCTRL – Internal Global Clock Buffer

   Any BUFGCTRL can drive any DCM in the Virtex-4 device using the dedicated global routing. A BUFGCTRL can drive the DCM CLKIN pin when used to connect two DCM in series.

3. IBUF – Input Buffer

   When IBUF drives CLKIN input, the PAD to DCM input skew is not compensated.

Feedback Clock Input - CLKFB

The feedback clock (CLKFB) input pin provides a reference or feedback signal to the DCM to delay-compensate the clock outputs, and align it with the clock input. To provide the necessary feedback to the DCM, connect only the CLK0 DCM outputs to the CLKFB pin and set the CLK_FEEDBACK attribute to 1X. When the CLKFB pin is connected, CLK0, CLK2X, CLKDV, and CLKFX will be deskewed to CLKIN. When the CLKFB pin is not connected, DCM clock outputs are not deskewed to CLKIN. However, the phase relationship between all output clocks is preserved.

During internal feedback configuration, the CLK0 output of a DCM connects to a global buffer on the same top or bottom half of the device. The output of the global buffer connects to the CLKFB input of the same DCM.

During the external feedback configuration, the following rules apply:

1. To forward the clock, the CLK0 of the DCM must directly drive an OBUF or a BUFG-to-DDR configuration.
2. External to the FPGA, the forwarded clock signal must be connected to the IBUFG (GCLK pin) or the IBUF driving the CLKFB of the DCM.

The feedback clock input signal can be driven by one of the following buffers:

1. BUFG – Global Clock Input Buffer

   This is the preferred source for an external feedback configuration. When an IBUFG drives a CLKFB pin of a DCM in the same (top or bottom) half of the device, the pad to DCM skew is compensated for deskew.

2. BUFGCTRL – Internal Global Clock Buffer

   This is an internal feedback configuration.

3. IBUF – Input Buffer

This is an external feedback configuration. When IBUF is used, the PAD to DCM input skew is not compensated.

Phase-Shift Clock Input - PSCLK

The phase-shift clock (PSCLK) input pin provides the source clock for the DCM phase shift. The frequency of PSCLK is the same as, lower than, or higher than the frequency of CLKIN. The phase-shift clock signal can be driven by any clock source (external or internal), including:

1. IBUF - Input Buffer.
2. IBUFG - Global Clock Input Buffer.

   To access the dedicated routing, the IBUFGs is on the same edge of the device (top or bottom) as the DCM can be used to drive a PSCLK input of the DCM.
3. BUFGCTRL - An Internal Global Buffer.
4. Internal Clock - Any internal clock using general purpose routing.

The frequency range of PSCLK is defined by PSCLK_FREQ_LF/HF. This input must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Dynamic Reconfiguration Clock Input - DCLK

The DCLK input pin provides the source clock for the DCM's dynamic reconfiguration circuit. The frequency of DCLK can be asynchronous (in phase and frequency) to CLKIN. The dynamic reconfiguration clock signal is driven by any clock source (external or internal), including:

1. IBUF - Input Buffer.
2. IBUFG - Global Clock Input Buffer.

   Only the IBUFGs on the same edge of the device (top or bottom) as the DCM can be used to drive a CLKIN input of the DCM.
3. BUFGCTRL - An Internal Global Buffer.
4. Internal Clock - Any internal clock using general purpose routing.

The frequency range of DCLK is described in the Virtex-4 Data Sheet. When dynamic reconfiguration is not used, this input must be tied to ground. For more information on Dynamic Configuration, please see the *Configuration User Guide.*

Control and Data Input Ports

Reset Input - RST

The reset (RST) input pin resets the DCM circuitry. The RST signal is an active High asynchronous reset. Asserting the RST signal asynchronously forces all DCM outputs Low (the LOCKED signal, all status signals, and all output clocks within four source clock cycles). Because the reset is asynchronous, the last cycle of the clocks can exhibit an unintended short pulse, severely distorting duty-cycle, and no longer deskew with respect to one another while deasserting Low. Only use the RST pin when reconfiguring the device or changing the input frequency. Deasserting the RST signal synchronously starts the locking process at the next CLKIN cycle.

To ensure a proper DCM reset and locking process, the RST signal must be deasserted after the CLKIN signal has been present and stable for at least three clock cycles.

The time it takes for the DCM to lock after a reset is specified as LOCK_DLL (for a DLL output) and LOCK_FX (for a DFS output). See the LOCK_DLL timing parameter in the Virtex-4 Data Sheet. The DCM locks faster at higher frequencies.

In all designs, the DCM must be held in reset until the clock is stable. During configuration, the DCM will be held in reset until GWE is released. If the clock is stable when GWE is released, DCM reset after configuration is not necessary.

Phase-Shift Increment/Decrement Input - PSINCDEC

The PSINCDEC input signal is synchronous with PSCLK. The PSINCDEC input signal is used to increment or decrement the phase-shift factor. As a result, the output clock will be phase shifted. The PSINCDEC signal is asserted High for increment, or deasserted Low for decrement. This input must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Phase-Shift Enable Input - PSEN

The PSEN input signal is synchronous with PSCLK. A variable phase-shift operation is initiated by the PSEN input signal. It must be activated for one period of PSCLK. After PSEN is initiated, the phase change is effective for up to 100 CLKIN pulse cycles, plus three PSCLK cycles, and is indicated by a High pulse on PSDONE. There are no sporadic changes or glitches on any output during the phase transition. From the time PSEN is enabled until PSDONE is flagged, the DCM output clock moves bit-by-bit from its original phase shift to the target phase shift. The phase-shift is complete when PSDONE is flagged. PSEN must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Dynamic Reconfiguration Data Input - DI[15:0]

The DI input bus provides reconfiguration data for dynamic reconfiguration. When not used, all bits must be assigned zeros. Please see the dynamic reconfiguration section of the *Configuration User Guide* for more information.

Dynamic Reconfiguration Address Input - DADDR[6:0]

The DADDR input bus provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros. The DO output bus will reflect the DCM's status. Please see the dynamic reconfiguration section of the *Configuration User Guide* for more information.

Dynamic Reconfiguration Write Enable Input - DWE

The DWE input pin provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low. Please see the dynamic reconfiguration section of the *Configuration User Guide* for more information.

Dynamic Reconfiguration Enable Input - DEN

The DEN input pin provides the enable control signal to access the dynamic reconfiguration feature. To reflect the DCM status signals on the DO output bus, when not used, it should be tied to High because if DEN is tied Low, DO will always output a Low signal. Please see the dynamic reconfiguration section of the *Configuration User Guide* for more information.

Clock Output Ports

A DCM provides nine clock outputs with specific frequency and phase relationships. When CLKFB is connected, all DCM clock outputs are deskewed to CLKIN. When CLKFB is not connected, the DCM outputs are not deskewed. However, the phase relationship between all output clocks is preserved.

1x Output Clock - CLK0

The CLK0 output clock provides a clock with the same frequency as the DCM's effective CLKIN frequency. By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True. When CLKFB is connected, CLK0 is deskewed to CLKIN.

1x Output Clock, 90° Phase Shift - CLK90

The CLK90 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 90°.

1x Output Clock, 180° Phase Shift - CLK180

The CLK180 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 180°.

1x Output Clock, 270° Phase Shift - CLK270

The CLK270 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 270°.

2x Output Clock - CLK2X

The CLK2X output clock provides a clock that is phase aligned to CLK0, with twice the CLK0 frequency, and with an automatic 50/50 duty-cycle correction. Until the DCM is locked, the CLK2X output appears as a 1x version of the input clock with a 25/75 duty cycle. This behavior allows the DCM to lock on the correct edge with respect to the source clock.

2x Output Clock, 180° Phase Shift - CLK2X180

The CLK2X180 output clock provides a clock with the same frequency as the DCM's CLK2X only phase-shifted by 180°.

Frequency Divide Output Clock - CLKDV

The frequency divide (CLKDV) output clock provides a clock that is phase aligned to CLK0 with a frequency that is a fraction of the effective CLKIN frequency. The fraction is determined by the CLKDV_DIVIDE attribute.

By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True.

Frequency Output Clock - CLKFX

The frequency (CLKFX) output clock provides a clock with the following frequency definition:

CLKFX Frequency = (M/D)x (Effective CLKIN Frequency)

In this equation, M is the multiplier (numerator) with a value defined by the CLKFX_MULTIPLY attribute. D is the divisor (denominator) with a value defined by the CLKFX_DIVIDE attribute. Specifications for M and D, as well as input and output frequency ranges for the frequency synthesizer, are provided in the Virtex-4 Data Sheet.

The rising edge of CLKFX output is phase aligned to the rising edges of CLK0, CLK2X, and CLKDV. When M and D to have no common factor, the alignment occurs only once every D cycles of CLK0.

By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True.

Frequency Synthesis Output Clock, 180° - CLKFX180

The CLKFX180 output clock provides a clock with the same frequency as the DCM's CLKFX only phase-shifted by 180°.

Status and Data Output Ports

Locked Output - LOCKED

The LOCKED output signals the status of the DCM circuitry by locking it to the desired frequency or phase shift. To achieve lock, the DCM samples several thousand clock cycles. After the DCM achieves lock, the LOCKED signal is asserted High. The DCM timing parameters section of the Virtex-4 Data Sheet provides estimates for locking times.

To guarantee an established system clock at the end of the start-up cycle, the DCM can delay the completion of the device configuration process until after the DCM is locked. The STARTUP_WAIT attribute activates this feature.

Until the LOCKED signal is activated, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular, the CLK2X output can appear as a 1x clock with a 25/75 duty cycle.

Phase Shift Done Output - PSDONE

The PSDONE output signal is synchronous to PSCLK. It indicates, by pulsing High for one period of PSCLK, the completion of a requested phase shift. This signal also indicates that a change to the phase shift is available. The PSDONE output signal is not valid if the phase shift feature is not being used or is in fixed mode.

Status of Dynamic Reconfiguration Data Output - DO[15:0]

The DO output bus provides DCM status when not using the dynamic reconfiguration feature and a data output when using dynamic reconfiguration. Further information on using DO as the data output is available in the dynamic reconfiguration section of the *Configuration User Guide*.

If DEN, DWE, DADDR, DI, and DO are not used, using DCM_BASE or DCM_PS instead of DCM_ADV is strongly recommended. Otherwise, all unused inputs and output pins should be left unconnected or assigned to the previously recommended values.

DCM Status Mapping to DO Bus

DO Bit	Status	Description
DO[0]	Phase-shift overflow	Asserted when the DCM is phase shifted beyond the allowed phase shift value or when the absolute delay range of the phase-shift delay line is exceeded.
DO[1]	CLKIN stopped	Asserted when the input clock is stopped (CLKIN remains High or Low for one or more clock cycles). When CLKIN is stopped, the DO[1] CLKIN stopped status will assert within nine CLKIN cycles. When CLKIN is restarted, CLK0 will start toggling and DO[1] will deassert within nine clock cycles.
DO[2]	CLKFX stopped	Asserted when CLKFX stops. The DO[2] CLKFX stopped status will assert within 257 to 260 CLKIN cycles after CLKFX stopped. CLKFX will not resume, and DO[2] will not deassert until the DCM is reset.
DO[3]	CLKFB stopped	Asserted the feedback clock is stopped (CLKFB remains High or Low for one or more clock cycles). The DO[3] CLKFB stopped status will assert within six CLKIN cycles after CLKFB is stopped. CLKFB stopped will deassert within six CLKIN cycles when CLKFB resumes after being stopped momentarily. An occasionally skipped CLKFB will not affect the DCM operation. However, stopping CLKFB for a long time can result in the DCM losing LOCKED. When LOCKED is lost, the DCM needs to be reset to resume operation.
DO[15:4]	Not assigned	

Dynamic Reconfiguration Ready Output - DRDY

The DRDY output pin provides ready status for the DCM's dynamic reconfiguration feature. The dynamic reconfiguration section of the *Configuration User Guide* provides more details on using DO as a data output.

DCM Attributes

A handful of DCM attributes govern the DCM functionality. This section provides a detailed description of each attribute. For more information on applying these attributes in UCF, VHDL, or Verilog code, refer to the Xilinx *Constraints Guide*.

CLKDV_DIVIDE Attribute

The CLKDV_DIVIDE attribute controls the CLKDV frequency. Since the source clock frequency is divided by the value of this attribute, the possible values for CLKDV_DIVIDE are: 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16. The default value is 2. In the low frequency mode, any CLKDV_DIVIDE value produces a CLKDV output with a 50/50 duty-cycle. In the high frequency mode, the CLKDV_DIVIDE value must be set to an integer value to produce a CLKDV output with a 50/50 duty-cycle.

Non-Integer CLKDV_DIVIDE

CLKDV_DIVIDE Value	CLKDV Duty Cycle(High Frequency Mode)
1.5	1/3
2.5	2/5

CLKDV_DIVIDE Value	CLKDV Duty Cycle(High Frequency Mode)
3.5	3/7
4.5	4/9
5.5	5/11
6.5	6/13
7.5	7/15

CLKFX_MULTIPLY and CLKFX_DIVIDE Attribute

The CLKFX_MULTIPLY attribute sets the multiply value (M) of the CLKFX output. The CLKFX_DIVIDE attribute sets the divisor (D) value of the CLKFX output. Both control the CLKFX output making the CLKFX frequency equal the effective CLKIN (source clock) frequency multiplied by M/D. The possible values for M are any integer from two to 32. The possible values for D are any integer from one to 32. The default settings are M = 4 and D = 1.

CLKIN_PERIOD Attribute

The CLKIN_PERIOD attribute specifies the source clock period (in nanoseconds). The default value is 0.0 ns.

CLKIN_DIVIDE_BY_2 Attribute

The CLKIN_DIVIDE_BY_2 attribute determines the effective CLKIN frequency applied to the DCM circuitry. When set to False, the effective CLKIN frequency of the DCM equals the source clock frequency driving the CLKIN input. When set to True, the CLKIN frequency is divided by two before it reaches the rest of the DCM circuitry. Thus, the DCM circuitry sees half the frequency applied to the CLKIN input and operates based on this frequency. For example, if a 100 MHz clock drives CLKIN, and CLKIN_DIVIDE_BY_2 is set to True; then the effective CLKIN frequency is 50 MHz. Thus, CLK0 output is 50 MHz and CLK2X output is 100 MHz. The effective CLKIN frequency must be used to evaluate any operation or specification derived from CLKIN frequency. The possible values for CLKIN_DIVIDE_BY_2 are True and False. The default value is False.

CLKOUT_PHASE_SHIFT Attribute

The CLKOUT_PHASE_SHIFT attribute indicates the mode of the phase shift applied to the DCM outputs. The possible values are NONE, FIXED, VARIABLE_POSITIVE, VARIABLE_CENTER, or DIRECT. The default value is NONE.

When set to NONE, a phase shift cannot be performed and a phase-shift value has no effect on the DCM outputs. When set to FIXED, the DCM outputs are phase shifted by a fixed phase from the CLKIN. The phase-shift value is determined by PHASE_SHIFT attribute. If the CLKOUT_PHASE_SHIFT attribute is set to FIXED or NONE, then the PSEN, PSINCDEC, and the PSCLK inputs must be tied to ground.

When set to VARIABLE_POSITIVE, the DCM outputs can be phase shifted in variable mode in the positive range with respect to CLKIN. When set to VARIABLE_CENTER, the DCM outputs can be phase shifted in variable mode, in the positive and negative range with respect to CLKIN. If set to VARIABLE_POSITIVE or VARIABLE_CENTER, each phase shift increment (or decrement) will increase (or decrease) the phase shift by a period of 1/256 x CLKIN.

When set to DIRECT, the DCM output can be phase shifted in variable mode in the positive range with respect to CLKIN. Each phase shift increment/decrement will increase/decrease the phase shift by one DCM_TAP.

The starting phase in the VARIABLE_POSITIVE and VARIABLE_CENTER modes is determined by the phase-shift value. The starting phase in the DIRECT mode is always zero, regardless of the value specified by the PHASE_SHIFT attribute. Thus, the PHASE_SHIFT attribute should be set to zero when DIRECT mode is used. A non-zero phase-shift value for DIRECT mode can be loaded to the DCM using Dynamic Reconfiguration Ports.

CLK_FEEDBACK Attribute

The CLK_FEEDBACK attribute determines the type of feedback applied to the CLKFB. The possible values are 1X or NONE. The default value is 1X. When set to 1X, CLKFB pin must be driven by CLK0. When set to NONE leave the CLKFB pin unconnected.

DESKEW_ADJUST Attribute

The DESKEW_ADJUST attribute affects the amount of delay in the feedback path. The possible values are SYSTEM_SYNCHRONOUS, SOURCE_SYNCHRONOUS, 0, 1, 2, 3, ... or 31. The default value is SYSTEM_SYNCHRONOUS.

For most designs, the default value is appropriate. In a source-synchronous design, set this attribute to SOURCE_SYNCHRONOUS. The remaining values should only be used when consulting with Xilinx.

DFS_FREQUENCY_MODE Attribute

The DFS_FREQUENCY_MODE attribute specifies the frequency mode of the frequency synthesizer (DFS). The possible values are Low and High. The default value is Low. The frequency ranges for both frequency modes are specified in the Virtex-4 Data Sheet. DFS_FREQUENCY_MODE determines the frequency range of CLKIN, CLKFX, and CLKFX180.

DLL_FREQUENCY_MODE Attribute

The DLL_FREQUENCY_MODE attribute specifies either the High or Low frequency mode of the delay-locked loop (DLL). The default value is Low. The frequency ranges for both frequency modes are specified in the Virtex-4 Data Sheet.

DUTY_CYCLE_CORRECTION Attribute

The DUTY_CYCLE_CORRECTION attribute controls the duty cycle correction of the 1x clock outputs: CLK0, CLK90, CLK180, and CLK270. The possible values are True and False. The default value is True. When set to True, the 1x clock outputs are duty cycle corrected to a 50/50 duty cycle. It is strongly recommended to always set the DUTY_CYCLE_CORRECTION attribute to True. Setting this attribute to False does not necessarily produce output clocks with the same duty cycle as the source clock.

DCM_PERFORMANCE_MODE Attribute

The DCM_PERFORMANCE_MODE attribute allows the choice of optimizing the DCM either for high frequency and low jitter or for low frequency and a wide phase-shift range. The attribute values are MAX_SPEED and MAX_RANGE. The default

value is MAX_SPEED. When set to MAX_SPEED, the DCM is optimized to produce high frequency clocks with low jitter. However, the phase-shift range is smaller than when MAX_RANGE is selected. When set to MAX_RANGE, the DCM is optimized to produce low frequency clocks with a wider phase-shift range. The DCM_PERFORMANCE_MODE affects the following specifications: DCM input and output frequency range, phase-shift range, output jitter, DCM_TAP, CLKIN_CLKFB_PHASE, CLKOUT_PHASE, and duty-cycle precision. The Virtex-4 Data Sheet specifies these values.

For most cases, the DCM_PERFORMANCE_MODE attribute should be set to MAX_SPEED (default). Only consider changing to MAX_RANGE in the following situations:

- The frequency needs to be below the low frequency limit of the MAX_SPEED setting.
- A greater absolute phase-shift range is required.

FACTORY_JF Attribute

The FACTORY_JF attribute affects the DCM's jitter filter characteristic. This attribute is set the default value of F0F0 and should not be modified unless otherwise instructed by Xilinx.

PHASE_SHIFT Attribute

The PHASE_SHIFT attribute determines the amount of phase shift applied to the DCM outputs. This attribute can be used in both fixed or variable phase-shift mode. If used with variable mode, the attribute sets the starting phase shift. When CLKOUT_PHASE_SHIFT = VARIABLE_POSITIVE, the PHASE_SHIFT value range is 0 to 255. When CLKOUT_PHASE_SHIFT = VARIABLE_CENTER or FIXED, the PHASE_SHIFT value range is -255 to 255. When CLKOUT_PHASE_SHIFT = DIRECT, the PHASE_SHIFT value range is 0 to 1023. The default value is 0.

STARTUP_WAIT Attribute

The STARTUP_WAIT attribute determines whether the startup cycle waits for DCM to lock. The possible values for this attribute are True and False. The default value is False. When STARTUP_WAIT is set to True, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the startup cycle specified by LCK_cycle until the DCM is locked.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CLK_FEEDBACK	STRING	"1X" or "NONE	“1X”	Specifies the clock feedback of the allowed value
CLKDV_DIVIDE	FLOAT	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 CLKDLL, CLKDLLE, CLKDLLHF, or DCM 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_MULTIPLY	INTEGER	2 to 32	4	Specifies the frequency multiplier value for the CLKFX output.

Attribute	Type	Allowed Values	Default	Description
CLKIN_DIVIDE_BY_2	BOOLEAN	FALSE, TRUE	FALSE	Allows for the input clock frequency to be divided in half when such a reduction is necessary to meet the DCM input clock frequency requirements.
CLKIN_PERIOD	FLOAT	1.25 to 1000.00	0.0	Specifies period of input clock in ns from 1.25 to 1000.00.
CLKOUT_PHASE_SHIFT	STRING	"NONE" or "FIXED" or "VARIABLE_POSITIVE" or "VARIABLE_CENTER" or "DIRECT	"NONE"	Specifies the phase shift mode of allowed value.
DCM_AUTOCALIBRATION	BOOLEAN	TRUE, FALSE	TRUE	Specifies the additional circuitry necessary to ensure proper DCM operation. It is suggested that users consult with Xilinx before changing this attribute.
DCM_PERFORMANCE_MODE	STRING	"MAX_SPEED" or "MAX_RANGE	"MAX_SPEED"	Allows selection between maximum frequency and minimum jitter for low frequency and maximum phase shift range
DESKEW_ADJUST	STRING	"SOURCE_SYNCHRONOUS", "SYSTEM_SYNCHRONOUS" or "0" to "15	"SYSTEM_SYNCHRONOUS"	Affects the amount of delay in the feedback path, and should be used for source-synchronous interfaces.
DFS_FREQUENCY_MODE	STRING	"LOW" or "HIGH"	"LOW"	Specifies the frequency mode of the frequency synthesizer.
DLL_FREQUENCY_MODE	STRING	"LOW" or "HIGH"	"LOW"	Specifies the DLL's frequency mode.
DUTY_CYCLE_CORRECTION	BOOLEAN	TRUE, FALSE	TRUE	Corrects the duty cycle of the CLK0, CLK90, CLK180, and CLK270 outputs.
FACTORY_JF	16-Bit Hexadecimal	Any 16-Bit Hexadecimal value	F0F0	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 1023	0	Specifies the phase shift numerator. The range depends on CLKOUT_PHASE_SHIFT.
STARTUP_WAIT	BOOLEAN	FALSE, TRUE	FALSE	When TRUE, the configuration startup sequence waits in the specified cycle until the DCM locks.

Usage

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

VHDL Instantiation Template

```
--    DCM_ADV    : In order to incorporate this function into the design,
--     VHDL      : the following instance declaration needs to be placed
--   instance    : in the body of the design code.  The instance name
--  declaration  : (DCM_ADV_inst) and/or the port declarations after the
--     code      : "=>" declaration maybe changed to properly reference and
--               : connect this function to the design.  Unused inputs
--               : and outputs may 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 will be used
--               : for simulation.

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

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- DCM_ADV: Digital Clock Manager Circuit for Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   DCM_ADV_inst : DCM_ADV
   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 2 to 32
      CLKIN_DIVIDE_BY_2 => FALSE,    -- TRUE/FALSE to enable CLKIN divide by two feature
      CLKIN_PERIOD => 10.0, -- Specify period of input clock in ns from 1.25 to 1000.00
      CLKOUT_PHASE_SHIFT => "NONE",  -- Specify phase shift mode of NONE, FIXED,
                                     -- VARIABLE_POSITIVE, VARIABLE_CENTER or DIRECT
      CLK_FEEDBACK => "1X",  -- Specify clock feedback of NONE or 1X
      DCM_PERFORMANCE_MODE => "MAX_SPEED",   -- Can be MAX_SPEED or MAX_RANGE
      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",    -- LOW, HIGH, or HIGH_SER frequency mode for DLL
      DUTY_CYCLE_CORRECTION => TRUE,  -- Duty cycle correction, TRUE or FALSE
      FACTORY_JF => X"F0F0",          -- FACTORY JF Values Suggested to be set to X"F0F0"
      PHASE_SHIFT => 0,  -- Amount of fixed phase shift from -255 to 1023
      STARTUP_WAIT => FALSE)  -- Delay configuration DONE until DCM LOCK, TRUE/FALSE
   port map (
      CLK0 => CLK0,         -- 0 degree DCM CLK output
      CLK180 => CLK180,     -- 180 degree DCM CLK output
      CLK270 => CLK270,     -- 270 degree DCM CLK output
      CLK2X => CLK2X,       -- 2X DCM CLK output
      CLK2X180 => CLK2X180, -- 2X, 180 degree DCM CLK out
      CLK90 => CLK90,       -- 90 degree DCM CLK output
      CLKDV => CLKDV,       -- Divided DCM CLK out (CLKDV_DIVIDE)
      CLKFX => CLKFX,       -- DCM CLK synthesis out (M/D)
      CLKFX180 => CLKFX180, -- 180 degree CLK synthesis out
      DO => DO,             -- 16-bit data output for Dynamic Reconfiguration Port (DRP)
      DRDY => DRDY,         -- Ready output signal from the DRP
      LOCKED => LOCKED,     -- DCM LOCK status output
      PSDONE => PSDONE,     -- Dynamic phase adjust done output
      CLKFB => CLKFB,       -- DCM clock feedback
      CLKIN => CLKIN,       -- Clock input (from IBUFG, BUFG or DCM)
      DADDR => DADDR,       -- 7-bit address for the DRP
      DCLK => DCLK,         -- Clock for the DRP
      DEN => DEN,           -- Enable input for the DRP
      DI => DI,             -- 16-bit data input for the DRP
      DWE => DWE,           -- Active high allows for writing configuration memory
      PSCLK => PSCLK,       -- Dynamic phase adjust clock input
      PSEN => PSEN,         -- Dynamic phase adjust enable input
      PSINCDEC => PSINCDEC, -- Dynamic phase adjust increment/decrement
      RST => RST            -- DCM asynchronous reset input
   );

   -- End of DCM_ADV_inst instantiation
```

Verilog Instantiation Template

```
//   DCM_ADV   : 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_ADV_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 may be removed or commented out.

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

   // DCM_ADV: Digital Clock Manager Circuit for Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   DCM_ADV #(
      .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
```

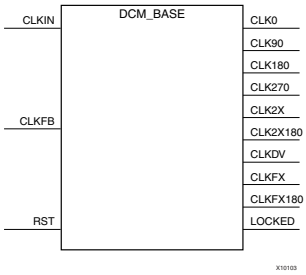
```
    .CLKFX_DIVIDE(1),   // Can be any integer from 1 to 32
    .CLKFX_MULTIPLY(4), // Can be any integer from 2 to 32
    .CLKIN_DIVIDE_BY_2("FALSE"), // TRUE/FALSE to enable CLKIN divide by two feature
    .CLKIN_PERIOD(10.0), // Specify period of input clock in ns from 1.25 to 1000.00
    .CLKOUT_PHASE_SHIFT("NONE"), // Specify phase shift mode of NONE, FIXED,
                                 // VARIABLE_POSITIVE, VARIABLE_CENTER or DIRECT
    .CLK_FEEDBACK("1X"),  // Specify clock feedback of NONE, 1X or 2X
    .DCM_PERFORMANCE_MODE("MAX_SPEED"), // Can be MAX_SPEED or MAX_RANGE
    .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"), // LOW, HIGH, or HIGH_SER frequency mode for DLL
    .DUTY_CYCLE_CORRECTION("TRUE"), // Duty cycle correction, TRUE or FALSE
    .FACTORY_JF(16'hf0f0), // FACTORY JF value suggested to be set to 16'hf0f0
    .PHASE_SHIFT(0), // Amount of fixed phase shift from -255 to 1023
    .STARTUP_WAIT("FALSE") // Delay configuration DONE until DCM LOCK, TRUE/FALSE
) DCM_ADV_inst(
    .CLK0(CLK0),          // 0 degree DCM CLK output
    .CLK180(CLK180),      // 180 degree DCM CLK output
    .CLK270(CLK270),      // 270 degree DCM CLK output
    .CLK2X(CLK2X),        // 2X DCM CLK output
    .CLK2X180(CLK2X180), // 2X, 180 degree DCM CLK out
    .CLK90(CLK90),        // 90 degree DCM CLK output
    .CLKDV(CLKDV),        // Divided DCM CLK out (CLKDV_DIVIDE)
    .CLKFX(CLKFX),        // DCM CLK synthesis out (M/D)
    .CLKFX180(CLKFX180), // 180 degree CLK synthesis out
    .DO(DO),              // 16-bit data output for Dynamic Reconfiguration Port (DRP)
    .DRDY(DRDY),          // Ready output signal from the DRP
    .LOCKED(LOCKED),      // DCM LOCK status output
    .PSDONE(PSDONE),      // Dynamic phase adjust done output
    .CLKFB(CLKFB),        // DCM clock feedback
    .CLKIN(CLKIN),        // Clock input (from IBUFG, BUFG or DCM)
    .DADDR(DADDR),        // 7-bit address for the DRP
    .DCLK(DCLK),          // Clock for the DRP
    .DEN(DEN),            // Enable input for the DRP
    .DI(DI),              // 16-bit data input for the DRP
    .DWE(DWE),            // Active high allows for writing configuration memory
    .PSCLK(PSCLK),        // Dynamic phase adjust clock input
    .PSEN(PSEN),          // Dynamic phase adjust enable input
    .PSINCDEC(PSINCDEC), // Dynamic phase adjust increment/decrement
    .RST(RST)             // DCM asynchronous reset input
);

// End of DCM_ADV_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*. For more information on applying these attributes in UCF, VHDL, or Verilog code, refer to the Xilinx *Constraints Guide*.

DCM_BASE

Primitive: Digital Clock Manager with Basic Features



The DCM_BASE primitive accesses the basic, frequently used, DCM features and simplifies the user-interface ports.

The clock deskew, frequency synthesis, and fixed-phase shifting features are available to use with DCM_BASE.

Clock Deskew

The DCM contains a delay-locked loop (DLL) to completely eliminate clock distribution delays, by deskewing the DCM's output clocks with respect to the input clock. The DLL contains delay elements (individual small buffers) and control logic. The incoming clock drives a chain of delay elements, thus the output of every delay element represents a version of the incoming clock delayed at a different point.

The control logic contains a phase detector and a delay-line selector. The phase detector compares the incoming clock signal (CLKIN) against a feedback input (CLKFB) and steers the delay line selector, essentially adding delay to the output of DCM until the CLKIN and CLKFB coincide.

Frequency Synthesis

Separate outputs provide a doubled frequency (CLK2X and CLK2X180). Another output, CLKDV, provides a frequency that is a specified fraction of the input frequency.

Two other outputs, CLKFX and CLKFX180, provide an output frequency derived from the input clock by simultaneous frequency division and multiplication. The user can specify any integer multiplier (M) and divisor (D) within the range specified in the DCM Timing Parameters section of the Virtex-4 Data Sheet. An internal calculator determines the appropriate tap selection, to make the output edge coincide with the input clock whenever mathematically possible. For example, M = 9 and D = 5, multiply the frequency by 1.8, and the output rising edge is coincident with the input rising edge every five input periods equaling every nine output periods.

The DCM_BASE model will allow only NONE and FIXED CLKOUT_PHASE_SHIFT MODES. If any other mode is used (e.g., VARIABLE), the model will give an error message. If users *must* use the VARIABLE mode, they should use the DCM_PS model.

Port Descriptions

There are four types of DCM ports available in the Virtex-4 architecture:

1. Clock Input Ports
2. Control and Data Input Ports
3. Clock Output Ports
4. Status and Data Output Ports

Following are the available ports and port names for this primitive

Available Ports	Port Names
Clock Input	CLKIN, CLKFB
Control and Data Input	RST
Clock Output	CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKDV, CLKFX, CLKFX180
Status and Data Output	LOCKED

Clock Input Ports

Source Clock Input - CLKIN

The source clock (CLKIN) input pin provides the source clock to the DCM. The CLKIN frequency must fall in the ranges specified in the Virtex-4 Data Sheet. The clock input signal comes from one of the following buffers:

1. IBUFG – Global Clock Input Buffer

   The DCM compensates for the clock input path when an IBUFG on the same edge (top or bottom) of the device as the DCM is used.

2. BUFGCTRL – Internal Global Clock Buffer

   Any BUFGCTRL can drive any DCM in the Virtex-4 device using the dedicated global routing. A BUFGCTRL can drive the DCM CLKIN pin when used to connect two DCM in series. This path can or cannot be compensated for deskew depending on the component driving the BUFGCTRL and CLKFB pin.

3. BUF – Input Buffer

When IBUF drives CLKIN input, the PAD to DCM input skew is not compensated.

Feedback Clock Input - CLKFB

The feedback clock (CLKFB) input pin provides a reference or feedback signal to the DCM to delay-compensate the clock outputs, and align it with the clock input. To provide the necessary feedback to the DCM, connect only the CLK0 or CLK2X DCM outputs to the CLKFB pin and set the CLK_FEEDBACK attribute to 1X. When the CLKFB pin is connected, CLK0, CLK2X, CLKDV, and CLKFX will be deskewed to CLKIN. When the CLKFB pin is not connected, DCM clock outputs are not deskewed to CLKIN. However, the phase relationship between all output clocks is preserved.

During internal feedback configuration, the CLK0/CLK2X output of a DCM connects to a global buffer on the same top or bottom half of the device. The output of the global buffer connects to the CLKFB input of the same DCM.

During the external feedback configuration, the following rules apply:

1. To forward the clock, the CLK0 or CLK2X of the DCM must directly drive an OBUF or a BUFG-to-DDR configuration.
2. External to the FPGA, the forwarded clock signal must be connected to the IBUFG (GCLK pin) or the IBUF driving the CLKFB of the DCM.

The feedback clock input signal can be driven by one of the following buffers:

1. IBUFG – Global Clock Input Buffer

This is the preferred source for an external feedback configuration. When an IBUFG drives a CLKFB pin of a DCM in the same (top or bottom) half of the device, the pad to DCM skew is compensated for deskew.

2. BUFGCTRL – Internal Global Clock Buffer

   This is an internal feedback configuration.

3. IBUF – Input Buffer

This is an external feedback configuration. When IBUF is used, the PAD to DCM input skew is not compensated.

Control and Data Input Ports

Reset Input - RST

The reset (RST) input pin resets the DCM circuitry. The RST signal is an active High asynchronous reset. Asserting the RST signal asynchronously forces all DCM outputs Low (the LOCKED signal, all status signals, and all output clocks within four source clock cycles). Because the reset is asynchronous, the last cycle of the clocks can exhibit an unintended short pulse, severely distorting duty-cycle, and no longer deskew with respect to one another while deasserting Low. Only use the RST pin when reconfiguring the device or changing the input frequency. Deasserting the RST signal synchronously starts the locking process at the next CLKIN cycle.

To ensure a proper DCM reset and locking process, the RST signal must be deasserted after the CLKIN signal has been present and stable for at least three clock cycles.

The time it takes for the DCM to lock after a reset is specified as LOCK_DLL (for a DLL output) and LOCK_FX (for a DFS output). See the LOCK_DLL timing parameter in the Virtex-4 Data Sheet. The DCM locks faster at higher frequencies.

In all designs, the DCM must be held in reset until the clock is stable. During configuration, the DCM will be held in reset until GWE is released. If the clock is stable when GWE is released, DCM reset after configuration is not necessary.

Clock Output Ports

A DCM provides nine clock outputs with specific frequency and phase relationships. When CLKFB is connected, all DCM clock outputs are deskewed to CLKIN. When CLKFB is not connected, the DCM outputs are not deskewed. However, the phase relationship between all output clocks is preserved.

1x Output Clock - CLK0

The CLK0 output clock provides a clock with the same frequency as the DCM's effective CLKIN frequency. By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True. When CLKFB is connected, CLK0 is deskewed to CLKIN.

1x Output Clock, 90° Phase Shift - CLK90

The CLK90 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 90°.

1x Output Clock, 180° Phase Shift - CLK180

The CLK180 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 180°.

1x Output Clock, 270° Phase Shift - CLK270

The CLK270 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 270°.

2x Output Clock - CLK2X

The CLK2X output clock provides a clock that is phase aligned to CLK0, with twice the DCM's effective CLKIN frequency, and with an automatic 50/50 duty-cycle correction. By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True. The CLKIN_DIVIDE_BY_2 attribute description provides further information. Until the DCM is locked, the CLK2X output appears as a 1x version of the input clock with a 25/75 duty cycle. This behavior allows the DCM to lock on the correct edge with respect to the source clock.

2x Output Clock, 180° Phase Shift - CLK2X180

The CLK2X180 output clock provides a clock with the same frequency as the DCM's CLK2X only phase-shifted by 180°.

Frequency Divide Output Clock - CLKDV

The frequency divide (CLKDV) output clock provides a clock that is phase aligned to CLK0 with a frequency that is a fraction of the effective CLKIN frequency. The fraction is determined by the CLKDV_DIVIDE attribute.

By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True.

Frequency Multiply Output Clock - CLKFX

The frequency multiply (CLKFX) output clock provides a clock with the following frequency definition:

CLKFX frequency = (M/D) x (Effective CLKIN frequency)

In this equation, M is the multiplier (numerator) with a value defined by the CLKFX_MULTIPLY attribute. D is the divisor (denominator) with a value defined by the CLKFX_DIVIDE attribute. Specifications for M and D, as well as input and output frequency ranges for the frequency synthesizer, are provided in the Virtex-4 Data Sheet.

The rising edge of CLKFX output is phase aligned to the rising edges of CLK0, CLK2X, and CLKDV. When M and D to have no common factor, the alignment occurs only once every D cycles of CLK0.

By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True.

Frequency Multiply Output Clock, 180° - CLKFX180

The CLKFX180 output clock provides a clock with the same frequency as the DCM's CLKFX only phase-shifted by 180°.

Status and Data Output Ports

Locked Output - LOCKED

The LOCKED output signals the status of the DCM circuitry by locking it to the desired frequency or phase shift. To achieve lock, the DCM samples several thousand clock cycles. After the DCM achieves lock, the LOCKED signal is asserted High. The DCM timing parameters section of the Virtex-4 Data Sheet provides estimates for locking times.

To guarantee an established system clock at the end of the start-up cycle, the DCM can delay the completion of the device configuration process until after the DCM is locked. The STARTUP_WAIT attribute activates this feature.

Until the LOCKED signal is activated, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular, the CLK2X output can appear as a 1x clock with a 25/75 duty cycle.

DCM Status Mapping to DO Bus

DO Bit	Status	Description
DO[0]	Phase-shift overflow	Asserted when the DCM is phase shifted beyond the allowed phase shift value or when the absolute delay range of the phase-shift delay line is exceeded.
DO[1]	CLKIN stopped	Asserted when the input clock is stopped (CLKIN remains High or Low for one or more clock cycles). When CLKIN is stopped, the DO[1] CLKIN stopped status will assert within nine CLKIN cycles. When CLKIN is restarted, CLK0 will start toggling and DO[1] will deassert within nine clock cycles.
DO[2]	CLKFX stopped	Asserted when CLKFX stops. The DO[2] CLKFX stopped status will assert within 257 to 260 CLKIN cycles after CLKFX stopped. CLKFX will not resume, and DO[2] will not deassert until the DCM is reset.
DO[3]	CLKFB stopped	Asserted the feedback clock is stopped (CLKFB remains High or Low for one or more clock cycles). The DO[3] CLKFB stopped status will assert within six CLKIN cycles after CLKFB is stopped. CLKFB stopped will deassert within six CLKIN cycles when CLKFB resumes after being stopped momentarily. An occasionally skipped CLKFB will not affect the DCM operation. However, stopping CLKFB for a long time can result in the DCM losing LOCKED. When LOCKED is lost, the DCM needs to be reset to resume operation.
DO[15:4]	Not assigned	

DCM Attributes

A handful of DCM attributes govern the DCM functionality. This section provides a detailed description of each attribute. For more information on applying these attributes in UCF, VHDL, or Verilog code, refer to the Xilinx Constraints Guide.

CLKDV_DIVIDE Attribute

The CLKDV_DIVIDE attribute controls the CLKDV frequency. The source clock frequency is divided by the value of this attribute. The possible values for

CLKDV_DIVIDE are: 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16. The default value is 2. In the low frequency mode, any CLKDV_DIVIDE value produces a CLKDV output with a 50/50 duty-cycle. In the high frequency mode, the CLKDV_DIVIDE value must be set to an integer value to produce a CLKDV output with a 50/50 duty-cycle.

Non-Integer CLKDV_DIVIDE

CLKDV_DIVIDE Value	CLKDV Duty Cycle(High Frequency Mode)
1.5	1/3
2.5	2/5
3.5	3/7
4.5	4/9
5.5	5/11
6.5	6/13
7.5	7/15

CLKFX_MULTIPLY and CLKFX_DIVIDE Attribute

The CLKFX_MULTIPLY attribute sets the multiply value (M) of the CLKFX output. The CLKFX_DIVIDE attribute sets the divisor (D) value of the CLKFX output. Both control the CLKFX output making the CLKFX frequency equal the effective CLKIN (source clock) frequency multiplied by M/D. The possible values for M are any integer from two to 32. The possible values for D are any integer from one to 32. The default settings are M = 4 and D = 1.

CLKIN_PERIOD Attribute

The CLKIN_PERIOD attribute specifies the source clock period (in nanoseconds). The default value is 0.0 ns.

CLKIN_DIVIDE_BY_2 Attribute

The CLKIN_DIVIDE_BY_2 attribute determines the effective CLKIN frequency applied to the DCM circuitry. When set to False, the effective CLKIN frequency of the DCM equals the source clock frequency driving the CLKIN input. When set to True, the CLKIN frequency is divided by two before it reaches the rest of the DCM circuitry. Thus, the DCM circuitry sees half the frequency applied to the CLKIN input and operates based on this frequency. For example, if a 100 MHz clock drives CLKIN, and CLKIN_DIVIDE_BY_2 is set to True; then the effective CLKIN frequency is 50 MHz. Thus, CLK0 output is 50 MHz and CLK2X output is 100 MHz. The effective CLKIN frequency must be used to evaluate any operation or specification derived from CLKIN frequency. The possible values for CLKIN_DIVIDE_BY_2 are True and False. The default value is False.

CLKOUT_PHASE_SHIFT Attribute

The CLKOUT_PHASE_SHIFT attribute indicates the mode of the phase shift applied to the DCM outputs. The possible values are NONE, FIXED, VARIABLE_POSITIVE, VARIABLE_CENTER, or DIRECT. The default value is NONE.

When set to NONE, a phase shift cannot be performed and a phase-shift value has no affect on the DCM outputs. When set to FIXED, the DCM outputs are phase shifted by

a fixed phase from the CLKIN. The phase-shift value is determined by PHASE_SHIFT attribute. If the CLKOUT_PHASE_SHIFT attribute is set to FIXED or NONE, then the PSEN, PSINCDEC, and the PSCLK inputs must be tied to ground.

When set to VARIABLE_POSITIVE, the DCM outputs can be phase shifted in variable mode in the positive range with respect to CLKIN. When set to VARIABLE_CENTER, the DCM outputs can be phase shifted in variable mode, in the positive and negative range with respect to CLKIN. If set to VARIABLE_POSITIVE or VARIABLE_CENTER, each phase shift increment (or decrement) will increase (or decrease) the phase shift by a period of 1/256 x CLKIN.

When set to DIRECT, the DCM output can be phase shifted in variable mode in the positive range with respect to CLKIN. Each phase shift increment/decrement will increase/decrease the phase shift by one DCM_TAP.

The starting phase in the VARIABLE_POSITIVE and VARIABLE_CENTER modes is determined by the phase-shift value. The starting phase in the DIRECT mode is always zero, regardless of the value specified by the PHASE_SHIFT attribute. Thus, the PHASE_SHIFT attribute should be set to zero when DIRECT mode is used. A non-zero phase-shift value for DIRECT mode can be loaded to the DCM using Dynamic Reconfiguration Ports.

CLK_FEEDBACK Attribute

The CLK_FEEDBACK attribute determines the type of feedback applied to the CLKFB. The possible values are 1X or NONE. The default value is 1X. When set to 1X, CLKFB pin must be driven by CLK0. When set to NONE leave the CLKFB pin unconnected.

DESKEW_ADJUST Attribute

The DESKEW_ADJUST attribute affects the amount of delay in the feedback path. The possible values are SYSTEM_SYNCHRONOUS, SOURCE_SYNCHRONOUS,

0, 1, 2, 3, ... or 31. The default value is SYSTEM_SYNCHRONOUS.

For most designs, the default value is appropriate. In a source-synchronous design, set this attribute to SOURCE_SYNCHRONOUS. The remaining values should only be used when consulting with Xilinx.

DFS_FREQUENCY_MODE Attribute

The DFS_FREQUENCY_MODE attribute specifies the frequency mode of the frequency synthesizer (DFS). The possible values are Low and High. The default value is Low. The frequency ranges for both frequency modes are specified in the Virtex-4 Data Sheet. DFS_FREQUENCY_MODE determines the frequency range of CLKIN, CLKFX, and CLKFX180.

DLL_FREQUENCY_MODE Attribute

The DLL_FREQUENCY_MODE attribute specifies either the High or Low frequency mode of the delay-locked loop (DLL). The default value is Low. The frequency ranges for both frequency modes are specified in the Virtex-4 Data Sheet.

DUTY_CYCLE_CORRECTION Attribute

The DUTY_CYCLE_CORRECTION attribute controls the duty cycle correction of the 1x clock outputs: CLK0, CLK90, CLK180, and CLK270. The possible values are True and False. The default value is True. When set to True, the 1x clock outputs are duty cycle corrected to a 50/50 duty cycle. It is strongly recommended to always set the DUTY_CYCLE_CORRECTION attribute to True. Setting this attribute to False does not necessarily produce output clocks with the same duty cycle as the source clock.

DCM_PERFORMANCE_MODE Attribute

The DCM_PERFORMANCE_MODE attribute allows the choice of optimizing the DCM either for high frequency and low jitter or for low frequency and a wide phase-shift range. The attribute values are MAX_SPEED and MAX_RANGE. The default value is MAX_SPEED. When set to MAX_SPEED, the DCM is optimized to produce high frequency clocks with low jitter. However, the phase-shift range is smaller than when MAX_RANGE is selected. When set to MAX_RANGE, the DCM is optimized to produce low frequency clocks with a wider phase-shift range. The DCM_PERFORMANCE_MODE affects the following specifications: DCM input and output frequency range, phase-shift range, output jitter, DCM_TAP, CLKIN_CLKFB_PHASE, CLKOUT_PHASE, and duty-cycle precision. The Virtex-4 Data Sheet specifies these values.

For most cases, the DCM_PERFORMANCE_MODE attribute should be set to MAX_SPEED (default). Consider changing to MAX_RANGE in the following situations:

- The frequency needs to be below the low frequency limit of the MAX_SPEED setting.
- A greater absolute phase-shift range is required.

FACTORY_JF Attribute

The FACTORY_JF attribute affects the DCM's jitter filter characteristic. This attribute is set the default value of F0F0 and should not be modified unless otherwise instructed by Xilinx.

PHASE_SHIFT Attribute

The PHASE_SHIFT attribute determines the amount of phase shift applied to the DCM outputs. This attribute can be used in either NONE or FIXED phase-shift mode. When the CLKOUT_PHASE_SHIFT = FIXED, the PHASE_SHIFT value range is 0 to 255. When CLKOUT_PHASE_SHIFT = FIXED_CENTER, the PHASE_SHIFT value range is -255 to 255. When CLKOUT_PHASE_SHIFT = DIRECT, the PHASE_SHIFT value range is 0 to 1023. The default value is 0.

If you need to use the VARIABLE PHASE_SHIFT mode, you must use DCM_PS.

STARTUP_WAIT Attribute

The STARTUP_WAIT attribute determines whether the startup cycle waits for DCM to lock. The possible values for this attribute are True and False. The default value is False. When STARTUP_WAIT is set to True, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the startup cycle specified by LCK_cycle until the DCM is locked.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CLK_FEEDBACK	STRING	"1X"or "NONE	“1X”	Specifies the feedback input to the DCM (CLK0, or CLK2X).
CLKDV_DIVIDE	FLOAT	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 CLKDLL, CLKDLLE, CLKDLLHF, or DCM 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_MULTIPLY	INTEGER	2 to 32	4	Specifies the frequency multiplier value for the CLKFX output.
CLKIN_DIVIDE_BY_2	BOOLEAN	FALSE, TRUE	FALSE	Allows for the input clock frequency to be divided in half when such a reduction is necessary to meet the DCM input clock frequency requirements.
CLKIN_PERIOD	FLOAT	1.25 to 1000.00	0.0	Specifies the period of input clock in ns from 1.25 to 1000.00.
CLKOUT_PHASE_ SHIFT	STRING	"NONE" or "FIXED	"NONE”	Specifies the phase shift mode of allowed value.
DCM_AUTOCALI-BRATION	BOOLEAN	TRUE, FALSE	TRUE	Specifies the additional circuitry necessary to ensure proper DCM operation. It is suggested that users consult with Xilinx before changing this attribute.
DCM_PERFORMANCE_ MODE	STRING	"MAX_SPEED" or "MAX_RANGE”	"MAX_SPEED”	Allows selection between maximum frequency and minimum jitter for low frequency and maximum phase shift range
DESKEW_ADJUST	STRING	"SOURCE_SYNCHR ONOUS", "SYSTEM_SYNCHR ONOUS" or "0" to "15	"SYSTEM_SYN CHRONOUS"	Affects the amount of delay in the feedback path, and should be used for source-synchronous interfaces.
DFS_FREQUENCY_ MODE	STRING	"LOW" or "HIGH	“LOW”	Specifies the frequency mode of the frequency synthesizer.
DLL_FREQUENCY_ MODE	STRING	"LOW" or "HIGH	"LOW”	This specifies the DLL's frequency mode.
DUTY_CYCLE_ CORRECTION	BOOLEAN	TRUE, FALSE	TRUE	Corrects the duty cycle of the CLK0, CLK90, CLK180, and CLK270 outputs.
FACTORY_JF	16-Bit Hexadecimal	Any 16-Bit Hexadecimal value	F0F0	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 1023	0	Specifies the phase shift numerator. The range depends on CLKOUT_PHASE_SHIFT.
STARTUP_WAIT	BOOLEAN	FALSE, TRUE	FALSE	When set to TRUE, the configuration startup sequence waits in the specified cycle until the DCM locks.

Usage

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

VHDL Instantiation Template

```
--   DCM_BASE    : In order to incorporate this function into the design,
--     VHDL      : the following instance declaration needs to be placed
--   instance    : in the body of the design code.  The instance name
--  declaration  : (DCM_BASE_inst) and/or the port declarations after the
--     code      : "=>" declaration maybe changed to properly reference and
--               : connect this function to the design.  Unused inputs
--               : and outputs may 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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- DCM_BASE: Digital Clock Manager Circuit for Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   DCM_BASE_inst : DCM_BASE
   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 2 to 32
      CLKIN_DIVIDE_BY_2 => FALSE, -- TRUE/FALSE to enable CLKIN divide by two feature
      CLKIN_PERIOD => 10.0, -- Specify period of input clock in ns from 1.25 to 1000.00
      CLKOUT_PHASE_SHIFT => "NONE", -- Specify phase shift mode of NONE or FIXED
      CLK_FEEDBACK => "1X",         -- Specify clock feedback of NONE or 1X
      DCM_PERFORMANCE_MODE => "MAX_SPEED",   -- Can be MAX_SPEED or MAX_RANGE
      DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS", -- SOURCE_SYNCHRONOUS, SYSTEM_SYNCHRONOUS or
                                             --   an integer from 0 to 15
      DFS_FREQUENCY_MODE => "LOW",   -- LOW or HIGH frequency mode for frequency synthesis
      DLL_FREQUENCY_MODE => "LOW",   -- LOW, HIGH, or HIGH_SER frequency mode for DLL
      DUTY_CYCLE_CORRECTION => TRUE, -- Duty cycle correction, TRUE or FALSE
      FACTORY_JF => X"F0F0",          -- FACTORY JF Values Suggested to be set to X"F0F0"
      PHASE_SHIFT => 0, -- Amount of fixed phase shift from -255 to 1023
      STARTUP_WAIT => FALSE) -- Delay configuration DONE until DCM LOCK, TRUE/FALSE
   port map (
      CLK0 => CLK0,         -- 0 degree DCM CLK ouptput
      CLK180 => CLK180,     -- 180 degree DCM CLK output
      CLK270 => CLK270,     -- 270 degree DCM CLK output
      CLK2X => CLK2X,       -- 2X DCM CLK output
      CLK2X180 => CLK2X180, -- 2X, 180 degree DCM CLK out
      CLK90 => CLK90,       -- 90 degree DCM CLK output
      CLKDV => CLKDV,       -- Divided DCM CLK out (CLKDV_DIVIDE)
      CLKFX => CLKFX,       -- DCM CLK synthesis out (M/D)
      CLKFX180 => CLKFX180, -- 180 degree CLK synthesis out
      LOCKED => LOCKED,     -- DCM LOCK status output
      CLKFB => CLKFB,       -- DCM clock feedback
      CLKIN => CLKIN,       -- Clock input (from IBUFG, BUFG or DCM)
      RST => RST            -- DCM asynchronous reset input
   );

   -- End of DCM_BASE_inst instantiation
```

Verilog Template

```
//  DCM_BASE   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
```

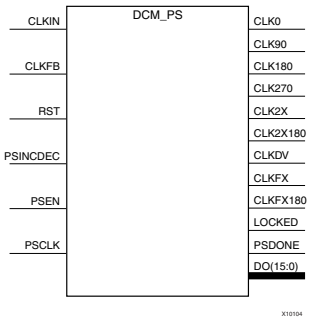
```
//  instance   : in the body of the design code.  The instance name
// declaration : (DCM_BASE_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  Unused inputs
//             : and outputs may be removed or commented out.

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

   // DCM_BASE: Digital Clock Manager Circuit for Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   DCM_BASE #(
      .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(10.0), // Specify period of input clock in ns from 1.25 to 1000.00
      .CLKOUT_PHASE_SHIFT("NONE"), // Specify phase shift mode of NONE or FIXED
      .CLK_FEEDBACK("1X"), // Specify clock feedback of NONE, 1X or 2X
      .DCM_PERFORMANCE_MODE("MAX_SPEED"), // Can be MAX_SPEED or MAX_RANGE
      .DESKEW_ADJUST("SYSTEM_SYNCHRONOUS"), // SOURCE_SYNCHRONOUS, SYSTEM_SYNCHRONOUS or
                                            //   an integer from 0 to 15
      .DFS_FREQUENCY_MODE("LOW"), // LOW or HIGH frequency mode for frequency synthesis
      .DLL_FREQUENCY_MODE("LOW"), // LOW, HIGH, or HIGH_SER frequency mode for DLL
      .DUTY_CYCLE_CORRECTION("TRUE"), // Duty cycle correction, TRUE or FALSE
      .FACTORY_JF(16'hf0f0), // FACTORY JF value suggested to be set to 16'hf0f0
      .PHASE_SHIFT(0), // Amount of fixed phase shift from -255 to 1023
      .STARTUP_WAIT("FALSE") // Delay configuration DONE until DCM LOCK, TRUE/FALSE
   ) DCM_BASE_inst(
      .CLK0(CLK0),         // 0 degree DCM CLK output
      .CLK180(CLK180),     // 180 degree DCM CLK output
      .CLK270(CLK270),     // 270 degree DCM CLK output
      .CLK2X(CLK2X),       // 2X DCM CLK output
      .CLK2X180(CLK2X180), // 2X, 180 degree DCM CLK out
      .CLK90(CLK90),       // 90 degree DCM CLK output
      .CLKDV(CLKDV),       // Divided DCM CLK out (CLKDV_DIVIDE)
      .CLKFX(CLKFX),       // DCM CLK synthesis out (M/D)
      .CLKFX180(CLKFX180), // 180 degree CLK synthesis out
      .LOCKED(LOCKED),     // DCM LOCK status output
      .CLKFB(CLKFB),       // DCM clock feedback
      .CLKIN(CLKIN),       // Clock input (from IBUFG, BUFG or DCM)
      .RST(RST)            // DCM asynchronous reset input
   );

   // End of DCM_BASE_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

DCM_PS

Primitive: Digital Clock Manager with Basic and Phase-Shift Features



The DCM_PS primitive access all DCM features and ports available in DCM_BASE with additional ports used by the variable phase shifting feature. DCM_PS also has these available DCM features: clock deskew, frequency synthesis, and fixed or variable phase shifting.

DCM_PS also supports the following, powerful, clock management features:

Clock Deskew

The DCM contains a delay-locked loop (DLL) to completely eliminate clock distribution delays, by deskewing the DCM's output clocks with respect to the input clock. The DLL contains delay elements (individual small buffers) and control logic. The incoming clock drives a chain of delay elements, thus the output of every delay element represents a version of the incoming clock delayed at a different point.

The control logic contains a phase detector and a delay-line selector. The phase detector compares the incoming clock signal (CLKIN) against a feedback input (CLKFB) and steers the delay line selector, essentially adding delay to the output of DCM until the CLKIN and CLKFB coincide.

Frequency Synthesis

Separate outputs provide a doubled frequency (CLK2X and CLK2X180). Another output, CLKDV, provides a frequency that is a specified fraction of the input frequency.

Two other outputs, CLKFX and CLKFX180, provide an output frequency derived from the input clock by simultaneous frequency division and multiplication. The user can specify any integer multiplier (M) and divisor (D) within the range specified in the DCM Timing Parameters section of the Virtex-4 Data Sheet. An internal calculator determines the appropriate tap selection, to make the output edge coincide with the input clock whenever mathematically possible. For example, M = 9 and D = 5, multiply the frequency by 1.8, and the output rising edge is coincident with the input rising edge every five input periods equaling every nine output periods.

Phase Shifting

The three outputs driving the same frequency as CLK0 are delayed by a fourth, a half, and then three-fourths of a clock period. An additional control signal optionally shifts all of the nine clock outputs by a fixed fraction of the input clock period (defined during configuration and described in multiples of the clock period divided by 256).

The user can also dynamically and repetitively move the phase forwards or backwards by one unit of the clock period divided by 256. Any phase shift is always invoked as a specific fraction of the clock period, and is always implemented by moving delay taps with a resolution of DCM_TAP.

Port Descriptions

There are four types of DCM ports available in the Virtex-4 architecture:

1. Clock Input Ports

2. Control and Data Input Ports
3. Clock Output Ports
4. Status and Data Output Ports

The following ports are available in the DCM_PS primitive:

Available Ports	Port Names
Clock Input	CLKIN, CLKFB, PSCLK
Control and Data Input	RST, PSINCDEC, PSEN
Clock Output	CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKDV, CLKFX, CLKFX180
Status and Data Output	LOCKED, PSDONE, DO[15:0]

Clock Input Ports

Source Clock Input - CLKIN

The source clock (CLKIN) input pin provides the source clock to the DCM. The CLKIN frequency must fall in the ranges specified in the Virtex-4 Data Sheet. The clock input signal comes from one of the following buffers:

1. IBUFG – Global Clock Input Buffer

   When an IBUFG drives a CLKFB pin of a DCM in the same (top or bottom) half of the device, the pad to DCM skew is compensated for deskew.

2. BUFGCTRL – Internal Global Clock Buffer

   Any BUFGCTRL can drive any DCM in the Virtex-4 device using the dedicated global routing. A BUFGCTRL can drive the DCM CLKIN pin when used to connect two DCM in series. This path can or cannot be compensated for deskew depending on the component driving the BUFGCTRL and CLKFB pin.

3. BUF – Input Buffer

When IBUF drives CLKIN input, the PAD to DCM input skew is not compensated.

Feedback Clock Input - CLKFB

The feedback clock (CLKFB) input pin provides a reference or feedback signal to the DCM to delay-compensate the clock outputs, and align it with the clock input. To provide the necessary feedback to the DCM, connect only the CLK0 or CLK2X DCM outputs to the CLKFB pin and set the CLK_FEEDBACK attribute to 1X. When the CLKFB pin is connected, CLK0, CLK2X, CLKDV, and CLKFX will be deskewed to CLKIN. When the CLKFB pin is not connected, DCM clock outputs are not deskewed to CLKIN. However, the phase relationship between all output clocks is preserved.

During internal feedback configuration, the CLK0/CLK2X output of a DCM connects to a global buffer on the same top or bottom half of the device. The output of the global buffer connects to the CLKFB input of the same DCM.

During the external feedback configuration, the following rules apply:

1. To forward the clock, the CLK0 or CLK2X of the DCM must directly drive an OBUF or a BUFG-to-DDR configuration.

2. External to the FPGA, the forwarded clock signal must be connected to the IBUFG (GCLK pin) or the IBUF driving the CLKFB of the DCM.

The feedback clock input signal can be driven by one of the following buffers:

1. IBUFG – Global Clock Input Buffer

   This is the preferred source for an external feedback configuration. When an IBUFG drives a CLKFB pin of a DCM in the same (top or bottom) half of the device, the pad to DCM skew is compensated for deskew.

2. BUFGCTRL – Internal Global Clock Buffer

   This is an internal feedback configuration.

3. IBUF – Input Buffer

This is an external feedback configuration. When IBUF is used, the PAD to DCM input skew is not compensated.

Phase-Shift Clock Input - PSCLK

The phase-shift clock (PSCLK) input pin provides the source clock for the DCM phase shift. The frequency of PSCLK is the same as, lower than, or higher than the frequency of CLKIN. The phase-shift clock signal can be driven by any clock source (external or internal), including:

1. IBUF - Input Buffer.
2. IBUFG - Global Clock Input Buffer

   To access the dedicated routing, only the IBUFGs on the same edge (top or bottom) as the DCM can be used to drive a PSCLK input of the DCM.

3. BUFGCTRL - An Internal Global Buffer.
4. Internal Clock - Any internal clock using internal routing.

The frequency range of PSCLK is defined by PSCLK_FREQ_LF/HF. This input must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Control and Data Input Ports

Reset Input - RST

The reset (RST) input pin resets the DCM circuitry. The RST signal is an active High asynchronous reset. Asserting the RST signal asynchronously forces all DCM outputs Low (the LOCKED signal, all status signals, and all output clocks within four source clock cycles). Because the reset is asynchronous, the last cycle of the clocks can exhibit an unintended short pulse, severely distorted duty-cycle, and no longer deskew with respect to one another while deasserting Low. Only use the RST pin when reconfiguring the device or changing the input frequency. Deasserting the RST signal synchronously starts the locking process at the next CLKIN cycle.

To ensure a proper DCM reset and locking process, the RST signal must be deasserted after the CLKIN signal has been present and stable for at least three clock cycles.

The time it takes for the DCM to lock after a reset is specified as LOCK_DLL (for a DLL output) and LOCK_FX (for a DFS output). See the LOCK_DLL timing parameter in the Virtex-4 Data Sheet. The DCM locks faster at higher frequencies.

In all designs, the DCM must be held in reset until the clock is stable. During configuration, the DCM will be held in reset until GWE is released. If the clock is stable when GWE is released, DCM reset after configuration is not necessary.

Phase-Shift Increment/Decrement Input - PSINCDEC

The PSINCDEC input signal is synchronous with PSCLK. The PSINCDEC input signal is used to increment or decrement the phase-shift factor. As a result, the output clock will be phase shifted. The PSINCDEC signal is asserted High for increment, or deasserted Low for decrement. This input must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Phase-Shift Enable Input - PSEN

The PSEN input signal is synchronous with PSCLK. A variable phase-shift operation is initiated by the PSEN input signal. It must be activated for one period of PSCLK. After PSEN is initiated, the phase change is effective for up to 100 CLKIN pulse cycles, plus three PSCLK cycles, and is indicated by a High pulse on PSDONE. There are no sporadic changes or glitches on any output during the phase transition. From the time PSEN is enabled until PSDONE is flagged, the DCM output clock moves bit-by-bit from its original phase shift to the target phase shift. The phase-shift is complete when PSDONE is flagged. PSEN must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Clock Output Ports

A DCM provides nine clock outputs with specific frequency and phase relationships. When CLKFB is connected, all DCM clock outputs are deskewed to CLKIN. When CLKFB is not connected, the DCM outputs are not deskewed. However, the phase relationship between all output clocks is preserved.

1x Output Clock - CLK0

The CLK0 output clock provides a clock with the same frequency as the DCM's effective CLKIN frequency. By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True. When CLKFB is connected, CLK0 is deskewed to CLKIN.

1x Output Clock, 90° Phase Shift - CLK90

The CLK90 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 90°.

1x Output Clock, 180° Phase Shift - CLK180

The CLK180 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 180°.

1x Output Clock, 270° Phase Shift - CLK270

The CLK270 output clock provides a clock with the same frequency as the DCM's CLK0 only phase-shifted by 270°.

2x Output Clock - CLK2X

The CLK2X output clock provides a clock that is phase aligned to CLK0, with twice the DCM's effective CLKIN frequency, and with an automatic 50/50 duty-cycle correction. By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True. The CLKIN_DIVIDE_BY_2 attribute description provides further information. Until the DCM is locked, the CLK2X output appears as a 1x version of the input clock with a 25/75 duty cycle. This behavior allows the DCM to lock on the correct edge with respect to the source clock.

2x Output Clock, 180° Phase Shift - CLK2X180

The CLK2X180 output clock provides a clock with the same frequency as the DCM's CLK2X only phase-shifted by 180°.

Frequency Divide Output Clock - CLKDV

The frequency divide (CLKDV) output clock provides a clock that is phase aligned to CLK0 with a frequency that is a fraction of the effective CLKIN frequency. The fraction is determined by the CLKDV_DIVIDE attribute.

By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True.

Frequency Multiply Output Clock - CLKFX

The frequency multiply (CLKFX) output clock provides a clock with the following frequency definition:

CLKFX Frequency = (M/D) x (Effective CLKIN frequency)

In this equation, M is the multiplier (numerator) with a value defined by the CLKFX_MULTIPLY attribute. D is the divisor (denominator) with a value defined by the CLKFX_DIVIDE attribute. Specifications for M and D, as well as input and output frequency ranges for the frequency synthesizer, are provided in the Virtex-4 Data Sheet.

The rising edge of CLKFX output is phase aligned to the rising edges of CLK0, CLK2X, and CLKDV. When M and D to have no common factor, the alignment occurs only once every D cycles of CLK0.

By default, the effective CLKIN frequency is equal to the CLKIN frequency, except when the CLKIN_DIVIDE_BY_2 attribute is set to True.

Frequency Multiply Output Clock, 180° - CLKFX180

The CLKFX180 output clock provides a clock with the same frequency as the DCM's CLKFX only phase-shifted by 180°.

Status and Data Output Ports

Locked Output - LOCKED

The LOCKED output signals the status of the DCM circuitry by locking it to the desired frequency or phase shift. To achieve lock, the DCM samples several thousand clock cycles. After the DCM achieves lock, the LOCKED signal is asserted High. The DCM timing parameters section of the Virtex-4 Data Sheet provides estimates for locking times.

To guarantee an established system clock at the end of the start-up cycle, the DCM can delay the completion of the device configuration process until after the DCM is locked. The STARTUP_WAIT attribute activates this feature.

Until the LOCKED signal is activated, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular, the CLK2X output can appear as a 1x clock with a 25/75 duty cycle.

DCM Status Mapping to DO Bus

DO Bit	Status	Description
DO[0]	Phase-shift overflow	Asserted when the DCM is phase shifted beyond the allowed phase shift value or when the absolute delay range of the phase-shift delay line is exceeded.
DO[1]	CLKIN stopped	Asserted when the input clock is stopped (CLKIN remains High or Low for one or more clock cycles). When CLKIN is stopped, the DO[1] CLKIN stopped status will assert within nine CLKIN cycles. When CLKIN is restarted, CLK0 will start toggling and DO[1] will deassert within nine clock cycles.
DO[2]	CLKFX stopped	Asserted when CLKFX stops. The DO[2] CLKFX stopped status will assert within 257 to 260 CLKIN cycles after CLKFX stopped. CLKFX will not resume, and DO[2] will not deassert until the DCM is reset.
DO[3]	CLKFB stopped	Asserted the feedback clock is stopped (CLKFB remains High or Low for one or more clock cycles). The DO[3] CLKFB stopped status will assert within six CLKIN cycles after CLKFB is stopped. CLKFB stopped will deassert within six CLKIN cycles when CLKFB resumes after being stopped momentarily. An occasionally skipped CLKFB will not affect the DCM operation. However, stopping CLKFB for a long time can result in the DCM losing LOCKED. When LOCKED is lost, the DCM needs to be reset to resume operation.
DO[15:4]	Not assigned	

Phase Shift Done Output - PSDONE

The PSDONE output signal is synchronous to PSCLK. It indicates, by pulsing High for one period of PSCLK, the completion of a requested phase shift. This signal also indicates that a change to the phase shift is available. The PSDONE output signal is not valid if the phase shift feature is not being used or is in fixed mode.

Status of Dynamic Reconfiguration Data Output - DO[15:0]

The DO output bus provides DCM status when not using the dynamic reconfiguration feature and a data output when using dynamic reconfiguration. Further information on using DO as the data output is available in the dynamic reconfiguration section of the *Configuration User Guide*.

If DEN, DWE, DADDR, DI, and DO are not used, using DCM_BASE or DCM_PS instead of DCM_ADV is strongly recommended. Otherwise, all unused inputs and output pins should be left unconnected or assigned to the previously recommended values.

DCM Attributes

A handful of DCM attributes govern the DCM functionality. This section provides a detailed description of each attribute. For more information on applying these attributes in UCF, VHDL, or Verilog code, refer to the Xilinx *Constraints Guide*.

CLKDV_DIVIDE Attribute

The CLKDV_DIVIDE attribute controls the CLKDV frequency. Since the source clock frequency is divided by the value of this attribute, the possible values for CLKDV_DIVIDE are: 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16. The default value is 2. In the low frequency mode, any CLKDV_DIVIDE value produces a CLKDV output with a 50/50 duty-cycle. In the high frequency mode, the CLKDV_DIVIDE value must be set to an integer value to produce a CLKDV output with a 50/50 duty-cycle.

Non-Integer CLKDV_DIVIDE

CLKDV_DIVIDE Value	CLKDV Duty Cycle(High Frequency Mode)
1.5	1/3
2.5	2/5
3.5	3/7
4.5	4/9
5.5	5/11
6.5	6/13
7.5	7/15

CLKFX_MULTIPLY and CLKFX_DIVIDE Attribute

The CLKFX_MULTIPLY attribute sets the multiply value (M) of the CLKFX output. The CLKFX_DIVIDE attribute sets the divisor (D) value of the CLKFX output. Both control the CLKFX output making the CLKFX frequency equal the effective CLKIN (source clock) frequency multiplied by M/D. The possible values for M are any integer from two to 32. The possible values for D are any integer from one to 32. The default settings are M = 4 and D = 1.

CLKIN_PERIOD Attribute

The CLKIN_PERIOD attribute specifies the source clock period (in nanoseconds). The default value is 0.0 ns.

CLKIN_DIVIDE_BY_2 Attribute

The CLKIN_DIVIDE_BY_2 attribute determines the effective CLKIN frequency applied to the DCM circuitry. When set to False, the effective CLKIN frequency of the DCM equals the source clock frequency driving the CLKIN input. When set to True, the CLKIN frequency is divided by two before it reaches the rest of the DCM circuitry. Thus, the DCM circuitry sees half the frequency applied to the CLKIN input and operates based on this frequency. For example, if a 100 MHz clock drives CLKIN, and CLKIN_DIVIDE_BY_2 is set to True; then the effective CLKIN frequency is 50 MHz. Thus, CLK0 output is 50 MHz and CLK2X output is 100 MHz. The effective CLKIN frequency must be used to evaluate any operation or specification derived from

CLKIN frequency. The possible values for CLKIN_DIVIDE_BY_2 are True and False. The default value is False.

CLKOUT_PHASE_SHIFT Attribute

The CLKOUT_PHASE_SHIFT attribute indicates the mode of the phase shift applied to the DCM outputs. The possible values are NONE, FIXED, VARIABLE_POSITIVE, VARIABLE_CENTER, or DIRECT. The default value is NONE.

When set to NONE, a phase shift cannot be performed and a phase-shift value has no affect on the DCM outputs. When set to FIXED, the DCM outputs are phase shifted by a fixed phase from the CLKIN. The phase-shift value is determined by PHASE_SHIFT attribute. If the CLKOUT_PHASE_SHIFT attribute is set to FIXED or NONE, then the PSEN, PSINCDEC, and the PSCLK inputs must be tied to ground.

When set to VARIABLE_POSITIVE, the DCM outputs can be phase shifted in variable mode in the positive range with respect to CLKIN. When set to VARIABLE_CENTER, the DCM outputs can be phase shifted in variable mode, in the positive and negative range with respect to CLKIN. If set to VARIABLE_POSITIVE or VARIABLE_CENTER, each phase shift increment (or decrement) will increase (or decrease) the phase shift by a period of 1/256 x CLKIN.

When set to DIRECT, the DCM output can be phase shifted in variable mode in the positive range with respect to CLKIN. Each phase shift increment/decrement will increase/decrease the phase shift by one DCM_TAP.

The starting phase in the VARIABLE_POSITIVE and VARIABLE_CENTER modes is determined by the phase-shift value. The starting phase in the DIRECT mode is always zero, regardless of the value specified by the PHASE_SHIFT attribute. Thus, the PHASE_SHIFT attribute should be set to zero when DIRECT mode is used. A non-zero phase-shift value for DIRECT mode can be loaded to the DCM using Dynamic Reconfiguration Ports.

CLK_FEEDBACK Attribute

The CLK_FEEDBACK attribute determines the type of feedback applied to the CLKFB. The possible values are 1X or NONE. The default value is 1X. When set to 1X, CLKFB pin must be driven by CLK0. When set to NONE leave the CLKFB pin unconnected.

DESKEW_ADJUST Attribute

The DESKEW_ADJUST attribute affects the amount of delay in the feedback path. The possible values are SYSTEM_SYNCHRONOUS, SOURCE_SYNCHRONOUS,

0, 1, 2, 3, ... or 31. The default value is SYSTEM_SYNCHRONOUS.

For most designs, the default value is appropriate. In a source-synchronous design, set this attribute to SOURCE_SYNCHRONOUS. The remaining values should only be used when consulting with Xilinx. For more information on the source synchronous interface, reference XAPP259.

DFS_FREQUENCY_MODE Attribute

The DFS_FREQUENCY_MODE attribute specifies the frequency mode of the frequency synthesizer (DFS). The possible values are Low and High. The default value is Low. The frequency ranges for both frequency modes are specified in the

Virtex-4 Data Sheet. DFS_FREQUENCY_MODE determines the frequency range of CLKIN, CLKFX, and CLKFX180.

DLL_FREQUENCY_MODE Attribute

The DLL_FREQUENCY_MODE attribute specifies either the High or Low frequency mode of the delay-locked loop (DLL). The default value is Low. The frequency ranges for both frequency modes are specified in the Virtex-4 Data Sheet.

DUTY_CYCLE_CORRECTION Attribute

The DUTY_CYCLE_CORRECTION attribute controls the duty cycle correction of the 1x clock outputs: CLK0, CLK90, CLK180, and CLK270. The possible values are True and False. The default value is True. When set to True, the 1x clock outputs are duty cycle corrected to a 50/50 duty cycle. It is strongly recommended to always set the DUTY_CYCLE_CORRECTION attribute to True. Setting this attribute to False does not necessarily produce output clocks with the same duty cycle as the source clock.

DCM_PERFORMANCE_MODE Attribute

The DCM_PERFORMANCE_MODE attribute allows the choice of optimizing the DCM either for high frequency and low jitter or for low frequency and a wide phase-shift range. The attribute values are MAX_SPEED and MAX_RANGE. The default value is MAX_SPEED. When set to MAX_SPEED, the DCM is optimized to produce high frequency clocks with low jitter. However, the phase-shift range is smaller than when MAX_RANGE is selected. When set to MAX_RANGE, the DCM is optimized to produce low frequency clocks with a wider phase-shift range. The DCM_PERFORMANCE_MODE affects the following specifications: DCM input and output frequency range, phase-shift range, output jitter, DCM_TAP, CLKIN_CLKFB_PHASE, CLKOUT_PHASE, and duty-cycle precision. The Virtex-4 Data Sheet specifies these values.

For most cases, the DCM_PERFORMANCE_MODE attribute should be set to MAX_SPEED (default). Only consider changing to MAX_RANGE in the following situations:

- The frequency needs to be below the low frequency limit of the MAX_SPEED setting.
- A greater absolute phase-shift range is required.

FACTORY_JF Attribute

The FACTORY_JF attribute affects the DCM's jitter filter characteristic. This attribute is set the default value of F0F0 and should not be modified unless otherwise instructed by Xilinx.

PHASE_SHIFT Attribute

The PHASE_SHIFT attribute determines the amount of phase shift applied to the DCM outputs. This attribute can be used in both fixed or variable phase-shift mode. If used with variable mode, the attribute sets the starting phase shift. When CLKOUT_PHASE_SHIFT = VARIABLE_POSITIVE, the PHASE_SHIFT value range is 0 to 255. When CLKOUT_PHASE_SHIFT = VARIABLE_CENTER or FIXED, the PHASE_SHIFT value range is -255 to 255. When CLKOUT_PHASE_SHIFT = DIRECT, the PHASE_SHIFT value range is 0 to 1023. The default value is 0.

STARTUP_WAIT Attribute

The STARTUP_WAIT attribute determines whether the startup cycle waits for DCM to lock. The possible values for this attribute are True and False. The default value is False. When STARTUP_WAIT is set to True, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the startup cycle specified by LCK_cycle until the DCM is locked.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CLK_FEEDBACK	STRING	"1X" or "NONE	"1X"	Specifies the clock feedback of allowed value.
CLKDV_DIVIDE	FLOAT	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 CLKDLL, CLKDLLE, CLKDLLHF, or DCM 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_MULTIPLY	INTEGER	2 to 32	4	Specifies the frequency multiplier value for the CLKFX output.
CLKIN_DIVIDE_BY_2	BOOLEAN	FALSE, TRUE	FALSE	Allows for the input clock frequency to be divided in half when such a reduction is necessary to meet the DCM input clock frequency requirements.
CLKIN_PERIOD	FLOAT	1.25 to 1000.00	0.0	Specifies the period of input clock in ns from 1.25 to 1000.00.
CLKOUT_PHASE_SHIFT	STRING	"NONE" or "FIXED" or "VARIABLE_POSITIVE" or "VARIABLE_CENTER" or "DIRECT	"NONE"	Specifies the phase shift mode of allowed value.
DCM_AUTOCALI-BRATION	BOOLEAN	TRUE, FALSE	TRUE	Specifies the additional circuitry necessary to ensure proper DCM operation. It is suggested that users consult with Xilinx before changing this attribute.
DCM_PERFORMANCE_MODE	STRING	"MAX_SPEED" or "MAX_RANGE	"MAX_SPEED"	Allows selection between maximum frequency and minimum jitter for low frequency and maximum phase shift range
DESKEW_ADJUST	STRING	"SOURCE_SYNCHRONOUS", "SYSTEM_SYNCHRONOUS" or "0" to "15	"SYSTEM_SYNCHRONOUS"	Affects the amount of delay in the feedback path, and should be used for source-synchronous interfaces.
DFS_FREQUENCY_MODE	STRING	"LOW" or "HIGH	"LOW"	Specifies the frequency mode of the frequency synthesizer.
DLL_FREQUENCY_MODE	STRING	"LOW" or "HIGH	"LOW"	This specifies the DLL's frequency mode.
DUTY_CYCLE_CORRECTION	BOOLEAN	TRUE, FALSE	TRUE	Corrects the duty cycle of the CLK0, CLK90, CLK180, and CLK270 outputs.

Attribute	Type	Allowed Values	Default	Description
FACTORY_JF	16-Bit Hexadecimal	Any 16-Bit Hexadecimal value	F0F0	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 1023	0	Specifies the phase shift numerator. The range depends on CLKOUT_PHASE_ SHIFT.
STARTUP_WAIT	BOOLEAN	FALSE, TRUE	FALSE	When set to TRUE, the configuration startup sequence waits in the specified cycle until the DCM locks.

Usage

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

VHDL Instantiation Template

```
--    DCM_PS     : In order to incorporate this function into the design,
--     VHDL      : the following instance declaration needs to be placed
--   instance    : in the body of the design code.  The instance name
--  declaration  : (DCM_PS_inst) and/or the port declarations after the
--     code      : "=>" declaration maybe changed to properly reference and
--               : connect this function to the design.  Unused inputs
--               : and outputs may 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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- DCM_PS: Digital Clock Manager Circuit for Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   DCM_PS_inst : DCM_PS
   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 2 to 32
      CLKIN_DIVIDE_BY_2 => FALSE, -- TRUE/FALSE to enable CLKIN divide by two feature
      CLKIN_PERIOD => 10.0, -- Specify period of input clock in ns from 1.25 to 1000.00
      CLKOUT_PHASE_SHIFT => "NONE", -- Specify phase shift mode of NONE, FIXED,
                                    -- VARIABLE_POSITIVE, VARIABLE_CENTER or DIRECT
      CLK_FEEDBACK => "1X",  -- Specify clock feedback of NONE or 1X
      DCM_PERFORMANCE_MODE => "MAX_SPEED",   -- Can be MAX_SPEED or MAX_RANGE
      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",   -- LOW, HIGH, or HIGH_SER frequency mode for DLL
      DUTY_CYCLE_CORRECTION => TRUE, -- Duty cycle correction, TRUE or FALSE
      FACTORY_JF => X"F0F0",          -- FACTORY JF Values Suggested to be set to X"F0F0"
      PHASE_SHIFT => 0,      -- Amount of fixed phase shift from -255 to 1023
      STARTUP_WAIT => FALSE) -- Delay configuration DONE until DCM LOCK, TRUE/FALSE
```

```
   port map (
      CLK0 => CLK0,          -- 0 degree DCM CLK ouptput
      CLK180 => CLK180,      -- 180 degree DCM CLK output
      CLK270 => CLK270,      -- 270 degree DCM CLK output
      CLK2X => CLK2X,        -- 2X DCM CLK output
      CLK2X180 => CLK2X180, -- 2X, 180 degree DCM CLK out
      CLK90 => CLK90,        -- 90 degree DCM CLK output
      CLKDV => CLKDV,        -- Divided DCM CLK out (CLKDV_DIVIDE)
      CLKFX => CLKFX,        -- DCM CLK synthesis out (M/D)
      CLKFX180 => CLKFX180, -- 180 degree CLK synthesis out
      DO => DO,              -- 16-bit data output for Dynamic Reconfiguration Port (DRP)
      LOCKED => LOCKED,      -- DCM LOCK status output
      PSDONE => PSDONE,      -- Dynamic phase adjust done output
      CLKFB => CLKFB,        -- DCM clock feedback
      CLKIN => CLKIN,        -- Clock input (from IBUFG, BUFG or DCM)
      PSCLK => PSCLK,        -- Dynamic phase adjust clock input
      PSEN => PSEN,          -- Dynamic phase adjust enable input
      PSINCDEC => PSINCDEC, -- Dynamic phase adjust increment/decrement
      RST => RST             -- DCM asynchronous reset input
   );

   -- End of DCM_PS_inst instantiation
```

Verilog Instantiation Template

```
//   DCM_PS   : 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_PS_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 may be removed or commented out.

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

   // DCM_PS: Digital Clock Manager Circuit for Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   DCM_PS #(
      .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(10.0), // Specify period of input clock in ns from 1.25 to 1000.00
      .CLKOUT_PHASE_SHIFT("NONE"), // Specify phase shift mode of NONE, FIXED,
                                   // VARIABLE_POSITIVE, VARIABLE_CENTER or DIRECT
      .CLK_FEEDBACK("1X"),  // Specify clock feedback of NONE, 1X or 2X
      .DCM_PERFORMANCE_MODE("MAX_SPEED"), // Can be MAX_SPEED or MAX_RANGE
      .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"), // LOW, HIGH, or HIGH_SER frequency mode for DLL
      .DUTY_CYCLE_CORRECTION("TRUE"), // Duty cycle correction, TRUE or FALSE
      .FACTORY_JF(16'hf0f0), // FACTORY JF value suggested to be set to 16'hf0f0
      .PHASE_SHIFT(0), // Amount of fixed phase shift from -255 to 1023
      .STARTUP_WAIT("FALSE") // Delay configuration DONE until DCM LOCK, TRUE/FALSE
   ) DCM_PS_inst(
      .CLK0(CLK0),          // 0 degree DCM CLK output
      .CLK180(CLK180),      // 180 degree DCM CLK output
      .CLK270(CLK270),      // 270 degree DCM CLK output
      .CLK2X(CLK2X),        // 2X DCM CLK output
      .CLK2X180(CLK2X180), // 2X, 180 degree DCM CLK out
      .CLK90(CLK90),        // 90 degree DCM CLK output
      .CLKDV(CLKDV),        // Divided DCM CLK out (CLKDV_DIVIDE)
      .CLKFX(CLKFX),        // DCM CLK synthesis out (M/D)
      .CLKFX180(CLKFX180), // 180 degree CLK synthesis out
      .DO(DO),              // 16-bit data output for Dynamic Reconfiguration Port (DRP)
      .LOCKED(LOCKED),      // DCM LOCK status output
      .PSDONE(PSDONE),      // Dynamic phase adjust done output
      .CLKFB(CLKFB),        // DCM clock feedback
      .CLKIN(CLKIN),        // Clock input (from IBUFG, BUFG or DCM)
      .PSCLK(PSCLK),        // Dynamic phase adjust clock input
      .PSEN(PSEN),          // Dynamic phase adjust enable input
      .PSINCDEC(PSINCDEC), // Dynamic phase adjust increment/decrement
      .RST(RST)             // DCM asynchronous reset input
   );
```

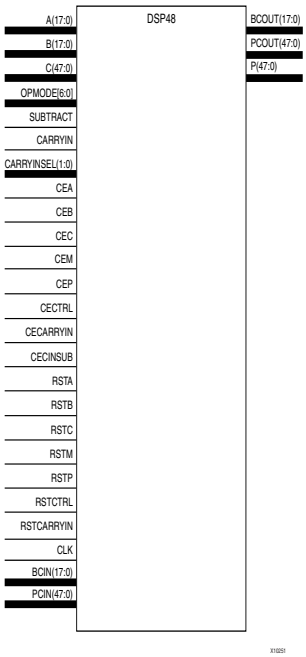
```
// End of DCM_PS_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

DSP48

Primitive: 18x18 Signed Multiplier Followed by a Three-Input Adder with Optional Pipeline Registers



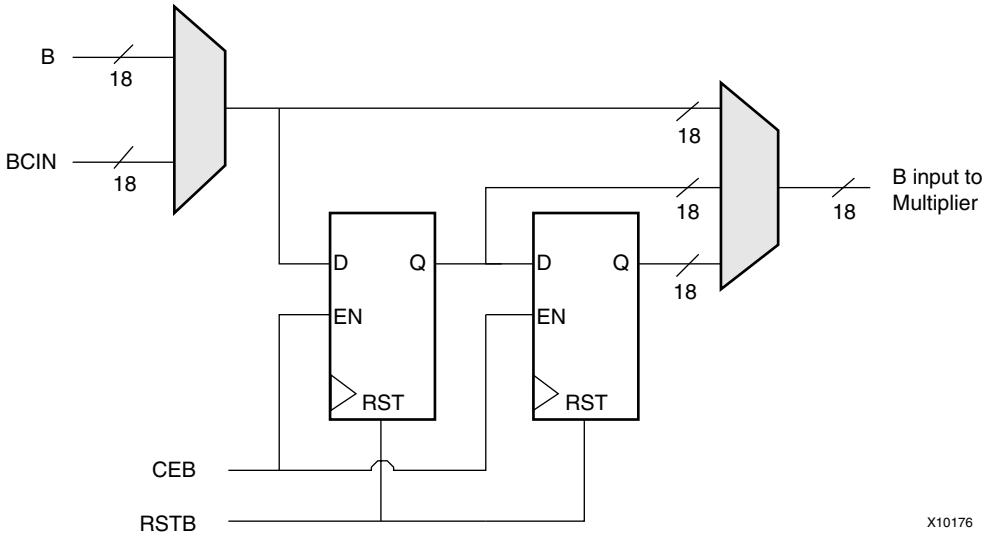
The DSP48 slice has a 48-bit output and is primarily intended for use in digital-signal processing applications. However, the flexibility of this component means that it can be applied to many more applications than a typical MACC unit.

A basic DSP48 slice consists of a multiplier followed by an adder. The multiplier accepts two, 18-bit, signed, two's complement operands producing a 36-bit, signed, twos complement result. The result is sign extended to 48 bits. The adder accepts three, 48-bit, signed, twos' complement operands producing a 48-bit, singed, twos' complement result.

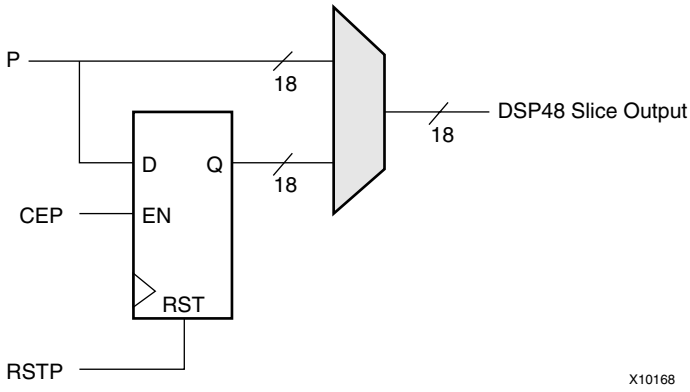
Possible operands for the adder include the multiplier output and external source or the registered output of the adder providing an accumulate function. The 48-bit output allows for 4096 accumulations of 36-bit operands before overflow occurs.

DSP48 Input and Output Signals

Signal Name	Direction	Size	Function
CLK	I	1	The DSP48 clock
A	I	18	The multiplier's A input, can also be used as adder's MSW input
B	I	18	The multiplier's B input, can also be used as adder's LSW input
BCIN	I	18	The multiplier's cascaded B input, can also be used as adder's LSW input
C	I	48	The adder's C input
PCIN	I	48	Cascaded adder's C Input from previous DSP slice
CARRYIN	I	1	The adders carry input
SUBTRACT	I	1	0= add, 1= (C, PCIN)-(mult,A:B)
OPMODE	I	7	Controls input to adder in DSP48 slices- see OPMODE table
CARRYINSEL	I	2	Selects carry source - see CARRINSEL table
CEA	I	1	Clock enable - 0=hold 1=enable AREG
CEB	I	1	Clock enable - 0=hold 1=enable BREG
CEC	I	1	Clock enable - 0=hold 1=enable CREG
CEP	I	1	Clock enable - 0=hold 1=enable PREG



B Input Logic



P Output Logic

Synthesis Attributes Used to Define Pipeline Registers

The following table describes the synthesis attributes used to define the pipeline registers.

Attribute	Function
AREG	0=bypass, 1=single, 2=dual
BREG	0=bypass, 1=single, 2=dual
CREG	0=bypass, 1=single
PREG	0=bypass, 1=single
MREG	0=bypass, 1=single
SUBTRACTREG	0=bypass, 1=single
OPMODEREG	0=bypass, 1=single
CARRYINSELREG	0=bypass, 1=single

Twos' Complement Signed Multiplier

The multiplier inside the DSP48 slice is an 18-bit x 18-bit twos' complement multiplier with a 36-bit signed twos' complement result. Cascading of multipliers to achieve larger products is supported. Applications such as signed-signed, signed-unsigned, and unsigned-unsigned multiplication, logical, arithmetic, barrel-shifter, twos' complement and magnitude return are easily implemented. There are two independent dynamic data input ports. The input ports can represent 18-bit signed or 17-bit unsigned data.

X, Y, and Z Multiplexers

The Operational Mode (OpMode) inputs provide a way for the design to change its functionality on the fly. For example, the loading of an accumulator to restart an accumulation process. The OpMode bits can be optionally registered under the control of the configuration RAM.

The following tables list the possible values of OpMode and resulting function at the outputs of the three multiplexers supplying data to the adder/subtracter. The 7-bit OpMode control can be further broken down into multiplexer select bits. Not all possible combinations for the multiplexer select bits are allowed. If the multiplier output is selected then both the X and Y multiplexer are consumed with the multiplier output.

OpMode Control Bit Select X, Y, and Z Multiplexer Outputs

OPMODE Binary			X Multiplexer Output Fed to Add/Subtract
Z	Y	X	
XXX	XX	00	ZERO (Default)
XXX	01	01	Multiplier Output
XXX	XX	10	P
XXX	XX	11	A concatenated B

OpMode Control Bit Select X, Y, and Z Multiplexer Outputs

OPMODE Binary			Y Multiplexer Output Fed to Add/Subtract
Z	Y	X	
XXX	00	XX	ZERO (Default)
XXX	01	01	Multiplier Output
XXX	10	XX	Illegal selection
XXX	11	XX	C

OpMode Controls X, Y, and Z Multiplexer Outputs

OPMODE Binary			Z Multiplexer Output Fed to Add/Subtract
Z	Y	X	
00	XX	XX	ZERO (Default)
001	XX	XX	PCIN
010	XX	XX	P
011	XX	XX	C
100	XX	XX	Illegal selection
101	XX	XX	Shift (PCIN)
110	XX	XX	Shift (P)
111	XX	XX	Illegal selection

Three Input Adder/Subtracter Control Logic

The adder/subtracter output is a function of control and data inputs. The OpMode, as shown in the previous section, selects the inputs to the X, Y, Z multiplexer that are directed to the three adder/subtracter inputs. It also described that when the multiplier output is selected, both X and Y multiplexers are occupied. With the inputs to the adder/subtracter specified the function of the adder/subtracter itself must be examined. As with the input multiplexers, the OpMode bits specify a portion of this function. The table below shows this function. The symbol ± in the table means either add or subtract and is specified by the state of the subtract control.

Hex OpMode	Binary OpMode	Output of Adder/Subtracter	Operation Description
[6:0]	Z Y X		
0x00	000 00 00	±CIN	Zero
0x02	000 00 10	±(P + CIN)	Hold P
0x03	000 00 11	±(A:B + CIN)	A:B select
0x05	000 01 01	± (A ×B+CIN)	Multiply
0x0c	000 11 00	± (C + CIN)	C select
0x0e	000 11 10	± (C + P + CIN)	Feedback add
0x0f	000 11 11	± (A:B +C +CIN)	36-bit adder
0x10	001 00 00	PCIN ± CIN	P cascade select
0x12	001 00 10	PCIN ± (P + CIN)	P cascade feedback add
0x13	001 00 11	PCIN ±(A:B+CIN)	P cascade add
0x15	001 01 01	PCIN ±(A ×B+CIN)	P cascade multiply add
0x1c	001 11 00	PCIN ±(C+CIN)	P cascade add
0x1e	001 11 10	PCIN ±(C+P+ CIN)	P cascade feedback add add
0x1c	001 11 11	PCIN ±(A:B+C + CIN)	P cascade add add
0x20	010 00 00	P±CIN	Hold P
0x22	010 00 10	P±(P +CIN)	Double feedback add
0x23	010 00 11	P±(A:B +CIN)	Feedback add
0x25	010 01 01	P±(A ×B+ CIN)	Multiply-accumulate

Hex OpMode [6:0]	Binary OpMode Z Y X	Output of Adder/Subtracter	Operation Description
0x2c	010 11 00	P±(C +CIN)	Feedback add
0x2e	010 11 10	P±(C +P+CIN)	Double feedback add
0x2f	010 11 11	P±(A:B +C+CIN)	Feedback add add
0x30	011 00 00	C±CIN	C Select
0x32	011 00 10	C±(P +CIN)	Feedback add
0x33	011 00 11	C±(A:B + CIN)	36-bit adder
0x35	011 01 01	C±(A ×B+CIN)	Multiply add
0x3c	011 11 00	C±(C + CIN)	Double
0x3e	011 11 10	C ± (C +P +CIN)	Double add feedback add
0x3f	011 11 11	C±(A:B + C+ CIN)	Double add
0x50	101 00 00	Shift(PCIN)± CIN	17-bit shift P cascade select
0x52	101 00 10	Shift(PCIN)±(P +CIN)	17-bit shift P cascade feedback add
0x53	101 00 11	Shift(PCIN)±(A:B +CIN)	17-bit shift P cascade add
0x55	101 01 01	Shift(PCIN)± (A ×B+ CIN)	17-bit shift P cascade multiply add
0x5c	101 11 00	Shift(PCIN)±(C +CIN)	17-bit shift P cascade add
0x5e	101 11 10	Shift(PCIN)±(C +P+CIN)	17-bit shift P cascade feedback add add
0x5c	101 11 11	Shift(PCIN) ±(A:B+C +CIN)	17-bit shift P cascade add add
0x60	110 00 00	Shift(P) ± CIN	17-bit shift feedback
0x62	110 00 10	Shift(P) ± (P + CIN)	17-bit shift feedback feedback add
0x63	110 00 11	Shift(P) ± (A:B + CIN)	17-bit shift feedback add
0x65	110 01 01	Shift(P) ± (A ×B+CIN)	17-bit shift feedback multiply add
0x6c	110 11 00	Shift(P) ± (C + CIN)	17-bit shift feedback add
0x6e	110 11 10	Shift(P) ± (C + P + CIN)	17-bit shift feedback feedback add add
0x6f	110 11 11	Shift(P) ± (A:B + C + CIN)	17-bit shift feedback add add

Rounding Modes Supported by Carry Logic

In addition to the OpMode inputs, the data inputs to the three input adder/subtracter, and the subtract control bit, the adder/subtracter output is a result of the carry-input logic.

CarryInSel signals, the Subtract control signal, and the OpMode control signals can be optionally registered under the control of the configuration RAM (denoted by the grey colored multiplexer symbol). This allows the control signals pipeline delay to match the pipeline delay for data in the design. The CarryInSel signals, the Subtract control signal, and the OpMode control signals share a common reset signal (RSTCTRL) and the Subtract control signal, and the OpMode control signals share a

common clock enable signal. The clock enable allows control signals to stall along with data when needed.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
AREG	INTEGER	0, 1, or 2	1	Number of pipeline registers on the A input, 0, 1 or 2
B_INPUT	STRING	"DIRECT" or "CASCADE"	"DIRECT"	"DIRECT"=multiplicand is B; "CASCADE"=multiplicant is BCIN.
BREG	INTEGER	0, 1, or 2	1	Number of pipeline registers on the B input, 0, 1 or 2.
CARRYINREG	INTEGER	0 OR 1	1	Number of pipeline registers for the CARRYIN input.
CARRYINSELREG	INTEGER	0 or 1	1	Number of pipeline registers for the CARRYINSEL.
CREG	INTEGER	0, 1, or 2	1	Number of pipeline registers on the C input, 0 or 1.
LEGACY_MODE	STRING	"NONE", "MULT18X18" or "MULT18X18S"	"MULT18X18S"	An internal attribute setting for the DCM. It should not be modified from the default value unless instructed by Xilinx
MREG	INTEGER	0 or 1	1	Number of multiplier pipeline registers, 0 or 1
OPMODEREG	INTEGER	0 or 1	1	Number of pipeline regsiters on OPMODE input, 0 or 1.
PREG	INTEGER	0 or 1	1	Number of pipeline registers on the P output, 0 or 1.
SUBTRACTREG	INTEGER	0 or 1	1	Number of pipeline registers on the SUBTRACT input, 0 or 1.

Usage

This design element is supported for schematics, instantiations, and inference. It is suggested that you use the Architecture Wizard in order to properly create instantiation code for the DSP48 block.

VHDL Instantiation Template

```
--    DSP48     : In order to incorporate this function into the design,
--    VHDL      : the following instance declaration needs to be placed
--  instance    : in the body of the design code.  The instance name
-- declaration  : (DSP48_inst) and/or the port declarations after the
--    code      : "=>" declaration maybe changed to properly reference and
--              : connect this function to the design.  Unused inputs
--              : and outputs can be 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 will be used
--              : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- DSP48: DSP Function Block
   --        Virtex-4
```

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

   DSP48_inst : DSP48
   generic map (
      AREG => 1,  -- Number of pipeline registers on the A input, 0, 1 or 2
      BREG => 1,  -- Number of pipeline registers on the B input, 0, 1 or 2
      B_INPUT => "DIRECT", -- B input DIRECT from fabric or CASCADE from another DSP48
      CARRYINREG => 1,     -- Number of pipeline registers for the CARRYIN input, 0 or 1
      CARRYINSELREG => 1,  -- Number of pipeline registers for the CARRYINSEL, 0 or 1
      CREG => 1,           -- Number of pipeline registers on the C input, 0 or 1
      LEGACY_MODE => "MULT18X18S", -- Backward compatibility, NONE, MULT18X18 or MULT18X18S
      MREG => 1,       -- Number of multiplier pipeline registers, 0 or 1
      OPMODEREG => 1, -- Number of pipeline regsiters on OPMODE input, 0 or 1
      PREG => 1,       -- Number of pipeline registers on the P output, 0 or 1
      SUBTRACTREG => 1) -- Number of pipeline registers on the SUBTRACT input, 0 or 1
   port map (
      BCOUT => BCOUT,  -- 18-bit B cascade output
      P => P,          -- 48-bit product output
      PCOUT => PCOUT,  -- 48-bit cascade output
      A => A,          -- 18-bit A data input
      B => B,          -- 18-bit B data input
      BCIN => BCIN,    -- 18-bit B cascade input
      C => C,          -- 48-bit cascade input
      CARRYIN => CARRYIN,       -- Carry input signal
      CARRYINSEL => CARRYINSEL, -- 2-bit carry input select
      CEA => CEA,      -- A data clock enable input
      CEB => CEB,      -- B data clock enable input
      CEC => CEC,      -- C data clock enable input
      CECARRYIN => CECARRYIN, -- CARRYIN clock enable input
      CECINSUB => CECINSUB,   -- CINSUB clock enable input
      CECTRL => CECTRL, -- Clock Enable input for CTRL regsitersL
      CEM => CEM,       -- Clock Enable input for multiplier regsiters
      CEP => CEP,       -- Clock Enable input for P regsiters
      CLK => CLK,       -- Clock input
      OPMODE => OPMODE, -- 7-bit operation mode input
      PCIN => PCIN,     -- 48-bit PCIN input
      RSTA => RSTA,     -- Reset input for A pipeline registers
      RSTB => RSTB,     -- Reset input for B pipeline registers
      RSTC => RSTC,     -- Reset input for C pipeline registers
      RSTCARRYIN => RSTCARRYIN, -- Reset input for CARRYIN registers
      RSTCTRL => RSTCTRL, -- Reset input for CTRL registers
      RSTM => RSTM,       -- Reset input for multiplier registers
      RSTP => RSTP,       -- Reset input for P pipeline registers
      SUBTRACT => SUBTRACT -- SUBTRACT input
   );

   -- End of DSP48_inst instantiation
```

Verilog Instantiation Template

```
//    DSP48    : 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 : (DSP48_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---->

   // DSP48: DSP Function Block
   //        Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   DSP48 #(
      .AREG(1),           // Number of pipeline registers on the A input, 0, 1 or 2
      .BREG(1),           // Number of pipeline registers on the B input, 0, 1 or 2
      .B_INPUT("DIRECT"), // B input DIRECT from fabric or CASCADE from another DSP48
      .CARRYINREG(1),     // Number of pipeline registers for the CARRYIN input, 0 or 1
      .CARRYINSELREG(1),  // Number of pipeline registers for the CARRYINSEL, 0 or 1
      .CREG(1),           // Number of pipeline registers on the C input, 0 or 1
      .LEGACY_MODE("MULT18X18S"), // Backward compatibility, NONE, MULT18X18 or MULT18X18S
      .MREG(1),           // Number of multiplier pipeline registers, 0 or 1
      .OPMODEREG(1),      // Number of pipeline regsiters on OPMODE input, 0 or 1
      .PREG(1),           // Number of pipeline registers on the P output, 0 or 1
      .SUBTRACTREG(1)     // Number of pipeline registers on the SUBTRACT input, 0 or 1
   ) DSP48_inst (
```

```
    .BCOUT(BCOUT),  // 18-bit B cascade output
    .P(P),          // 48-bit product output
    .PCOUT(PCOUT),  // 48-bit cascade output
    .A(A),          // 18-bit A data input
    .B(B),          // 18-bit B data input
    .BCIN(BCIN),    // 18-bit B cascade input
    .C(C),          // 48-bit cascade input
    .CARRYIN(CARRYIN), // Carry input signal
    .CARRYINSEL(CARRYINSEL), // 2-bit carry input select
    .CEA(CEA),      // A data clock enable input
    .CEB(CEB),      // B data clock enable input
    .CEC(CEC),      // C data clock enable input
    .CECARRYIN(CECARRYIN), // CARRYIN clock enable input
    .CECINSUB(CECINSUB), // CINSUB clock enable input
    .CECTRL(CECTRL), // Clock Enable input for CTRL regsitersL
    .CEM(CEM),       // Clock Enable input for multiplier regsiters
    .CEP(CEP),       // Clock Enable input for P regsiters
    .CLK(CLK),       // Clock input
    .OPMODE(OPMODE), // 7-bit operation mode input
    .PCIN(PCIN),     // 48-bit PCIN input
    .RSTA(RSTA),     // Reset input for A pipeline registers
    .RSTB(RSTB),     // Reset input for B pipeline registers
    .RSTC(RSTC),     // Reset input for C pipeline registers
    .RSTCARRYIN(RSTCARRYIN), // Reset input for CARRYIN registers
    .RSTCTRL(RSTCTRL), // Reset input for CTRL registers
    .RSTM(RSTM), // Reset input for multiplier registers
    .RSTP(RSTP), // Reset input for P pipeline registers
    .SUBTRACT(SUBTRACT) // SUBTRACT input
);

// End of DSP48_inst instantiation
```

For More Information

Consult the *XtremeDSP Design Considerations User Guide*.

EMAC

Primitive: Fully integrated 10/100/1000 Mb/s Ethernet Media Access Controller (Ethernet MAC)

The Virtex-4 Tri-mode Ethernet Media Access Controller (Ethernet MAC) provides Ethernet connectivity to the Virtex-4 PowerPC™ Processor. The Ethernet MAC (EMAC) supports the following feature:

- Fully integrated 10/100/1000 Mb/s Ethernet MAC
- Complies with the IEEE 802.3-2002 specification
- Configurable full- or half-duplex operation
- Media Independent Interface (MII) Management (MDIO) interface to manage objects in the Physical (PHY) layer
- User accessable raw statistics vector outputs
- Supports VLAN frames
- Configurable inter-frame gap adjustment
- Configurable in-band Frame Check Sequence (FCS) field passing on both transmit and receive paths
- Provides auto pad on transmit and FCS field stripping on receive
- Configured and monitored through a host interface
- Hardware selectable Device Control Register (DCR) bus or 1G Ethernet MAC bus host interface
- Configurable flow control through Ethernet MAC Control PAUSE frames; symmetrically or asymmetrically enabled
- Configurable support for jumbo frames of any length
- Configurable receive address filter for unicast, multicast, and broadcast addresses
- Media Independent Interface (MII), Gigabit Media Independent Interface (GMII), and Reduced Gigabit Media Independent Interface (RGMII)
- Includes a 1000BASE-X Physical Coding Sublayer (PCS) and a Physical Medium Attachment (PMA) sublayer for use with the Multi-gigabit Transceiver (MGT) to provide a complete on-chip 1000BASE-X implementation
- Serial Gigabit Media Independent Interface (SGMII) supported through MGT interface to external copper PHY layer

For complete information about the Ethernet MAC in Virtex-4 devices, see the following documents:

- Virtex-4 Datasheet
- Virtex-4 Tri-mode Ethernet Media Access Controller (Ethernet MAC) User Guide

Port List and Definitions

Inputs	Outputs
RESET	
TIEEMAC0CONFIGVEC [79:0]	
TIEEMAC1CONFIGVEC [79:0]	
TIEEMAC0UNICASTADDR [47:0]	
TIEEMAC1UNICASTADDR [47:0]	
PHYEMAC0GTXCLK	
PHYEMAC1GTXCLK	
CLIENTEMAC0DCMLOCKED	EMAC0CLIENTANINTERRUPT
CLIENTEMAC1DCMLOCKED	EMAC1CLIENTANINTERRUPT
CLIENTEMAC0RXCLIENTCLKIN	EMAC0CLIENTRXCLIENTCLKOUT
	EMAC0CLIENTRXD [15:0]
	EMAC0CLIENTRXDVLD
	EMAC0CLIENTRXDVLDMSW
	EMAC0CLIENTRXGOODFRAME
	EMAC0CLIENTRXBADFRAME
	EMAC0CLIENTRXFRAMEDROP
	EMAC0CLIENTRXDVREG6
	EMAC0CLIENTRXSTATS [6:0]
	EMAC0CLIENTRXSTATSBYTEVLD
	EMAC0CLIENTRXSTATSVLD
CLIENTEMAC1RXCLIENTCLKIN	EMAC1CLIENTRXCLIENTCLKOUT
	EMAC1CLIENTRXD [15:0]
	EMAC1CLIENTRXDVLD
	EMAC1CLIENTRXDVLDMSW
	EMAC1CLIENTRXGOODFRAME
	EMAC1CLIENTRXBADFRAME
	EMAC1CLIENTRXFRAMEDROP
	EMAC1CLIENTRXDVREG6
	EMAC1CLIENTRXSTATS [6:0]
	EMAC1CLIENTRXSTATSBYTEVLD
	EMAC1CLIENTRXSTATSVLD
CLIENTEMAC0TXGMIIMIICLKIN	EMAC0CLIENTTXGMIIMIICLKOUT
CLIENTEMAC0TXCLIENTCLKIN	EMAC0CLIENTTXCLIENTCLKOUT
CLIENTEMAC0TXD [15:0]	EMAC0CLIENTTXACK
CLIENTEMAC0TXDVLD	EMAC0CLIENTTXCOLLISION
CLIENTEMAC0TXDVLDMSW	EMAC0CLIENTTXRETRANSMIT
CLIENTEMAC0TXUNDERRUN	EMAC0CLIENTTXSTATS
CLIENTEMAC0TXIFGDELAY [7:0]	EMAC0CLIENTTXSTATSBYTEVLD
CLIENTEMAC0TXFIRSTBYTE	EMAC0CLIENTTXSTATSVLD
CLIENTEMAC1TXGMIIMIICLKIN	EMAC1CLIENTTXGMIIMIICLKOUT

Inputs	Outputs
CLIENTEMAC1TXCLIENTCLKIN	EMAC1CLIENTTXCLIENTCLKOUT
CLIENTEMAC1TXD [15:0]	EMAC1CLIENTTXACK
CLIENTEMAC1TXDVLD	EMAC1CLIENTTXCOLLISION
CLIENTEMAC1TXDVLDMSW	EMAC1CLIENTTXRETRANSMIT
CLIENTEMAC1TXUNDERRUN	EMAC1CLIENTTXSTATS
CLIENTEMAC1TXIFGDELAY [7:0]	EMAC1CLIENTTXSTATSBYTEVLD
CLIENTEMAC1TXFIRSTBYTE	EMAC1CLIENTTXSTATSVLD
CLIENTEMAC0PAUSEREQ	
CLIENTEMAC0PAUSEVAL [15:0]	
CLIENTEMAC1PAUSEREQ	
CLIENTEMAC1PAUSEVAL [15:0]	
HOSTADDR [9:0]	HOSTMIIMRDY
HOSTCLK	HOSTRDDATA [31:0]
HOSTMIIMSEL	
HOSTOPCODE [1:0]	
HOSTREQ	
HOSTWRDATA [31:0]	
HOSTEMAC1SEL	
DCREMACCLK	DCRHOSTDONEIR
DCREMACENABLE	EMACDCRACK
DCREMACDBUS [0:31]	EMACDCRDBUS [0:31]
DCREMACABUS [8:9]	
DCREMACREAD	
DCREMACWRITE	
PHYEMAC0RXCLK	EMAC0PHYTXCLK
PHYEMAC0RXD [7:0]	EMAC0PHYTXD [7:0]
PHYEMAC0RXDV	EMAC0PHYTXEN
PHYEMAC0RXER	EMAC0PHYTXER
PHYEMAC0MIITXCLK	
PHYEMAC0COL	
PHYEMAC0CRS	
PHYEMAC1RXCLK	EMAC1PHYTXCLK
PHYEMAC1RXD [7:0]	EMAC1PHYTXD [7:0]
PHYEMAC1RXDV	EMAC1PHYTXEN
PHYEMAC1RXER	EMAC1PHYTXER
PHYEMAC1MIITXCLK	
PHYEMAC1COL	
PHYEMAC1CRS	
PHYEMAC0SIGNALDET	EMAC0PHYENCOMMAALIGN
PHYEMAC0PHYAD [4:0]	EMAC0PHYLOOPBACKMSB
PHYEMAC0RXCLKCORCNT [2:0]	EMAC0PHYMGTRXRESET

Inputs	Outputs
PHYEMAC0RXBUFSTATUS [1:0]	EMAC0PHYMGTTXRESET
PHYEMAC0RXCHARISCOMMA	EMAC0PHYPOWERDOWN
PHYEMAC0RXCHARISK	EMAC0PHYSYNCACQSTATUS
PHYEMAC0RXCHECKINGCRC	EMAC0PHYTXCHARDISPMODE
PHYEMAC0RXCOMMADET	EMAC0PHYTXCHARDISPVAL
PHYEMAC0RXDISPERR	EMAC0PHYTXCHARISK
PHYEMAC0RXLOSSOFSYNC [1:0]	
PHYEMAC0RXNOTINTABLE	
PHYEMAC0RXRUNDISP	
PHYEMAC0RXBUFERR	
PHYEMAC0TXBUFERR	
PHYEMAC1SIGNALDET	EMAC1PHYENCOMMAALIGN
PHYEMAC1PHYAD [4:0]	EMAC1PHYLOOPBACKMSB
PHYEMAC1RXCLKCORCNT [2:0]	EMAC1PHYMGTRXRESET
PHYEMAC1RXBUFSTATUS [1:0]	EMAC1PHYMGTTXRESET
PHYEMAC1RXCHARISCOMMA	EMAC1PHYPOWERDOWN
PHYEMAC1RXCHARISK	EMAC1PHYSYNCACQSTATUS
PHYEMAC1RXCHECKINGCRC	EMAC1PHYTXCHARDISPMODE
PHYEMAC1RXCOMMADET	EMAC1PHYTXCHARDISPVAL
PHYEMAC1RXDISPERR	EMAC1PHYTXCHARISK
PHYEMAC1RXLOSSOFSYNC [1:0]	
PHYEMAC1RXNOTINTABLE	
PHYEMAC1RXRUNDISP	
PHYEMAC1RXBUFERR	
PHYEMAC1TXBUFERR	
PHYEMAC0MCLKIN	EMAC0PHYMCLKOUT
PHYEMAC0MDIN	EMAC0PHYMDOUT
	EMAC0PHYMDTRI
PHYEMAC1MCLKIN	EMAC1PHYMCLKOUT
PHYEMAC1MDIN	EMAC1PHYMDOUT
	EMAC1PHYMDTRI

Usage

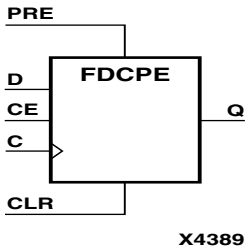
Refer to the Embedded Tri-mode Ethernet MAC Wrapper from the CORE Generator Tool for information regarding the use of this component.

For More Information

Consult the *Virtex-4 Tri-Mode Ethernet Media Access Controller (Ethernet MAC) User Guide*.

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.

ForVirtex-4 devices, the power on condition can be simulated by applying a High-level pulse on the GSR net.

The active level of the GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_VIRTEX4 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 is inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes must be applied.

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 will be used
--             : for simulation.

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

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

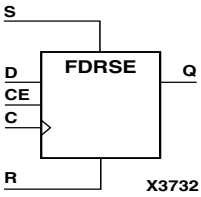
```
//     FDCPE    : In order to incorporate this function into the design,
//    Verilog   : the following instance declaration needs to be placed
//   instance   : in the body of the design code.  The instance name
//  declaration : (FDCPE_inst) and/or the port declarations within the
//     code     : 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 *Virtex-4 Configuration Guide*.

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. 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, when power is applied.

For Virtex-4 devices, the power on condition can be simulated by applying a High-level pulse on the GSR net.

The active level of the GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_VIRTEX4 symbol.

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 is inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes must 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

```
--    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 will be used
--             : for simulation.

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

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

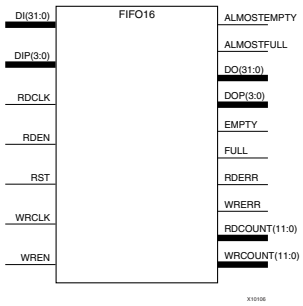
```
//     FDRSE    : In order to incorporate this function into the design,
//    Verilog   : the following instance declaration needs to be placed
//   instance   : in the body of the design code.  The instance name
//  declaration : (FDCRS_inst) and/or the port declarations within the
//     code     : 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 *Virtex-4 Configuration Guide*.

FIFO16

Primitive: Virtex-4 Block RAM based built-in FIFO



A large percentage of FPGA designs implement FIFOs using block RAMs. In the Virtex-4 architecture, additional dedicated logic in the block RAM enables users to easily implement synchronous or asynchronous FIFOs. This eliminates the need to use additional CLB logic for counter, comparator, or status flag generation and uses just one block RAM resource per FIFO. Both standard and first-word fall-through (FWFT) modes are supported.

The supported configurations are 4K x 4, 2K x 9, 1K x 18, and 512 x 36.

The block RAM can be configured as an asynchronous first-in/first-out (FIFO) memory with independent read and write clocks for either synchronous or asynchronous operation. Port A of the block RAM is used as a FIFO read port, and Port B is a FIFO write port. Data is read from the FIFO on the rising edge of read clock and written to the FIFO on the rising edge of write clock. Independent read and write port width selection is not supported in FIFO mode.

The available status flags are:

- Full (FULL): Synchronous to WRCLK
- Empty (EMPTY): Synchronous to RDCLK
- Almost Full (AFULL): Synchronous to WRCLK
- Almost Empty (AEMPTY): Synchronous to RDCLK
- Write Count (WRCOUNT): Synchronous to WRCLK
- Write Error (WRERR): Synchronous to WRCLK
- Read Count (RDCOUNT): Synchronous to RDCLK
- Read Error (RDERR): Synchronous to RDCLK

The following table shows the FIFO capacity in the two modes:

FIFO Capacity Standard Mode	FWFT Mode
4k+1 entries by 4 bits	4k+2 entries by 4 bits
2k+1 entries by 9 bits	2k+2 entries by 9 bits
1k+1 entries by 18 bits	1k+2 entries by 18 bits
512+1 entries by 36 bits	512+2 entries by 36 bits

Port Descriptions

FIFO I/O Port Names and Descriptions Port Name	Direction	Description
DI	Input	Data input
DIP	Input	Parity-bit input
WREN	Input	Write enable. When WREN = 1, data will be written to memory. When WREN = 0, write is disabled.
WRCLK	Input	Clock for write domain operation.
RDEN	Input	Read enable. When RDEN = 1, data will be read to output register. When RDEN = 0, read is disabled.
RDCLK	Input	Clock for read domain operation.
RESET	Input	Asynchronous reset of all FIFO functions, flags, and pointers.

FIFO I/O Port Names and Descriptions Port Name	Direction	Description
DO	Output	Data output, synchronous to RDCLK
DOP	Output	Parity-bit output, synchronous to RDCLK
FULL	Output	All entries in FIFO memory are filled.
ALMOSTFULL	Output	Almost all entries in FIFO memory have been filled. Synchronous to WRCLK. The value is user configurable.
EMPTY	Output	FIFO is empty. No additional read can be performed. Synchronous to RDCLK.
ALMOSTEMPTY	Output	Almost all valid entries in FIFO are read. Synchronous with RDCLK. The value is user configurable.
RDCOUNT	Output	The FIFO data read pointer. It is synchronous with RDCLK. The value will wrap around if the maximum read pointer value has been reached.
WRCOUNT	Output	The FIFO data write pointer. It is synchronous with WRCLK. The value will wrap around if the maximum write pointer value has been reached.
WRERR	Output	When the FIFO is full, any additional write operation generates an error flag. Synchronous with WRCLK.
RDERR	Output	When the FIFO is empty, any additional read operation generates an error flag. Synchronous with RDCLK.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ALMOST_EMPTY_OFFSET	12-Bit Hexadecimal	12-Bit Hexadecimal	12'h080	Sets the almost empty threshold.
ALMOST_FULL_OFFSET	12-Bit Hexadecimal	12-Bit Hexadecimal	12'h080	Sets almost full threshold.
DATA_WIDTH	INTEGER	4, 9, 18, 36	36	Sets data width to allowed value.
FIRST_WORD_FALL_THROUGH	BOOLEAN	FALSE, TRUE	FALSE	Sets the FIFO FWFT to "TRUE" or "FALSE."

Usage

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

Operating Mode

There are two operating modes in FIFO functions. They differ only in output behavior after the first word is written to a previously empty FIFO.

Standard Mode

After the first word is written into an empty FIFO, the Empty flag deasserts synchronously with RDCLK. After Empty is deasserted Low and RDEN is asserted, the first word will appear at DOUT on the rising edge of RDCLK.

First Word Fall Through Mode

After the first word is written into an empty FIFO, it automatically appears at DOUT after a few RDCLK cycles without asserting RDEN. Subsequent Read operations require Empty to be Low and RDEN to be High.

Status Flags

Full Flag

The Full flag is asserted when there are no more available entries in the FIFO queue. When the FIFO is full, the write pointer will be frozen. This ensures the read and write pointers point to the same entry and no overflow will occur. The Full flag is registered at the output and takes one write cycle to assert. The Full flag is deasserted three clock cycles after the last entry is read, and it is synchronous to WRCLK.

Write Error Flag

Once the Full flag has been asserted, any further write attempts will trigger the Write Error flag. The Write Error flag is deasserted when Write Enable or Full is deasserted Low. This signal is synchronous to WRCLK.

Almost Full Flag

The Almost Full flag is set when the FIFO has fewer than the number of available empty spaces specified by the ALMOST_FULL_OFFSET value. The Almost Full flag warns the user to stop writing. It deasserts when the number of empty spaces in the FIFO is greater than the ALMOST_FULL_OFFSET value, and is synchronous to WRCLK.

VHDL Instantiation Template

```
--    FIFO16     : 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 (FIFO16_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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- FIFO16: Virtex-4 512 deep x 36 wide BlockRAM Asynchrnous FIFO
   -- Xilinx  HDL Libraries Guide Version 8.1i

   FIFO16_inst : FIFO16
   generic map (
      ALMOST_FULL_OFFSET => X"080",  -- Sets almost full threshold
      ALMOST_EMPTY_OFFSET => X"080", -- Sets the almost empty threshold
      DATA_WIDTH => 36, -- Sets data width to 4, 9, 18, or 36
      FIRST_WORD_FALL_THROUGH => FALSE) -- Sets the FIFO FWFT to TRUE or FALSE
   port map (
      ALMOSTEMPTY => ALMOSTEMPTY,   -- 1-bit almost empty output flag
      ALMOSTFULL => ALMOSTFULL,     -- 1-bit almost full output flag
      DO => DO,                     -- 32-bit data output
      DOP => DOP,                   -- 4-bit parity data output
      EMPTY => EMPTY,               -- 1-bit empty output flag
```

```
      FULL => FULL,                 -- 1-bit full output flag
      RDCOUNT => RDCOUNT,           -- 12-bit read count output
      RDERR => RDERR,               -- 1-bit read error output
      WRCOUNT => WRCOUNT,           -- 12-bit write count output
      WRERR => WRERR,               -- 1-bit write error
      DI => DI,                     -- 32-bit data input
      DIP => DIP,                   -- 4-bit partity input
      RDCLK => RDCLK,               -- 1-bit read clock input
      RDEN => RDEN,                 -- 1-bit read enable input
      RST => RST,                   -- 1-bit reset input
      WRCLK => WRCLK,               -- 1-bit write clock input
      WREN => WREN                  -- 1-bit write enable input
   );

   -- End of FIFO16_inst instantiation
```

Verilog Template

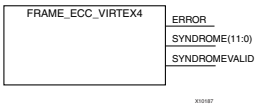
```
//     FIFO16    : In order to incorporate this function into the design,
//    Verilog    : the following instance declaration needs to be placed
//   instance    : in the body of the design code.  The instance name
//  declaration  : (FIFO16_512x36_inst) and/or the port declarations within the
//     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---->

   // FIFO16: Virtex-4 BlockRAM Asynchrnous FIFO configured for 512 deep x 36 wide
   // Xilinx HDL Libraries Guide Version 8.1i

   FIFO16 #(
      .ALMOST_FULL_OFFSET(12'h080),     // Sets almost full threshold
      .ALMOST_EMPTY_OFFSET(12'h080),    // Sets the almost empty threshold
      .DATA_WIDTH(36),                  // Sets data width to 4, 9, 18, or 36
      .FIRST_WORD_FALL_THROUGH("FALSE") // Sets the FIFO FWFT to "TRUE" or "FALSE"
   ) FIFO16_512x36_inst (
      .ALMOSTEMPTY(ALMOSTEMPTY), // 1-bit almost empty output flag
      .ALMOSTFULL(ALMOSTFULL),   // 1-bit almost full output flag
      .DO(DO),                   // 32-bit data output
      .DOP(DOP),                 // 4-bit parity data output
      .EMPTY(EMPTY),             // 1-bit empty output flag
      .FULL(FULL),               // 1-bit full output flag
      .RDCOUNT(RDCOUNT),         // 12-bit read count output
      .RDERR(RDERR),             // 1-bit read error output
      .WRCOUNT(WRCOUNT),         // 12-bit write count output
      .WRERR(WRERR),             // 1-bit write error
      .DI(DI),                   // 32-bit data input
      .DIP(DIP),                 // 4-bit partity input
      .RDCLK(RDCLK),             // 1-bit read clock input
      .RDEN(RDEN),               // 1-bit read enable input
      .RST(RST),                 // 1-bit reset input
      .WRCLK(WRCLK),             // 1-bit write clock input
      .WREN(WREN)                // 1-bit write enable input
   );

   // End of FIFO16_512x36_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

FRAME_ECC_VIRTEX4

Primitive: Reads a Single, Virtex-4 Configuration Frame and Computes a Hamming, Single-Error Correction, Double-Error Detection "Syndrome"



The FRAME_ECC_VIRTEX4 module reads a single Virtex-4 configuration frame of 1312-bits, 32-bits at a time. It will then compute a Hamming single error correction, double error detection "syndrome." This identifies the single frame bit (if any), which is in error and should be corrected. It also indicates the presence of two bit errors, which cannot be corrected. Note that the FRAME_ECC_VIRTEX4 primitive does not repair changed bits.

Port List and Definitions

Name	Type	Width	Function
ERROR	Output	1	Error Output
SYNDROME	Output	12	Indicates the location of the erroneous bit
SYNDROMEVALID	Output	1	When value is High, indicates the presence of zero, one or two bit errors in the frame

ERROR – Output

Indicates whether an error exists or not.

SYNDROME – Output

Provides the bit location of the error and whether zero, one, or two erroneous bits are present.

SYNDROMEVALID - Output

When asserted HIGH, SYNDROMEVALID indicates that the end of a frame readback.

Usage

In order to use the FRAME_ECC_VIRTEX4, this primitive must be instantiated in a design. Any readbacks must be performed through the SelectMAP, JTAG, or ICAP.

At the end of each frame readback, the SYNDROME_VALID pin will be asserted HIGH for one cycle of the readback clock (CCLK, TCK, or ICAP_CLK). The number of cycles required to read back a frame varies with the interface used.

When SYNDROME_VALID is asserted HIGH, the value on the SYNDROME pin indicates the presence of zero, one, or two bit errors in the frame. The following table summarizes the relationship of various SYNDROME value and error status.

Syndrome Value and Corresponding Error Status

Syndrome bit 11	Syndrome bit 10 to 0	Error Status
0	All 0s	No bit errors
0	Not equal to 0	One bit error, and syndrome value identifies the position of the erroneous bit
1	All 0s	Two bit errors, not correctable

Note: SYNDROME_VALID must be HIGH for the values on the table above to be useful.

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

VHDL Instantiation Template

```
--FRAME_ECC_VIRTEX4: 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 (FRAME_ECC_VIRTEX4_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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- FRAME_ECC_VIRTEX4: Configuration Frame Error Correction Circuitry
   --                    Virtex-4
   -- Xilinx HDL Libraries Guide version 8.1i

   FRAME_ECC_VIRTEX4_inst : FRAME_ECC_VIRTEX4
   port map (
      ERROR => ERROR,
      SYNDROME => SYNDROME,
      SYNDROMEVALID => SYNDROMEVALID
   );

   -- End of FRAME_ECC_VIRTEX4_inst instantiation
```

Verilog Instantiation Template

```
// FRAME_ECC_VIRTEX4 : 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     : (FRAME_ECC_VIRTEX4_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---->

// FRAME_ECC_VIRTEX4: Configuration Frame Error Correction Circuitry
//                    Virtex-4
// Xilinx HDL Libraries Guide Version 8.1i

   FRAME_ECC_VIRTEX4 FRAME_ECC_VIRTEX4_inst (
      .ERROR(ERROR),       // 1-bit output indicating an error
      .SYNDROME(SYNDROME), // 12-bit output location of erroroneous bit
      .SYNDROMEVALID(SYNDROMEVALID) //- 1-bit output indicating 0, 1 or 2 bit errors in frame
```

```
);
// End of FRAME_ECC_VIRTEX4_inst instantiation
```

For More Information

Consult the *Virtex-4 Configuration Guide*.

GT11_CUSTOM

Primitive: RocketIO MGTs with 622 Mb/s to 11.1 Gb/s Data Rates, 8 to 24 Transceivers per FPGA, and 2.5 GHz – 5.55 GHz VCO, Less Than 1ns RMS Jitter

RocketIO MGTs have flexible, programmable features that allow a multi-gigabit serial transceiver to be easily integrated into any Virtex-4 design. The RocketIO MGTs support the following features:

- 10.3 Gb/s data rates
- 8to 24 transceivers per FPGA
- 2.5 GHz – 5.55 GHz VCO, less than 1ns RMS jitter
- Transmitter pre-emphasis
- Receiver continuous time equalization
- On-chip AC coupled receiver, with optional by-pass
- Receiver signal detect and loss of signal indicator, out of band signal receiver
- Transmit driver idle state for out of band signaling-both outputs at Vcm
- 8B/10B or 64B/66B encoding, or no data encoding (pass through mode)
- Channel bonding
- Flexible Cyclic Redundancy Check (CRC) generation and checking
- Pins for transmitter and receiver termination voltage
- User reconfiguration using secondary (dynamic) configuration bus
- Multiple loopback paths including PMA RX-TX path

Inputs and Outputs

Inputs	Outputs
CHBONDI [4:0]	DRDY
CSUPMARESET	RXBUFERR
DADDR [7:0]	RXCALFAIL
DCLK	RXCOMMADET
DEN	RXCYCLELIMIT
DI [15:0]	RXLOCK
DWE	RXREALIGN
ENCHANSYNC	RXRECCLK1
ENMCOMMAALIGN	RXBCLK
ENPCOMMAALIGN	RXRECCLK2
GREFCLK	RXSIGDET
LOOPBACK [1:0]	TX1N
POWERDOWN	TX1P
REFCLK1	TXBUFERR
REFCLK2	TXCALFAIL
RX1N	TXCYCLELIMIT
RX1P	TXLOCK
RXBLOCKSYNC64B66BUSE	DO [15:0]
RXCLKSTABLE	RXLOSSOFSYNC [1:0]
RXCOMMADETUSE	RXCRCOUT [31:0]
RXCRCCLK	TXCRCOUT [31:0]
RXCRCDATAVALID	CHBONDO [4:0]
RXCRCDATAWIDTH [2:0]	RXSTATUS [5:0]
RXCRCIN [63:0]	RXDATA [63:0]
RXCRCINIT	RXCHARISCOMMA [7:0]
RXCRCINTCLK	RXCHARISK [7:0]
RXCRCPD	RXDISPERR [7:0]
RXCRCRESET	RXNOTINTABLE [7:0]
RXDATAWIDTH [1:0]	RXRUNDISP [7:0]
RXDEC64B66BUSE	TXRUNDISP [7:0]
RXDEC8B10BUSE	TXKERR [7:0]
RXDESCRAM64B66BUSE	
RXIGNOREBTF	
RXINTDATAWIDTH [1:0]	
RXPMARESET	
RXPOLARITY	
RXRESET	
RXSLIDE	
RXUSRCLK	
RXUSRCLK2	

Inputs	Outputs
TXBYPASS8B10B [7:0]	
TXCHARDISPMODE [7:0]	
TXCHARDISPVAL [7:0]	
TXCHARISK [7:0]	
TXCLKSTABLE	
TXCRCCLK	
TXCRCDATAVALID	
TXCRCDATAWIDTH [2:0]	
TXCRCIN [63:0]	
TXCRCINIT	
TXCRCINTCLK	
TXCRCPD	
TXCRCRESET	
TXDATA [63:0]	
TXDATAWIDTH [1:0]	
TXENC64B66BUSE	
TXENC8B10BUSE	
TXENOOB	
TXGEARBOX64B66BUSE	
TXINHIBIT	
TXINTDATAWIDTH [1:0]	
TXPMARESET	
TXPOLARITY	
TXRESET	
TXSCRAM64B66BUSE	
TXSYNC	
TXUSRCLK	
TXUSRCLK2	

Usage

Refer to the Architecture Wizard in the ISE software for information regarding the use of this component. If the Architecture Wizard is not used, two GT11 primitives should be instantiated with the combusout port connected t othe compbusin of the other GT11 instance.

VHDL and Verilog Instantiation

It is suggested that you use the *Architecture Wizard* in order to properly create instantiation code for the GT11_CUSTOM block.

For More Information

Consult the Virtex-4 Data Sheet and the *Virtex-4 RocketIO Transceiver User Guide*.

GT11_DUAL

Primitive: RocketIO MGT Tile (contains 2 GT11_CUSTOM) with · 622 Mb/s to 11.1 Gb/s data rates, · 8 to 24 transceivers per FPGA, and · 2.5 GHz – 5.55 GHz VCO, less than 1ns RMS jitter

RocketIO MGTs have flexible, programmable features that allow a multi-gigabit serial transceiver to be easily integrated into any Virtex-4 design. The RocketIO MGTs support the following features:

- 622 Mb/s to 11.1 Gb/s data rates
- 8 to 24 transceivers per FPGA
- 2.5 GHz – 5.55 GHz VCO, less than 1ns RMS jitter
- Transmitter pre-emphasis (pre-equalization)
- Receiver continuous time equalization
- On-chip AC coupled receiver
- Digital oversampled receiver for data rates up to 2.5 Gb/s
- Receiver signal detect and loss of signal indicator, out-of-band signal receiver
- Transmit driver idle state for out-of-band signaling, both outputs at Vcm
- 8B/10B or 64B/66B encoding, or no data encoding (pass through mode)
- Channel bonding
- Flexible Cyclic Redundancy Check (CRC) generation and checking
- Pins for transmitter and receiver termination voltage
- User reconfiguration using secondary (dynamic) configuration bus
- Multiple loopback paths including PMA RX-TX path

For complete information about the RocketIO MGTs in Virtex-4 devices, see the following documents:

- Virtex-4 Data Sheet
- *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide*

Inputs and Outputs

Inputs	Outputs
[1:0] LOOPBACK_A;	[1:0] RXLOSSOFSYNC_A;
[1:0] LOOPBACK_B;	[1:0] RXLOSSOFSYNC_B;
[1:0] RXDATAWIDTH_A;	[15:0] DO_A;
[1:0] RXDATAWIDTH_B;	[15:0] DO_B;
[1:0] RXINTDATAWIDTH_A;	[31:0] RXCRCOUT_A;
[1:0] RXINTDATAWIDTH_B;	[31:0] RXCRCOUT_B;
[1:0] TXDATAWIDTH_A;	[31:0] TXCRCOUT_A;
[1:0] TXDATAWIDTH_B;	[31:0] TXCRCOUT_B;
[1:0] TXINTDATAWIDTH_A;	[4:0] CHBONDO_A;
[1:0] TXINTDATAWIDTH_B;	[4:0] CHBONDO_B;
[15:0] DI_A;	[5:0] RXSTATUS_A;
[15:0] DI_B;	[5:0] RXSTATUS_B;
[2:0] RXCRCDATAWIDTH_A;	[63:0] RXDATA_A;
[2:0] RXCRCDATAWIDTH_B;	[63:0] RXDATA_B;
[2:0] TXCRCDATAWIDTH_A;	[7:0] RXCHARISCOMMA_A;
[2:0] TXCRCDATAWIDTH_B;	[7:0] RXCHARISCOMMA_B;
[4:0] CHBONDI_A;	[7:0] RXCHARISK_A;
[4:0] CHBONDI_B;	[7:0] RXCHARISK_B;
[63:0] RXCRCIN_A;	[7:0] RXDISPERR_A;
[63:0] RXCRCIN_B;	[7:0] RXDISPERR_B;
[63:0] TXCRCIN_A;	[7:0] RXNOTINTABLE_A;
[63:0] TXCRCIN_B;	[7:0] RXNOTINTABLE_B;
[63:0] TXDATA_A;	[7:0] RXRUNDISP_A;
[63:0] TXDATA_B;	[7:0] RXRUNDISP_B;
[7:0] DADDR_A;	[7:0] TXKERR_A;
[7:0] DADDR_B;	[7:0] TXKERR_B;
[7:0] TXBYPASS8B10B_A;	[7:0] TXRUNDISP_A;
[7:0] TXBYPASS8B10B_B;	[7:0] TXRUNDISP_B;
[7:0] TXCHARDISPMODE_A;	DRDY_A;
[7:0] TXCHARDISPMODE_B;	DRDY_B;
[7:0] TXCHARDISPVAL_A;	RXBUFERR_A;
[7:0] TXCHARDISPVAL_B;	RXBUFERR_B;
[7:0] TXCHARISK_A;	RXCALFAIL_A;
[7:0] TXCHARISK_B;	RXCALFAIL_B;
DCLK_A;	RXCOMMADET_A;
DCLK_B;	RXCOMMADET_B;
DEN_A;	RXCYCLELIMIT_A;
DEN_B;	RXCYCLELIMIT_B;
DWE_A;	RXLOCK_A;
DWE_B;	RXLOCK_B;

Inputs	Outputs
ENCHANSYNC_A;	RXMCLK_A;
ENCHANSYNC_B;	RXMCLK_B;
ENMCOMMAALIGN_A;	RXPCSHCLKOUT_A;
ENMCOMMAALIGN_B;	RXPCSHCLKOUT_B;
ENPCOMMAALIGN_A;	RXREALIGN_A;
ENPCOMMAALIGN_B;	RXREALIGN_B;
GREFCLK_A;	RXRECCLK1_A;
GREFCLK_B;	RXRECCLK1_B;
POWERDOWN_A;	RXRECCLK2_A;
POWERDOWN_B;	RXRECCLK2_B;
REFCLK1_A;	RXSIGDET_A;
REFCLK1_B;	RXSIGDET_B;
REFCLK2_A;	TX1N_A;
REFCLK2_B;	TX1N_B;
RX1N_A;	TX1P_A;
RX1N_B;	TX1P_B;
RX1P_A;	TXBUFERR_A;
RX1P_B;	TXBUFERR_B;
RXBLOCKSYNC64B66BUSE_A;	TXCALFAIL_A;
RXBLOCKSYNC64B66BUSE_B;	TXCALFAIL_B;
RXCLKSTABLE_A;	TXCYCLELIMIT_A;
RXCLKSTABLE_B;	TXCYCLELIMIT_B;
RXCOMMADETUSE_A;	TXLOCK_A;
RXCOMMADETUSE_B;	TXLOCK_B;
RXCRCCLK_A;	TXOUTCLK1_A;
RXCRCCLK_B;	TXOUTCLK1_B;
RXCRCDATAVALID_A;	TXOUTCLK2_A;
RXCRCDATAVALID_B;	TXOUTCLK2_B;
RXCRCINIT_A;	TXPCSHCLKOUT_A;
RXCRCINIT_B;	TXPCSHCLKOUT_B;
RXCRCINTCLK_A;	
RXCRCINTCLK_B;	
RXCRCPD_A;	
RXCRCPD_B;	
RXCRCRESET_A;	
RXCRCRESET_B;	
RXDEC64B66BUSE_A;	
RXDEC64B66BUSE_B;	
RXDEC8B10BUSE_A;	
RXDEC8B10BUSE_B;	
RXDESCRAM64B66BUSE_A;	

Inputs	Outputs
RXDESCRAM64B66BUSE_B;	
RXIGNOREBTF_A;	
RXIGNOREBTF_B;	
RXPMARESET_A;	
RXPMARESET_B;	
RXPOLARITY_A;	
RXPOLARITY_B;	
RXRESET_A;	
RXRESET_B;	
RXSLIDE_A;	
RXSLIDE_B;	
RXSYNC_A;	
RXSYNC_B;	
RXUSRCLK_A;	
RXUSRCLK_B;	
RXUSRCLK2_A;	
RXUSRCLK2_B;	
TXCLKSTABLE_A;	
TXCLKSTABLE_B;	
TXCRCCLK_A;	
TXCRCCLK_B;	
TXCRCDATAVALID_A;	
TXCRCDATAVALID_B;	
TXCRCINIT_A;	
TXCRCINIT_B;	
TXCRCINTCLK_A;	
TXCRCINTCLK_B;	
TXCRCPD_A;	
TXCRCPD_B;	
TXCRCRESET_A;	
TXCRCRESET_B;	
TXENC64B66BUSE_A;	
TXENC64B66BUSE_B;	
TXENC8B10BUSE_A;	
TXENC8B10BUSE_B;	
TXENOOB_A;	
TXENOOB_B;	
TXGEARBOX64B66BUSE_A;	
TXGEARBOX64B66BUSE_B;	
TXINHIBIT_A;	
TXINHIBIT_B;	

Inputs	Outputs
TXPMARESET_A;	
TXPMARESET_B;	
TXPOLARITY_A;	
TXPOLARITY_B;	
TXRESET_A;	
TXRESET_B;	
TXSCRAM64B66BUSE_A;	
TXSCRAM64B66BUSE_B;	
TXSYNC_A;	
TXSYNC_B;	
TXUSRCLK_A;	
TXUSRCLK_B;	
TXUSRCLK2_A;	
TXUSRCLK2_B;	

Usage

It is recommended that the GT11_DUAL is instantiated instead of the GT11_CUSTOM for all usages. It must be used if multiple GT11s are used in a design, or if the dynamic configuration bus is implemented. If the Architecture Wizard is not used, two GT11 primitives should be instantiated with the combusout port connected t othe compbusin of the other GT11 instance.

VHDL and Verilog Instantiation

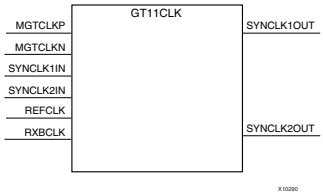
It is suggested that you use the *Architecture Wizard* ISE 8.1 in order to properly create instantiation code for the GT11_DUAL block.

For More Information

Consult the *Virtex-4 RocketIO Transceiver User Guide*.

GT11CLK

Primitive: A MUX That Can Select Fom Differential Package Input Clock, refclk From the Fabric, or rxbclk to Drive the Two Vertical Reference Clock Buses for the Column of MGTs



This block needs to be instantiated when using the dedicated package pins for RocketIO clocks. There are two available per MGT column.The attributes allow this package input to drive one or both SYNCLK clock trees. Please see the *Virtex-4 RocketIO MGT User Guide* for more details.

The attribute REFCLKSEL allows more clocking options. These options include: MGTCLK, SYNCLK1IN, SYNCLK2IN, REFCLK, RXBCLK.

Inputs and Outputs

Inputs are MGTCLKP, MGTCLKN

Outputs are SYNCLK1OUT, SYNCLK2OUT

VHDL and Verilog Instantiation

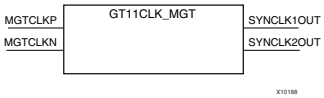
It is suggested that you use the *Architecture Wizard* in order to properly create instantiation code for the GT11CLK block.

For More Information

Consult the *Virtex-4 RocketIO Transceiver User Guide*.

GT11CLK_MGT

Primitive: Allows Differential Package Input to Drive the Two Vertical Reference Clock Buses for the Column of MGTs



This block needs to be instantiated when using the dedicated package pins for RocketIO clocks. There are two available per MGT column.The attributes allow this package input to drive one or both SYNCLK clock trees. Please see the *Virtex-4 RocketIO MGT User Guide* for more details.

GT11CLK is also available and has an attribute REFCLKSEL, with the following VALUE options: MGTCLK, SYNCLK1IN, SYNCLK2IN, REFCLK, RXBCLK.

Usage

This block allows more clocking options for MGTs.

Inputs and Outputs

Inputs are MGTCLKP, MGTCLKN

Outputs are SYNCLK1OUT, SYNCLK2OUT

VHDL and Verilog Instantiation

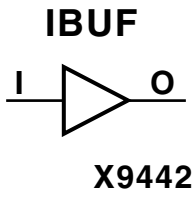
It is suggested that you use the *Architecture Wizard* in order to properly create instantiation code for the GT11CLK_MGT block.

For More Information

Consult the *Virtex-4 RocketIO Transceiver User Guide*.

IBUF

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



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 bi-directional 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
CAPACITANCE	STRING	"LOW", "NORMAL", "DON'T CARE"	"DON'T CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
IOSTANDARD	STRING	"DEFAULT	"DEFAULT"	Use to assign an I/O standard to an I/O primitive.

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 will be used
--                  : for simulation.

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

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 Version 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
```

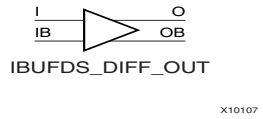
For More Information

Consult the *Virtex-4 User Guide*.

IBUFDS_DIFF_OUT

Differential I/O Input Buffer with Differential Outputs

IBUFDS_DIFF_OUT is a differential I/O input buffer with differential outputs. The differential output pair (O&OB) maintains the relation of its differential input pair.



Usage

This element is instantiated rather than inferred.

VHDL Instantiation Code

```
-- IBUFDS: Differential Input Buffer /w Differential Outputs
   --         Virtex-4
   -- Xilinx HDL Libraries Guide version 8.1i

   IBUFDS_DIFF_OUT_inst : IBUFDS_DIFF_OUT
   generic map (
      IOSTANDARD => "LVDS_25")
   port map (
      O => O,     -- Diff_p buffer output
      OB => OB,   -- Diff_n buffer output
      I => I,     -- Diff_p buffer input (connect directly to top-level port)
      IB => IB    -- Diff_n buffer input (connect directly to top-level port)
   );

   -- End of IBUFDS_DIFF_OUT _inst instantiation
```

Verilog Instantiation Code

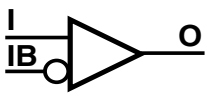
```
   // IBUFDS: Differential Input Buffer /w Differential Outputs
   //         Virtex-4
   // Xilinx HDL Libraries Guide version 8.1i
   IBUFDS_DIFF_OUT #(
      .IOSTANDARD("LVDS_25")     // Specify the input I/O standard
   ) IBUFDS_DIFF_OUT _inst (
      .O(O),   // Diff_p buffer output
      .OB(OB), // Diff_n buffer output
      .I(I),   // Diff_p buffer input (connect directly to top-level port)
      .IB(IB)  // Diff_n buffer input (connect directly to top-level port)
   );

   // End of IBUFDS_DIFF_OUT _inst instantiation
```

For More Information

Consult the *Virtex-4 Configuration Guide*.

IBUFDS

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



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). The IBUFDS component allow the specification of the particular I/O Standard to configure the IOBs as well as the specification of added delay for the incoming paths to properly align incoming data with the associated clock source. In general, an IBUFDS should be used for all differential data input or bi-directional pins.

Inputs		Outputs
I	IB	O
0	0	- *
0	1	0
1	0	1
1	1	- *

* The dash (-) means No Change.

Usage

The IBUFDS must be instantiated in order to be incorporated into the design. In order to do so, connect the input ports, I and IB, of the component directly to the associated top-level input or inout ports 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 Attributes

Attribute	Type	Allowed Values	Default	Description
CAPACITANCE	STRING	"LOW", "NORMAL", "DONT CARE"	"DONT CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
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 will be used
--             : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- IBUFDS: Differential Input Buffer
   --         Virtex-II/II-Pro, Spartan-3/3E
   -- Xilinx HDL Libraries Guide Version 8.1i

   IBUFDS_inst : IBUFDS
   generic map (
.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

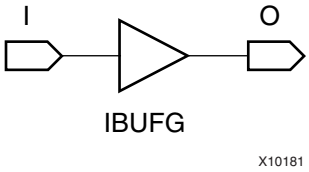
```
//   IBUFDS    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (IBUFDS_inst) and/or the port declarations within the
//    code     : 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
   //         Virtex-II/II-Pro/4, Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   IBUFDS #(
      .DIFF_TERM("FALSE"),      // Differential Termination (Virtex-4 only)
.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 *Virtex-4 User Guide*.

IBUFG

Primitive: Dedicated Input Buffer with Selectable I/O Interface



The IBUFG is an input buffer that connects to one of the dedicated clock pins of the device. Its purpose is to connect external clock source to the CLKIN or CLKFB pin of the DCM. It may also be used to connect directly to the low-skew clock routing resource in the device limiting the amount of clock delay incurred. Via attributes, the desired I/O standard for this clock pin may be specified. .

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

Usage

Synthesis tools can infer IBUFGs automatically by detecting ports that are connected to clock sources. In general, this is the preferred manner to use this buffer however if desired, it may be instantiated into the design. In order to do so, connect the input port, I, of the component directly to the associated top-level clock port and connect the output port, O, to either the DCM input pin or to all associated clocks in the design. Modify any necessary generic maps (VHDL) or named parameter value assignment (Verilog) in order to change the default behavior of the component. If a location constraint (LOC) is used for this port or buffer, ensure the location used is one of the dedicated clock pins for the device.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CAPACITANCE	STRING	"LOW", "NORMAL", "DONT CARE"	"DONT CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
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 will be used
--              : for simulation.

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

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 (
.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

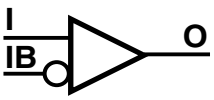
```
//    IBUFG    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (IBUFG_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

IBUFGDS

Primitive: Differential Signaling Dedicated Input Clock Buffer with Selectable I/O Interface and Optional Delay



X9255

The IBUFGDS is an input buffer that connects to one of the dedicated differential clock pin pairs of the device. Its purpose is to connect an external input differential clock to the CLKIN or CLKFB pin of the DCM. It may also be used to connect directly to the low-skew clock routing resource in the device limiting the amount of clock delay incurred. Via attributes, the desired I/O standard for this clock pin may be specified.

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

Usage

The IBUFGDS buffer must be instantiated in order to incorporate this into a design. In order to do so, connect the input port, I, of the component directly to the associated top-level clock port and connect the output port, O, to either the DCM input pin or to all associated clocks in the design. Modify any necessary generic maps (VHDL) or named parameter value assignment (Verilog) in order to change the default behavior of the component. If a location constraint (LOC) is used for this port or buffer, ensure the location used is one of the dedicated clock pins for the device.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CAPACITANCE	STRING	"LOW", "NORMAL", "DON'T CARE"	"DON'T CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
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 will be used
--             : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- IBUFGDS: Differential Global Clock Input Buffer
   --          Virtex-II/II-Pro, Spartan-3/3E
   -- Xilinx HDL Libraries Guide Version 8.1i

   IBUFGDS_inst : IBUFGDS
   generic map (
      DIFF_TERM => "FALSE", -- Differential Termination (Virtex-4 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

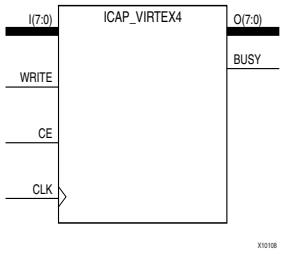
```
//   IBUFGDS   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (IBUFGDS_inst) and/or the port declarations within the
//    code     : 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)
   //          Virtex-II/II-Pro, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

ICAP_VIRTEX4

Primitive: Virtex-4 Internal Configuration Access Port



ICAP_VIRTEX4 provides user access to the Virtex-4 internal configuration access port (ICAP).

Port List and Definitions

Name	Type	Width	Function
BUSY	Output	1	Busy signal
O	Output	32	32-bit data bus output
CE	Input	1	Clock enable pin
CLK	Input	1	Clock input
WRITE	Input	1	Write signal
I	Input	32	32-bit data bus input

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ICAP_WIDTH	STRING	"X8" or "X32"	"X8"	Specifies the data width for the ICAP component.

Usage

ICAP_VIRTEX4 provides the same config user interface as the SelectIO map interface. The ICAP port can be connected to external pins or internal signals. This device is supported for schematics and instantiation only.

VHDL Instantiation Template

```
--  ICAP_VIRTEX4  : 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 (ICAP_VIRTEX4_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 will be used
--                : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- ICAP_VIRTEX4: Internal Configuration Access Port
   --               Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   ICAP_VIRTEX4_inst : ICAP_VIRTEX4
   generic map (
      ICAP_WIDTH => "X8") -- "X8" or "X32"
```

```
   port map (
      BUSY => BUSY,    -- Busy output
      O => O,          -- 32-bit data output
      CE => CE,        -- Clock enable input
      CLK => CLK,      -- Clock input
      I => I,          -- 32-bit data input
      WRITE => WRITE   -- Write input
   );

   -- End of ICAP_VIRTEX4_inst instantiation
```

Verilog Instantiation Template

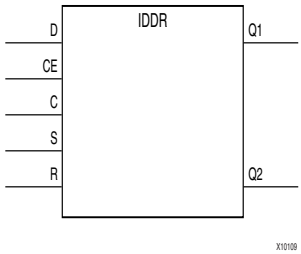
```
//  ICAP_VIRTEX4 : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (ICAP_VIRTEX4_inst) and/or the port declarations within the
//      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---->

   // ICAP_VIRTEX4: Internal Configuration Access Port
   //               Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   ICAP_VIRTEX4 #(
      .ICAP_WIDTH("X8") // "X8" or "X32"
   ) ICAP_VIRTEX4_inst (
      .BUSY(BUSY),      // Busy output
      .O(O),            // 32-bit data output
      .CE(CE),          // Clock enable input
      .CLK(CLK),        // Clock input
      .I(I),            // 32-bit data input
      .WRITE(WRITE)     // Write input
   );

   // End of ICAP_VIRTEX4_inst instantiation
```

For More Information

Consult the *Virtex-4 Configuration Guide*.

IDDR

Primitive: A Dedicated Input Register to Receive External Dual Data Rate (DDR) Signals into Virtex-4 FPGAs



The IDDR primitive is a dedicated input registers to receive external dual data rate (DDR) signals into Virtex-4 FPGAs. Unlike previous generations of Xilinx FPGAs, IDDR primitive is not limited to recovering the data for the FPGA fabric for processing at opposite edges. IDDR is available with modes that present the data to the FPGA fabric at the same clock edge. This feature allows designers to avoid additional timing complexities and CLB usage. In addition, IDDR will work in conjunction with SelectIO features of Virtex-4 architecture.

IDDR Ports

Q1 – Q2 – Data Output	These pins are the IDDR output that connects to the FPGA fabric. Q1 is the first data pair while, Q2 is the second data pair.
C – Clock Input Port	The C pin represents the clock input pin.
CE – Clock Enable Port	When asserted LOW, this port disables the output clock at port O.
D – Data Input (DDR)	This pin is where the DDR data is presented into the IDDR module. This pin connects to the IOB pad.
R - Reset	Depends on how SRTYPE is set.
S - Set	Asynchronous set pin. Set is assert HIGH.

Port List and Definitions

Name	Type	Width	Function
Q1 – Q2	Output	1 (each)	Data Output
C	Input	1	Clock input
CE	Input	1	Clock enable input
D	Input	1	Data Input (DDR)
R	Input	1	Reset
S	Input	1	Set

IDDR Modes

The following section describes the functionality of various modes of IDDR. These modes are set by the DDR_CLK_EDGE attribute.

OPPOSITE_EDGE

In the OPPOSITE_EDGE mode, data is recovered in the classic DDR methodology. Given a DDR data and clock at pin D and C respectively, Q1 will change after every positive edge of clock C, and Q2 will change after every negative edge of clock C.

SAME_EDGE

In the SAME_EDGE mode, data is still recovered by opposite edges of clock C. However, an extra register has been placed in front of the negative edge data register. This extra register is clocked with positive clock edge of clock signal C. As a result DDR data is now presented into the FPGA fabric at the same clock edge. However, because of this feature the data pair appears to be "separated." Q1 and Q2 no longer

have pair 1 and 2. Instead, the first pair presented is pair 1 and don't care, followed by pair 2 and 3 at the next clock cycle.

SAME_EDGE_PIPELINED

The SAME_EDGE_PIPELINED mode recovers data in a similar fashion as the SAME_EDGE mode. In order to avoid the "separated" effect of the SAME_EDGE mode, an extra register has been placed in front of the positive edge data register. A data pair will now appear at the Q1 and Q2 pin at the same time. However, using this mode, cost the user an additional cycle of latency for Q1 and Q2 signals to change.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DDR_CLK_EDGE	STRING	"OPPOSITE_EDGE", "SAME_EDGE", "SAME_EDGE_PIPE-LINED"	"OPPOSITE_EDGE"	DDR clock mode recovery mode selection
INIT_Q1	INTEGER	0 or 1	1	Q1 initialization value
INIT_Q2	INTEGER	0 or 1	1	Q2 initialization value
SRTYPE	STRING	"SYNC" or "ASYNC"	"SYNC"	Set/Reset type selection

Usage

This device is supported for inference and instantiation.

VHDL Template

```
--      IDDR     : 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 (IDDR_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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- IDDR: Double Data Rate Input Register with Set, Reset
   --       and Clock Enable. Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   IDDR_inst : IDDR
   generic map (
      DDR_CLK_EDGE => "OPPOSITE_EDGE", -- "OPPOSITE_EDGE", "SAME_EDGE"
                                       -- or "SAME_EDGE_PIPELINED"
      INIT_Q1 => '0', -- Initial value of Q1: '0' or '1'
      INIT_Q2 => '0', -- Initial value of Q2: '0' or '1'
      SRTYPE => "SYNC") -- Set/Reset type: "SYNC" or "ASYNC"
   port map (
      Q1 => Q1, -- 1-bit output for positive edge of clock
      Q2 => Q2, -- 1-bit output for negative edge of clock
      C => C,   -- 1-bit clock input
      CE => CE, -- 1-bit clock enable input
      D => D,   -- 1-bit DDR data input
      R => R,   -- 1-bit reset
      S => S    -- 1-bit set
      );

   -- End of IDDR_inst instantiation
```

Verilog Template

```
//      IDDR     : 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 : (IDDR_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---->

   // IDDR: Input Double Data Rate Input Register with Set, Reset
   //       and Clock Enable. Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   IDDR #(
      .DDR_CLK_EDGE("OPPOSITE_EDGE"), // "OPPOSITE_EDGE", "SAME_EDGE"
                                      //    or "SAME_EDGE_PIPELINED"
      .INIT_Q1(1'b0), // Initial value of Q1: 1'b0 or 1'b1
      .INIT_Q2(1'b0), // Initial value of Q2: 1'b0 or 1'b1
      .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
   ) IDDR_inst (
      .Q1(Q1), // 1-bit output for positive edge of clock
      .Q2(Q2), // 1-bit output for negative edge of clock
      .C(C),   // 1-bit clock input
      .CE(CE), // 1-bit clock enable input
      .D(D),   // 1-bit DDR data input
      .R(R),   // 1-bit reset
```

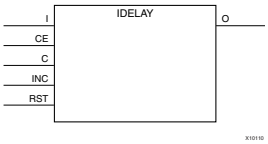
```
   .S(S)    // 1-bit set
);

// End of IDDR_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

IDELAY

Primitive: Startup calibration module for IDELAY elements



Virtex-4 modules have an IDELAY module in the input path of every user I/O. IDELAY allows the implementation of deskew algorithms to correctly capture incoming data. IDELAY can be applied to data signals, clock signals, or both. IDELAY features a fully-controllable, 64-tap delay line. Each tap delay is carefully calibrated to provide an absolute delay value of 78 ps independent of process, voltage, and temperature variations. Three modes of operation are available:

- Zero hold time delay mode

  This mode of operation allows backward compatibility for designs using the zero-hold time delay feature in Virtex-II and Virtex-II Pro devices. When used in this mode, the IDELAYCTRL primitive does not need to be instantiated.

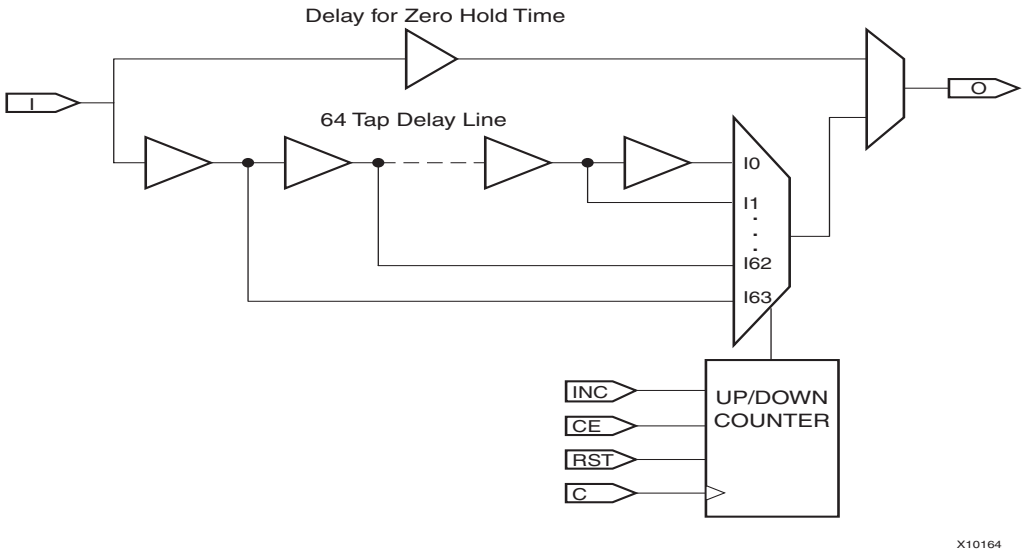
- Fixed tap-delay mode

  In the fixed tap-delay mode, the delay value is set to the number determined by the attribute IOBDELAY_VALUE. This value cannot be changed during run-time. When used in this mode, the IDELAYCTRL primitive must be instantiated.

- Variable tap-delay mode

  In the variable tap-delay mode, the delay value can be changed at run-time by manipulating the control signals CE and INC. When used in this mode, the IDELAYCTRL primitive must be instantiated.

The following figure shows the block diagram of the IDELAY module.



IDELAY Module Block Diagram

The following figure shows the IDELAY primitive.

IDELAY Primitive

The following table lists the availale ports in the IDELAY primitive.

AvailablePorts	Direction	Size	Function
I	Input	1	Serial input data from IOB
C	Input	1	Clock input
INC	Input	1	Increment/decrement number of tap delays
CE	Input	1	Enable increment/decrement function
RST	Input	1	Reset delay chain to pre-programmed value. If no value programmed, reset to 0.
O	Output	1	Combinatorial output

IDELAY Ports

Data Input and Output - I and O

IDELAY primitives are located in general purpose IOB locations. The input and output connectivity differs for each type of IOB location.

General Purpose IOBs

The input of IDELAY in a general-purpose IOB comes directly from the input buffer, IBUF. The output of IDELAY (O) is connected directly to the user logic. The input and output data path is combinatorial and is not affected by the clock signal (C). However, the user can choose to register the output signal (O) in the IOB.

Regional Clock-Capable IOBs

Regional clock-capable IOBs are located in one I/O pair directly above and below an HCLK IOB. The input of IDELAY in a regional clock-capable IOB comes directly from the input buffer, IBUF. The output of IDELAY in a regional clock-capable IOB can go to one of the following locations:

1. Directly to the user logic
2. BUFIO (in the case of a regional clock signal)

The regional clock buffer, BUFIO, connects the incoming regional clock signal to the regional I/O clock tree, IOCLK. BUFIO also connects to the regional clock buffer, BUFR to connect to the regional clock tree, rclk. The input and output data path is combinatorial and is not affected by the clock signal (C). However, the user can choose to register the output signal (O) in the IOB.

Global Clock-Capable IOBs

The global clock-capable IOBs are located in the center I/O column. The input of the IDELAY module in a global clock-capable IOB comes directly from the input global clock buffer, IBUFG. The output of the IDELAY module in a global clock-capable IOB can go to one of the following locations:

1. Directly to the user logic
2. BUFG (in the case of a global clock signal)

The global clock buffer, BUFG, connects the incoming regional clock signal to the global clock tree, gclk. The input and output data path is combinatorial and is not affected by the clock signal (C). However, the user can choose to register the output signal (O) in the IOB.

Clock Input - C

All control inputs to IDELAY (RST, CE and INC) are synchronous to the clock input (C). The data input and output (I and O) of IDELAY is not affected by this clock signal. This clock input is identical to the CLKDIV input for the ISERDES. All the clock sources used to drive CLKDIV can therefore drive the IDELAY clock input (C). The clock sources that can drive the clock input (C) are:

- Eight gclk (global clock tree)
- Two rclk (regional clock tree)

Module Reset - RST

The IDELAY reset signal, RST, resets the tap-delay line to a value set by the IOBDELAY_VALUE attribute. If the IOBDELAY_VALUE attribute is not specified, the tap-delay line is reset to 0.

Increment/Decrement Signals - CE, INC

The increment/decrement enable signal (CE) determines when the increment/decrement signal (INC) is activated. INC determines whether to increment or decrement the tap-delay line. When CE = 0, the tap delay remains constant no matter what the value of INC. When CE = 1, the tap-delay value increments or decrements depending on the value of INC. The tap delay is incremented or decremented synchronously with respect to the input clock (C). As long as CE = 1, the tap-delay increments or decrements by one every clock cycle. The increment/decrement operation is summarized in the following table:

Operation	RST	CE	INC
Reset to configured value of tap count	1	x	x
Increment tap count	0	1	1
Decrement tap count	0	1	0
No change	0	0	x

Note:

1. RST resets delay chain to tap count specified by attribute IOBDELAY_VALUE. If IOBDLEAY_VALUE not specified, tap count reset to 0.
2. RST, CE, and INC are synchronous to the input clock signal (C).

When CE is raised, the increment/decrement operation begins on the next positive clock cycle. When CE is lowered, the increment/decrement operation ceases on the next positive clock cycle.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IOBDELAY_TYPE	String	"DEFAULT", "FIXED", or "VARIABLE"	"DEFAULT"	This attribute sets the type of tap delay.
IOBDELAY_VALUE	Integer	0 to 63	0	This attribute specifies the initial number of tap delays.

IOBDELAY_TYPE Attribute

The IOBDELAY_TYPE attribute sets the type of delay used. The attribute values are DEFAULT, FIXED, and VARIABLE. The default value is DEFAULT. When set to DEFAULT, the zero-hold time delay element is selected. This delay element eliminates pad-to-pad hold time. The delay is matched to the internal clock-distribution delay of the Virtex-4 device. When used, it guarantees a pad-to-pad hold time of zero.

When set to FIXED, the tap-delay value is fixed at the number of taps determined by the IOBDELAY_VALUE attribute. This value is preset and cannot be changed dynamically.

When set to VARIABLE, the variable tap delay is selected. The tap delay can be incremented by setting CE = 1and INC = 1 or decremented by setting CE = 1 and INC = 0. The increment/decrement operation is synchronous to C, the input clock signal.

IOBDELAY_VALUE Attribute

The IOBDELAY_VALUE attribute specifies the initial number of tap delays. The possible values are any integers from 0 to 63. The default value is 0. When set to 0, the total delay becomes the delay of the output MUX which is approximately 400 ps.

The value of the tap delay reverts to IOBDELAY_VALUE when the tap delay is reset (RST = 1), or the IOBDELAY_TYPE is set to FIXED.

VHDL Instantiation Template

```
--     IDELAY    : 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 (IDELAY_inst) and/or the port declarations
--      code     : after the "=>" assignment maybe changed to properly
--               : connect this function to the design. Unused inputs
--               : or outputs may 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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- IDELAY: Input Delay Element
   --         Virtex-4
```

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

   IDELAY_inst : IDELAY
   generic map (
      IOBDELAY_TYPE => "DEFAULT", -- "DEFAULT", "FIXED" or "VARIABLE"
      IOBDELAY_VALUE => 0) -- Any value from 0 to 63
   port map (
      O => O,      -- 1-bit output
      C => C,      -- 1-bit clock input
      CE => CE,    -- 1-bit clock enable input
      I => I,      -- 1-bit data input
      INC => INC,  -- 1-bit increment input
      RST => RST   -- 1-bit reset input
   );

   -- End of IDELAY_inst instantiation
```

Verilog Instantiation Template

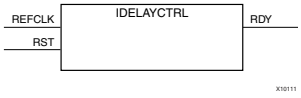
```
//     IDELAY    : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (IDELAY_inst) and/or the port declarations within the
//      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---->

   // IDELAY: Input Delay Element
   //         Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   IDELAY #(
      .IOBDELAY_TYPE("DEFAULT"), // "DEFAULT", "FIXED" or "VARIABLE"
      .IOBDELAY_VALUE(0)         // Any value from 0 to 63
   ) IDELAY_inst (
      .O(O),      // 1-bit output
      .C(C),      // 1-bit clock input
      .CE(CE),    // 1-bit clock enable input
      .I(I),      // 1-bit data input
      .INC(INC), // 1-bit increment input
      .RST(RST)  // 1-bit reset input
   );

   // End of IDELAY_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

IDELAYCTRL

Primitive: IDELAY tap delay value control



The IDELAYCTRL module must be instantiated when using the tap-delay line. This occurs when the IDELAY or ISERDES primitive is instantiated with the IOBDELAY_TYPE attribute set to Fixed or Variable. The IDELAYCTRL module provides a voltage bias, independent of process, voltage, and temperature variations to the tap-delay line using a fixed-frequency reference clock, REFCLK. This enables very accurate delay tuning.

Usage

The most efficient way to use the IDELAYCTRL module is to define and lock down the placement of every IDELAYCTRL instance used in a design. This is done by instantiating the IDELAYCTRL instances with location (LOC) constraints. Instantiating IDELAYCTRL instances without LOC constraints cause the implementation tools to replicate IDELAYCTRL instances throughout the device, even in HCLK regions not using the tap-delay line. This increases the power consumption, uses more global clock resources in every HCLK region, and increases the use of routing resources.

When instantiating IDELAYCTRL instances with defined LOC constraints, you must define and lock placement of all ISERDES and IDELAY components using the tap-delay line (IOBDELAY_TYPE attribute set to Fixed or Variable).

VHDL Instantiation Template

```
--   IDELAYCTRL  : 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 (IDELAYCTRL_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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- IDELAYCTRL : Input Delay Element Control
   --          Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   IDELAYCTRL_inst : IDELAYCTRL
   port map (
      RDY => RDY,       -- 1-bit output indicates validity of the REFCLK
      REFCLK => REFCLK, -- 1-bit reference clock input
      RST => RST        -- 1-bit reset input
   );

   -- End of IDELAYCTRL_inst instantiation
```

Verilog Instantiation Template

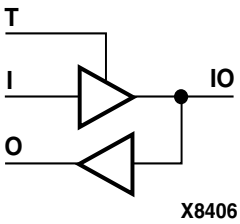
```
//    IDELAYCTRL : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (IDELAYCTRL_inst) and/or the port declarations within the
//      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---->

   // IDELAYCTRL: Input Delay Control Element (Must be used in conjunction with the IDELAY
   //             when used in FIXED or VARIABLE tap-delay mode)
   //             Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   IDELAYCTRL IDELAYCTRL_inst (
      .RDY(RDY),        // 1-bit ready output
      .REFCLK(REFCLK), // 1-bit reference clock input
      .RST(RST)         // 1-bit reset input
   );

   // End of IDELAYCTRL_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

IOBUF

Primitive: Bi-Directional Buffer with Selectable I/O Interface



Virtex-4 IOBUFs are bi-directional buffer. The I/O interface corresponds to a specific I/O standard. You can attach an IOSTANDARD attribute to an IOBUF instance.

IOBUF components that use the LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, and LVCMOS33 signaling standards have selectable drive and slew rates using the DRIVE and FAST or SLOW constraints. The defaults are DRIVE = 12 mA and SLOW slew.

IOBUFs are composites of IBUF and OBUFT elements. The O output is X (unknown) when IO (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 rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CAPACITANCE	STRING	"LOW", "NORMAL", "DONT CARE"	"DONT CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
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	INTEGER	"SLOW" or "FAST"	"SLOW"	Sets the output rise and fall time.

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 will be used
--             : for simulation.

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

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

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

   -- IOBUF: Single-ended Bi-directional Buffer
   --        All devices
   -- Xilinx  HDL Libraries Guide Version 8.1i

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

   -- End of IOBUF_inst instantiation
```

Verilog Instantiation Template

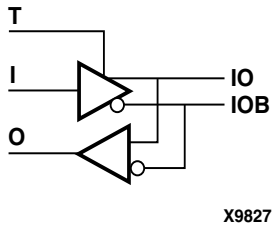
```
//      IOBUF      : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (IOBUF_inst) and/or the port declarations within the
//      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 Bi-directional Buffer
   //        All devices
   // Xilinx HDL Libraries Guide Version 8.1i

   IOBUF #(
      .DRIVE(12), // Specify the output drive strength
.IOSTANDARD("DEFAULT"), // Specify the I/O standard
      .SLEW("SLOW") // Specify the output slew rate
   ) IOBUF_inst (
      .O(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 *Virtex-4 User Guide*.

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	No Change
0	0	0	1	0
1	0	1	0	1

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 will be used
--             : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- IOBUFDS: Differential Bi-directional Buffer
   --          Virtex-II/II-Pro, Spartan-3/3E
   -- Xilinx  HDL Libraries Guide Version 8.1i

   IOBUFDS_inst : IOBUFDS
   generic map (
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

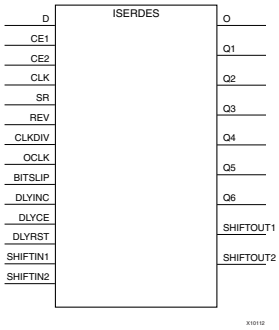
```
//     IOBUFDS   : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (IOBUFDS_inst) and/or the port declarations within the
//      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 Bi-directional Buffer
   //          Virtex-II/II-Pro/4, Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   IOBUFDS #(
.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 *Virtex-4 User Guide*.

ISERDES

Primitive: Dedicated I/O Buffer Input Deserializer



The Virtex-4 architecture provides a way for the user to easily implement source synchronous solutions by using the ISERDES module. Unlike previous generations of FPGAs, ISERDES is a dedicated source synchronous I/O architecture. This module helps user by saving logic resources in the FPGA fabric for source synchronous applications. Furthermore, ISERDES also avoids additional timing complexities that may be encountered when designing such solution in the FPGA fabric. ISERDES module is present in all Virtex-4 family of FPGA.

The ISERDES module contains or works in conjunction with the following modules: serial to parallel converters, serial delay chains, a word alignment unit (BITSLIP), and a clock enable (CE) module. In addition, ISERDES contains multiple clock inputs to accommodate various applications and will work in conjunction with the SelectIO features in Virtex-4 family.

ISERDES Ports (Detailed Description)

O – Combinatorial Output

This port is an unregistered output of the ISERDES module. It is the unregistered output of the delay chain. In addition, this output port can also be configured to bypass all the submodules within ISERDES module. This output can be used to drive the BUFIOs.

Q1 to Q6 – Registered Outputs

This port is a registered output of the ISERDES module. Using these outputs, the user has a selection of the following combination of ISERDES submodules path as the inputs:

1. Delay chain to serial to parallel converter to bitslip module.

2. Delay chain to serial to parallel converter.

These ports can be programmed from 2 to 6 bits. In the extended width mode, this port can be expanded up to 10 bits.

SHIFTOUT 1-2 – Data input expansion (master)

Carry out for data input expansion. Connect to SHIFTIN1/2 of slave.

BITSLIP – BITSLIP Control Pin

This pin allows the ISERDES to perform a BITSLIP operation when logic HIGH is given and the BITSLIP module is enabled.

CE 1-2 – Clock Enables

Clock Enables input that feeds into the CE module.

CLK – High Speed Forwarded Clock Input

This clock input is used to drive the Serial to Parallel Converter and the BITSLIP module. The possible source for the CLK port is from one of the following clock resources:

1. Eight global clock lines in a clock region

2. Two regional clock lines

3. Six clock capable I/Os (within adjacent clock region)

4. Fabric (through bypass)

CLKDIV – Divided High Speed Forwarded Clock Input

This clock input is used to drive the Serial to Parallel Converter, Delay Chain, the BITSLIP module, and CE module. This clock has to have slower frequency than the clock connected to the CLK port. The possible source for the CLKDIV port is from one of the following clock resources:

1. Eight global clock lines in a clock region

2. Two regional clock lines

D – Serial Input Data from IOB

The D is where all the incoming data enters the ISERDES module. This port works in conjunction with SelectIO features for Virtex-4 architecture to accommodate the desired I/O standards.

DLYCE – Delay Chain Enable Pin

This pin allows the user to increment/decrement the delay chain tap value by setting it to logic HIGH.

DLYINC – Delay Chain Increment/Decrement Pin

When the DLYCE pin is asserted HIGH, the value at DLYINC pin increments/decrements the delay chain value. Logic HIGH increments the tap value, while logic LOW decrements the tap value.

DLYRST – Delay Chain Reset Pin

Asserting this pin to logic HIGH sets the delay chain value to the IOBDELAY_VALUE.

OCLK – High Speed Clock for Memory Interfaces Applications

This clock input is used to drive the serial to parallel converter in the ISERDES module. The possible source for the OCLK port is from one of the following clock resources:

1. Eight global clock lines in a clock region

2. Two regional clock lines

3. Six clock capable I/Os (within adjacent clock region)

4. Fabric (through bypass)

This clock is an ideal solution for memory interfaces in which strobe signals are required.

OFB – OQ Internal Feedback

Internal feedback loop from OQ the output of OSERDES. For internal testing purposes.

REV - Reverse SR pin

When SR is used, a second input, REV forces the storage element into the opposite state. The reset condition predominates over the set condition.

SR - Set/Reset Input

The set/reset pin, SR forces the storage element into the state specified by the SRVAL attribute, set through the user constraints file (UCF). SRVAL = “1” forces a logic 1. SRVAL = “0” forces a logic "0." When SR is used, a second input (REV) forces the storage element into the opposite state. The reset condition predominates over the set condition. The following truth tables describes the operation of SR in conjunction with REV.

Truth Table When SRVAL = “0” (Default Condition)

SR	REV	Function
0	0	NOP
0	1	Set
1	0	Reset
1	1	Reset

Truth Table When SRVAL = “1”

SR	REV	Function
0	0	NOP
0	1	Reset
1	0	Set
1	1	Reset

SHIFTIN 1-2 – Data input expansion (slave)

Carry input for data input expansion. Connect to SHIFTOUT1/2 of master.

TFB – TQ Internal Feedback

Internal feedback loop from TQ output of OSERDES. For internal testing purposes.

ISERDES Submodules

Delay Chains Module

The Delay Chains module is a dedicated architecture that provides an adjustable or fixed timing relationship between input data and forwarded clock. This solution is achieved by placing delays in the ISERDES module that de-skew the inputs. The input delay chains can be preprogrammed (fixed) or dynamically changed (variable). In addition this module works in conjunction with IDELAYCTRL, a primitive available in Virtex-4 devices.

A number of attributes being required in order to use the Delay Chains module. The attributes are as follow:

1. IOBDELAY_VALUE

2. IOBDELAY

3. IOBDELAY_TYPE

IOBDELAY_VALUE can take values between 0 and 63. This attribute defines the number of delay taps used. Default value for this attribute is 0.

Setting the IOBDELAY attribute to "IBUF," "IFD," and "BOTH" allows the Delay Chains to be used in the combinatorial output (O output), registered output (Q1-Q6 output), and both respectively. Setting the IOBDELAY attribute to "NONE" bypasses the delay chains module.

The IOBDELAY_TYPE can take three different values: "DEFAULT," "FIXED," or "VARIABLE." The "DEFAULT" allows the user to use the 0 hold time value. Using the "FIXED" mode, the delay taps equal to value defined by IOBDELAY_VALUE. In this mode, the value can't be changed after the device is programmed. In the last mode, "VARIABLE," the delay value is set to an initial value defined by IOBDELAY_VALUE and adjustable after the device is programmed.

The Delay Chains module is controlled by DLYRST, DLYCE, and DLYINC pins. Each of the operations performed with these pins are synchronous to the CLKDIV clock signal. Asserting DLYRST to logic HIGH configures the delay tap to the value defined in IOBDELAY_VALUE. To increment/decrement the delay tap value, the user will need to use both DLYCE and DLYINC. For this operation to proceed, the DLYCE must be asserted to logic HIGH. Setting DLYINC to 1 will increment and setting DLYINC to 0 will decrement the delay tap value.

The following table identifies the Delay Chains Controls:

Operation	DLYRST	DLYCE	DLYINC
Reset to IOBDELAY_VALUE	1	X	X
Increment tap value	0	1	1
Decrement tap value	0	1	0
No change	0	0	X

Note: All Delay Chains operations are synchronous to CLKDIV.

Serial to Parallel Converter

The serial to parallel converter in the ISERDES module takes in serial data and convert them into data width choices from 2 to 6. Data widths larger than 6 (7,8, and 10) is achievable by cascading two ISERDES modules for data width expansion. In order to do this, one ISERDES must be set into a MASTER mode, while another is set into SLAVE mode. The user will also need to connect the SHIFTIN of "slave" and SHIFTOUT of "master" ports together. The "slave" will only use Q3 to Q6 ports as its output. The serial to parallel converter is available for both SDR and DDR modes.

This module is primarily controlled by CLK and CLKDIV clocks. The following table describes the relationship between CLK and CLKDIV for both SDR and DDR mode.

The following table illustrates the CLK/CLKDIV relationship of the serial to parallel converter.

SDR Data Width	DDR Data Width	CLK	CLKDIV
2	4	2X	X
3	6	3X	X
4	8	4X	X
5	10	5X	X
6	-	6X	X
7	-	7X	X
8	-	8X	X

CE Module

CE Module is essentially a 2:1 parallel to serial converter. This module is controlled by CLKDIV clock input and is used to control the clock enable port of the Serial to Parallel Converter module.

BITSLIP Module

The BITSLIP module is a "Barrel Shifter" type function that reorders an output sequence. An output pattern only changes whenever the BITSLIP is invoked. The maximum number of BITSLIP reordering is always equal to the number of bits in the pattern length minus one (DATA_WIDTH – 1). BITSLIP is supported for both SDR and DDR operations. However, note that the output reordering for SDR and DDR greatly differs.

In order to use the BITSLIP, the attribute "BITSLIP_ENABLE" must be set to "ON." Setting this attribute to "OFF" allows the user to bypass the BITSLIP module.

The BITSLIP operation is synchronous to the CLKDIV clock input. In order to invoke the BITSLIP module, the BITSLIP port must be asserted HIGH for one and only one CLKDIV cycle. After one CLKDIV cycle the BITSLIP port is asserted HIGH, the BITSLIP operation is completed. For DDR mode, a BITSLIP operation may not be stable until after two CLKDIV cycles. All outputs of the BITSLIP appear in one of the registered output ports (Q1 to Q6) BITSLIP operations are synchronous to CLKDIV.

Additional Features

Width Expansion

It is possible to use the ISERDES modules to recover data widths larger than 6. In order to use this feature, two ISERDES modules need to be instantiated. Both the ISERDES must be an adjacent master and slave pair. The attribute SERDES_MODE must be set to either "MASTER" or "SLAVE" in order to differentiate the modes of the ISERDES pair. In addition, the user must connect the SHIFOUT ports of the MASTER to the SHIFTIN ports of the SLAVE. This feature supports data widths of 7, 8, and 10 for SDR and DDR mode. The table below lists the data width availability for SDR and DDR mode.

Mode	Widths
SDR Data Widths	2,3,4,5,6,7,8
DDR Data Widths	4,6,8,10

Port List and Definitions

Name	Type	Width	Description
O	Output	1	Combinatorial Output
Q1 – 6	Output	1 (each)	Registered Outputs
SHIFTOUT1 – 2	Output	1 (each)	Carry out for data input expansion. Connect to SHIFTIN1/2 of slave.
BITSLIP	Input	1	Invokes the ISERDES to perform a BITSLIP operation when logic HIGH is given and the BITSLIP module is enabled.
CE1 – 2	Input	1 (each)	Clock enables inputs.
CLK	Input	1	High speed forwarded clock
CLKDIV	Input	1	Divided clock input.
D	Input	1	Serial input data from IOB
DLYCE	Input	1	Enable delay chain to be incremented or decremented
DLYINC	Input	1	If 1/0, increment/decrement delay chain 1 tap for every CLKDIV cycle.
DLYRST	Input	1	Resets delay line to programmed value of IOBDELAY_VALUE (=Tap Count). If no value programmed, resets delay line to 0 taps.
OCLK	Input	1	High-speed clock input for memory applications.

Name	Type	Width	Description
OFB	Input	1	Internal feedback loop from OQ output of OSERDES. For internal testing purposes.
REV	Input	1	Reverse SR
SR	Input	1	Set/Reset Input
SHIFTIN1 – 2	Input	1 (each)	Carry input for data input expansion. Connect to SHIFTOUT1/2 of master.
TFB	Input	1	Internal feedback loop from TQ output of OSERDES. For internal testing purposes.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BITSLIP_ENABLE	BOOLEAN	FALSE, TRUE	0	Allows the user to enable the bitslip controller.
DATA_RATE	STRING	"SDR" or "DDR"	"DDR"	Specify data rate of either allowed value.
DATA_WIDTH	STRING	If DATA_RATE = "DDR", value is limited to 4,6,8, or 10. If DATA_RATE = "SDR", value is limited to 2,3,4,5,6,7, or 8.	4	Defines the serial to parallel converter width. This value also depends on the SDR vs. DDR and the Mode of the ISERDES
INIT_Q1	1-Bit Binary	1-Bit Binary	1'b0	Defines the intial value of Q outputs
INIT_Q2	1-Bit Binary	1-Bit Binary	1'b0	Defines the intial value of Q outputs
INIT_Q3	1-Bit Binary	1-Bit Binary	1'b0	Defines the intial value of Q outputs
INIT_Q4	1-Bit Binary	1-Bit Binary	1'b0	Defines the intial value of Q outputs
INTERFACE_TYPE	STRING	"MEMORY" or "NETWORKING"	"MEMORY"	Determines which ISERDES use model is used.
IOBDELAY	STRING	"NONE", "IBUF", "IFD", "BOTH	"NONE"	Defines where the at the ISERDES outputs the Delay Chains will be used
IOBDELAY_TYPE	STRING	"DEFAULT", "FIXED", or "VARIABLE"	"DEFAULT"	Defines whether the Delay Chains is in fixed or variable mode
IOBDELAY_VALUE	INTEGER	0 to 63	0	Set initial tap delay to an integer from 0 to 63.
NUM_CE	INTEGER	1 or 2	2	Define number or clock enables to an integer of 1 or 2.
SERDES_MODE	STRING	"MASTER" or "SLAVE"	"MASTER"	Defines whether the ISERDES module is a master or slave when width expansion is used
SRVAL_Q1	1-Bit Binary	1-Bit Binary	1'b0	Define Q1 output value upon SR assertion - 1'b1 or 1'b0.
SRVAL_Q1 to SRVAL_Q4	BINARY	1^b0 or 1^b1	1'b0	Defines the value of Q outputs when reset is invoked
SRVAL_Q2	1-Bit Binary	1-Bit Binary	1'b0	Define Q2 output value upon SR assertion - 1'b1 or 1'b0.
SRVAL_Q3	1-Bit Binary	1-Bit Binary	1'b0	Define Q3 output value upon SR assertion - 1'b1 or 1'b0.
SRVAL_Q4	1-Bit Binary	1-Bit Binary	1'b0	Define Q4 output value upon SR assertion - 1'b1 or 1'b0.

VHDL Template

```
--     ISERDES    : 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 (ISERDES_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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- ISERDES: Input SERDES
   --         Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   ISERDES_inst : ISERDES
   generic map (
      BITSLIP_ENABLE => FALSE, -- TRUE/FALSE to enable bitslip controller
      DATA_RATE => "DDR", -- Specify data rate of "DDR" or "SDR"
      DATA_WIDTH => 4, -- Specify data width - For DDR 4,6,8, or 10
                       -- For SDR 2,3,4,5,6,7, or 8
      INIT_Q1 => '0',  -- INIT for Q1 register - '1' or '0'
      INIT_Q2 => '0',  -- INIT for Q2 register - '1' or '0'
      INIT_Q3 => '0',  -- INIT for Q3 register - '1' or '0'
      INIT_Q4 => '0',  -- INIT for Q4 register - '1' or '0'
      INTERFACE_TYPE => "MEMORY", -- Use model - "MEMORY" or "NETWORKING"
      IOBDELAY => "NONE", -- Specify outputs where delay chain will be applied
                          -- "NONE", "IBUF", "IFD", or "BOTH"
      IOBDELAY_TYPE => "DEFAULT", -- Set tap delay "DEFAULT", "FIXED", or "VARIABLE"
      IOBDELAY_VALUE => 0, -- Set initial tap delay to an integer from 0 to 63
      NUM_CE => 2, -- Define number or clock enables to an integer of 1 or 2
      SERDES_MODE => "MASTER", --Set SERDES mode to "MASTER" or "SLAVE"
      SRVAL_Q1 => '0', -- Define Q1 output value upon SR assertion - '1' or '0'
      SRVAL_Q2 => '0', -- Define Q1 output value upon SR assertion - '1' or '0'
      SRVAL_Q3 => '0', -- Define Q1 output value upon SR assertion - '1' or '0'
      SRVAL_Q4 => '0') -- Define Q1 output value upon SR assertion - '1' or '0'
   port map (
      O => O,     -- 1-bit output
      Q1 => Q1,   -- 1-bit output
      Q2 => Q2,   -- 1-bit output
      Q3 => Q3,   -- 1-bit output
      Q4 => Q4,   -- 1-bit output
      Q5 => Q5,   -- 1-bit output
      Q6 => Q6,   -- 1-bit output
      SHIFTOUT1 => SHIFTOUT1, -- 1-bit output
      SHIFTOUT2 => SHIFTOUT2, -- 1-bit output
      BITSLIP => BITSLIP,     -- 1-bit input
      CE1 => CE1,         -- 1-bit input
      CE2 => CE2,         -- 1-bit input
      CLK => CLK,         -- 1-bit input
      CLKDIV => CLKDIV,   -- 1-bit input
      D => D,             -- 1-bit input
      DLYCE => DLYCE,     -- 1-bit input
      DLYINC => DLYINC,   -- 1-bit input
      DLYRST => DLYRST,   -- 1-bit input
      OCLK => OCLK,       -- 1-bit input
      REV => REV,         -- 1-bit input
      SHIFTIN1 => SHIFTIN1, -- 1-bit input
      SHIFTIN2 => SHIFTIN2, -- 1-bit input
      SR => SR            -- 1-bit input
   );

   -- End of ISERDES_inst instantiation
```

Verilog Template

```
//     ISERDES   : 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 : (ISERDES_inst) and/or the port declarations within the
//      code     : parenthesis maybe changed to properly reference and
```

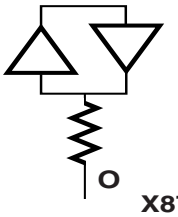
```
//               : connect this function to the design.  Delete or comment
//               : out inputs/outs that are not necessary.

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

   // ISERDES: Source Synchronous Input Deserializer
   //          Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   ISERDES #(
      .BITSLIP_ENABLE("FALSE"), // TRUE/FALSE to enable bitslip controller
      .DATA_RATE("DDR"), // Specify data rate of "DDR" or "SDR"
      .DATA_WIDTH(4), // Specify data width - For DDR 4,6,8, or 10
                      //    For SDR 2,3,4,5,6,7, or 8
      .INIT_Q1(1'b0), // INIT for Q1 register - 1'b1 or 1'b0
      .INIT_Q2(1'b0), // INIT for Q2 register - 1'b1 or 1'b0
      .INIT_Q3(1'b0), // INIT for Q3 register - 1'b1 or 1'b0
      .INIT_Q4(1'b0), // INIT for Q4 register - 1'b1 or 1'b0
      .INTERFACE_TYPE("MEMORY"), // Use model - "MEMORY" or "NETWORKING"
      .IOBDELAY("NONE"), // Specify outputs where delay chain will be applied
                         //    "NONE", "IBUF", "IFD", or "BOTH"
      .IOBDELAY_TYPE("DEFAULT"), // Set tap delay "DEFAULT", "FIXED", or "VARIABLE"
      .IOBDELAY_VALUE(0), // Set initial tap delay to an integer from 0 to 63
      .NUM_CE(2), // Define number or clock enables to an integer of 1 or 2
      .SERDES_MODE("MASTER"), // Set SERDES mode to "MASTER" or "SLAVE"
      .SRVAL_Q1(1'b0), // Define Q1 output value upon SR assertion - 1'b1 or 1'b0
      .SRVAL_Q2(1'b0), // Define Q2 output value upon SR assertion - 1'b1 or 1'b0
      .SRVAL_Q3(1'b0), // Define Q3 output value upon SR assertion - 1'b1 or 1'b0
      .SRVAL_Q4(1'b0) // Define Q4 output value upon SR assertion - 1'b1 or 1'b0
   ) ISERDES_inst (
      .O(O),     // 1-bit combinatorial output
      .Q1(Q1),   // 1-bit registered output
      .Q2(Q2),   // 1-bit registered output
      .Q3(Q3),   // 1-bit registered output
      .Q4(Q4),   // 1-bit registered output
      .Q5(Q5),   // 1-bit registered output
      .Q6(Q6),   // 1-bit registered output
      .SHIFTOUT1(SHIFTOUT1), // 1-bit carry output
      .SHIFTOUT2(SHIFTOUT2), // 1-bit carry output
      .BITSLIP(BITSLIP),     // 1-bit Bitslip input
      .CE1(CE1),         // 1-bit clock enable input
      .CE2(CE2),         // 1-bit clock enable input
      .CLK(CLK),         // 1-bit clock input
      .CLKDIV(CLKDIV),   // 1-bit divided clock input
      .D(D),             // 1-bit serial data input
      .DLYCE(DLYCE),     // 1-bit delay chain enable input
      .DLYINC(DLYINC),   // 1-bit delay increment/decrement input
      .DLYRST(DLYRST),   // 1-bit delay chain reset input
      .OCLK(OCLK),       // 1-bit high-speed clock input
      .REV(REV),         // 1-bit reverse SR input
      .SHIFTIN1(SHIFTIN1), // 1-bit carry input
      .SHIFTIN2(SHIFTIN2), // 1-bit carry input
      .SR(SR)            // 1-bit set/reset input
   );

   // End of ISERDES_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

KEEPER

Primitive: KEEPER Symbol



X8718

KEEPER is a weak keeper element that retains the value of the net connected to its bidirectional O pin. For example, if a logic 1 is being driven onto the net, 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 will be used
--             : for simulation.

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

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, CoolRunner-II
   -- 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

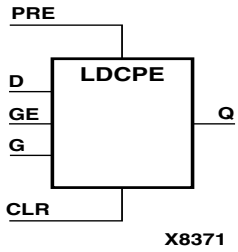
```
//     KEEPER    : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (KEEPER_inst) and/or the port declarations within the
//      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, CoolRunner-II
   // 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 *Virtex-4 User Guide*.

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.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the Virtex-4 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 is inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes must 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
      PRE => PRE  -- Asynchronous preset/set input
      );

   -- End of LDCPE_inst instantiation
```

Verilog Instantiation Template

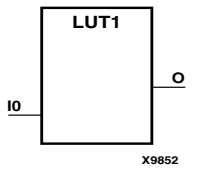
```
//     LDCPE    : In order to incorporate this function into the design,
//    Verilog   : the following instance declaration needs to be placed
//   instance   : in the body of the design code.  The instance name
//  declaration : (LDCPE_inst) and/or the port declarations within the
//     code     : 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
```

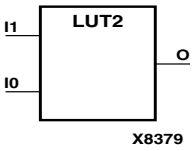
For More Information

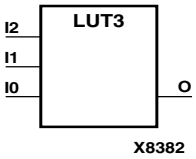
Consult the *Virtex-4 User Guide*.

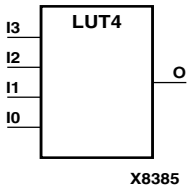
LUT1, 2, 3, 4

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 Virtex-4 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 generally 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 will be used
--             : for simulation.

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

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 Code 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
```

```
LUT1_D
```

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 will be used
--             : for simulation.

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

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 Code 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 will be used
--             : for simulation.

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

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 Code 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 will be used
--             : for simulation.
```

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

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 Code for LUT4

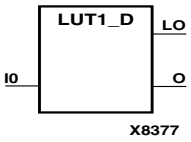
```
//     LUT4    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT4_inst) and/or the port declarations within the
//    code     : 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
```

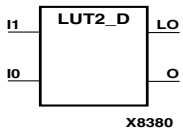
For More Information

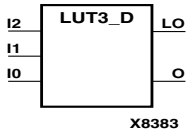
Consult the *Virtex-4 User Guide*.

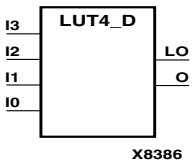
LUT1_D, LUT2_D, LUT3_D, LUT4_D

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 used to connect to another output within the same CLB slice and to the fast connect buffer.

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

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 generally 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 will be used
--             : for simulation.

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

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 Code 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 will be used
--              : for simulation.

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

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 Code 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 will be used
--             : for simulation.

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

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
```

Verilog Instantiation Code 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 will be used
```

```
--             : for simulation.

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

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 Code for LUT4_D

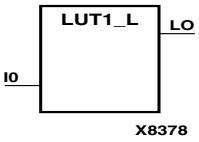
```
//   LUT4_D    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT4_D_inst) and/or the port declarations within the
//    code     : 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
```

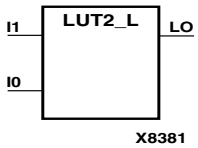
For More Information

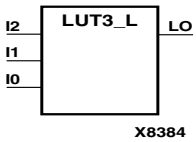
Consult the *Virtex-4 User Guide*.

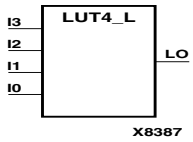
LUT1_L, LUT2_L, LUT3_L, LUT4_L

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 is used to connect to another output within the same CLB slice and to the fast connect buffer.

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

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 generally 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 will be used
--             : for simulation.

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

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 Code 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 will be used
--             : for simulation.

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

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 Code 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 will be used
```

```
--             : for simulation.

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

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 Code 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 will be used
--              : for simulation.

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

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 Code for LUT4_L

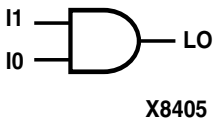
```
//   LUT4_L    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (LUT4_L_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

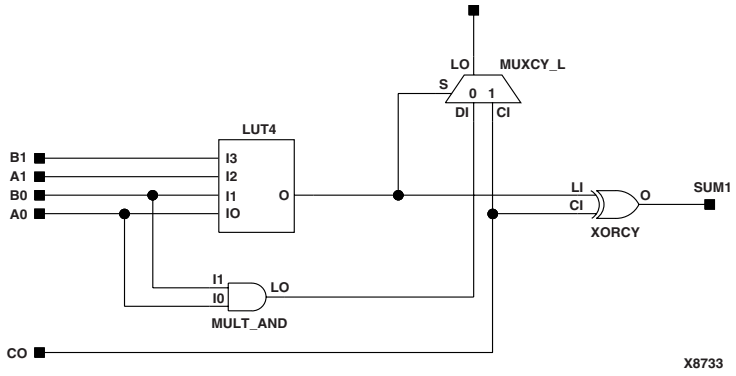
MULT_AND

Primitive: Fast Multiplier AND



MULT_AND is a logical AND gate component that can be used to reduce logic and improve speed when users are 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* be connected 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

For HDL, this design element is instantiated rather than 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 will be used
--             : for simulation.

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

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

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

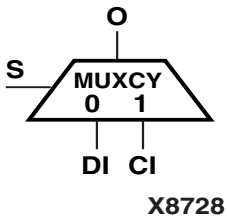
```
//  MULT_AND   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MULT_AND_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXCY

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



MUXCY is used to implement 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) for Virtex-4 devices.

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 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 only be instantiated.

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 will be used
--             : for simulation.

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

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

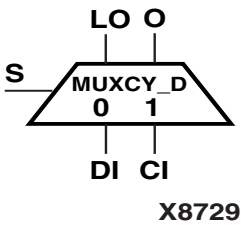
```
//    MUXCY    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXCY_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

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 is 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 is used to 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.

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 will be used
--             : for simulation.

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

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

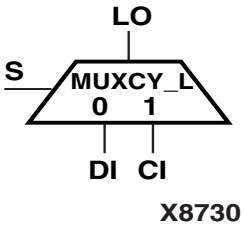
```
//   MUXCY_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXCY_D_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

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

For HDL, this design element can only be instantiated.

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 will be used
--             : for simulation.

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

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

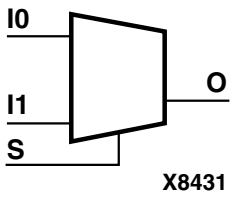
```
//   MUXCY_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXCY_L_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXF5

Primitive: 2-to-1 Lookup 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 only be instantiated.

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 will be used
--             : for simulation.

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

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 Code

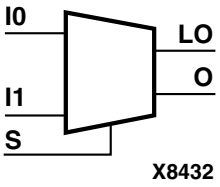
```
//    MUXF5    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF5_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXF5_D

Primitive: 2-to-1 Lookup 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 connects to other inputs in 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.

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 will be used
--             : for simulation.

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

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 Code

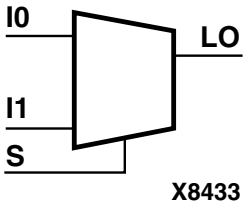
```
//   MUXF5_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF5_D_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXF5_L

Primitive: 2-to-1 Lookup 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 connects to other inputs in 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.

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 will be used
--             : for simulation.

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

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 Code

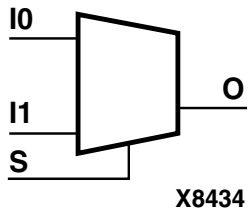
```
//   MUXF5_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF5_L_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXF6

Primitive: 2-to-1 Lookup Table Multiplexer with General Output



MUXF6 provides a multiplexer function in one half of a Virtex-4 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 will be used
--             : for simulation.

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

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 Code

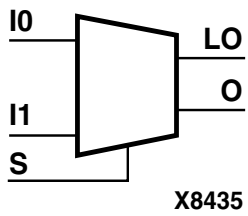
```
//    MUXF6    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF6_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXF6_D

Primitive: 2-to-1 Lookup Table Multiplexer with Dual Output



MUXF6_D provides a multiplexer function in a full Virtex-4 CLB, or one half of a Virtex-4 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.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO output connects to other inputs in the same CLB slice.

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 will be used
--             : for simulation.

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

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 Code

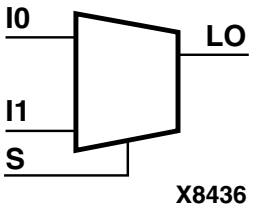
```
//   MUXF6_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF6_D_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXF6_L

Primitive: 2-to-1 Lookup Table Multiplexer with Local Output



MUXF6_L provides a multiplexer function in half of a Virtex-4 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 connects to other inputs in 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 will be used
--             : for simulation.

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

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 Code

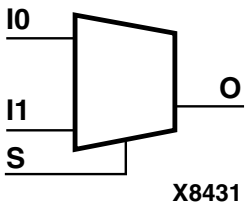
```
//   MUXF6_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF6_L_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

MUXF7

Primitive: 2-to-1 Lookup Table Multiplexer with General Output



MUXF7 provides a multiplexer function in a full Virtex-4 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 will be used
--             : for simulation.

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

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 Code

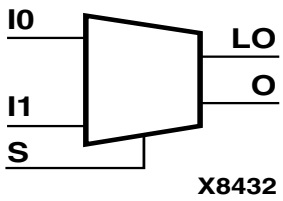
```
//    MUXF7    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF7_inst) and/or the port declarations within the
//    code     : 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 Virtex-II/II-Pro and Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

MUXF7_D

Primitive: 2-to-1 Lookup Table Multiplexer with Dual Output



MUXF7_D provides a multiplexer function in one full Virtex-4 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.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO output connects to other inputs in the same CLB slice.

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 will be used
--             : for simulation.

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

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 Code

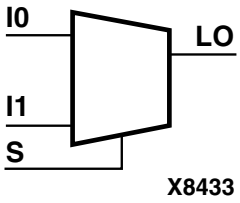
```
//   MUXF7_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF7_D_inst) and/or the port declarations within the
//    code     : 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 Virtex-II/II-Pro and Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

MUXF7_L

Primitive: 2-to-1 Lookup Table Multiplexer with Local Output



MUXF7 provides a multiplexer function in a full Virtex-4 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 LO output connects to other inputs in 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 will be used
--             : for simulation.

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

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 Code

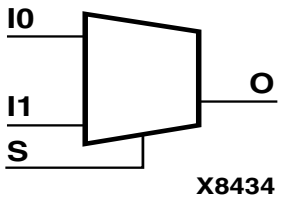
```
//   MUXF7_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF7_L_inst) and/or the port declarations within the
//    code     : 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 Virtex-II/II-Pro and Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

MUXF8

Primitive: 2-to-1 Lookup Table Multiplexer with General Output



MUXF8 provides a multiplexer function in full Virtex-4 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.

The variants, "MUXF8_D"and "MUXF8_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

```
--    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 will be used
--             : for simulation.

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

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 Code

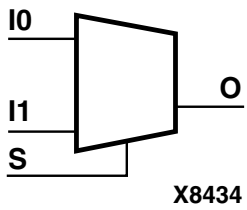
```
//    MUXF8    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF8_inst) and/or the port declarations within the
//    code     : 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 Virtex-II/II-Pro and Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

MUXF8_D

Primitive: 2-to-1 Lookup Table Multiplexer with Dual Output



X8434

MUXF8_D provides a multiplexer function in two full Virtex-4 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 connects to other inputs in 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 will be used
--             : for simulation.

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

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 Code

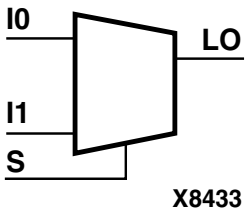
```
//   MUXF8_D   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF8_D_inst) and/or the port declarations within the
//    code     : 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 Virtex-II/II-Pro and Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

MUXF8_L

Primitive: 2-to-1 Lookup Table Multiplexer with Local Output



MUXF8_L provides a multiplexer function in two full Virtex-4 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 in 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

FThis 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 will be used
--              : for simulation.

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

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 Code

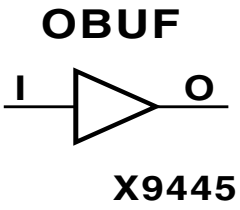
```
//   MUXF8_L   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (MUXF8_L_inst) and/or the port declarations within the
//    code     : 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 Virtex-II/II-Pro and Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

OBUF

Primitive: Single-ended Output Buffer



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 is generally used for specifying 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 the schematic since they will automatically be inserted into the design, if necessary. If you want 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 thus identified as OBUF4, OBUF8, and OBUF16.)

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

```
--    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 will be used
--             : for simulation.

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

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 Code

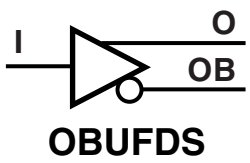
```
//      OBUF       : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (OBUF_inst) and/or the port declarations within the
//      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 *Virtex-4 User Guide*.

OBUFDS

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



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

CAPACITANCE	STRING	"LOW", "NORMAL", "DONT CARE"	"DONT CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
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

```
--   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 will be used
--              : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

```
   -- OBUFDS: Differential Output Buffer
   --         Virtex-II/II-Pro, Spartan-3/3E
   -- 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 Code

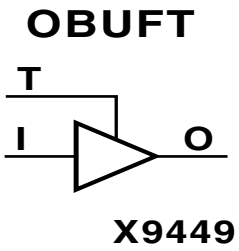
```
//     OBUFDS      : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (OBUFDS_inst) and/or the port declarations within the
//      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
   //         Virtex-II/II-Pro/4, 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 *Virtex-4 User Guide*.

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 tri-state capability, such as the case when building bi-directional I/O.

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

Usage

OBUFTs should be placed in the top-level schematic connecting the output directly to an output or bi-directional port marker.

OBUFTs 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 thus identified as OBUFT4, OBUFT8, and OBUFT16.) The bundles are discussed in the schematics version of the *Virtex-4 Libraries Guide*.

Available Attributes

CAPACITANCE	STRING	"LOW", "NORMAL", "DONT CARE"	"DONT CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
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

```
--    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 will be used
--             : for simulation.

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

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 Libraries Guide Version 8.1i

   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 Code

```
//      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 Libraries Guide Version 8.1i

   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
```

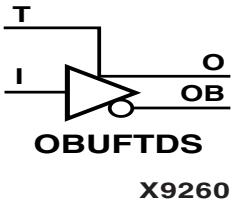
For More Information

Consult the *Virtex-4 User Guide*.

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 SelectIO 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

These design elements are instantiated rather than inferred.

Available Attributes

CAPACITANCE	STRING	"LOW", "NORMAL", "DONT CARE"	"DONT CARE"	Specifies whether it is desired to use an I/O with lower or normal intrinsic capacitance.
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   : 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 (OBUFTDS_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 will be used
--             : for simulation.

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

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

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

   -- OBUFTDS: Differential 3-state Output Buffer
   --          Virtex-II/II-Pro, Spartan-3/3E
   -- Xilinx  HDL Libraries Guide Version 8.1i

   OBUFTDS_inst : OBUFTDS
   generic map (
      IOSTANDARD => "DEFAULT")
   port map (
      O => O,     -- Diff_p output (connect directly to top-level port)
      OB => OB,   -- Diff_n output (connect directly to top-level port)
      I => I,     -- Buffer input
      T => T      -- 3-state enable input
   );

   -- End of OBUFTDS_inst instantiation
```

Verilog Instantiation Code

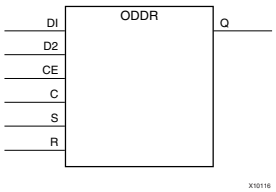
```
//     OBUFTDS   : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (OBUFTDS_inst) and/or the port declarations within the
//      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---->

   // OBUFTDS: Differential 3-state Output Buffer
   //          Virtex-II/II-Pro/4, Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   OBUFTDS #(
      .IOSTANDARD("DEFAULT") // Specify the output I/O standard
   ) OBUFTDS_inst (
      .O(O),     // Diff_p output (connect directly to top-level port)
      .OB(OB),   // Diff_n output (connect directly to top-level port)
      .I(I),     // Buffer input
      .T(T)      // 3-state enable input
   );

   // End of OBUFTDS_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

ODDR

Primitive: A Dedicated Output Register to Transmit Dual Data Rate (DDR) Signals From Virtex-4 FPGAs



ODDR primitives are dedicated output registers used in transmitting dual data rate (DDR) signals from Virtex-4 FPGAs. The ODDR primitive's interface with the FPGA fabric is not limited to opposite edges. ODDR is available with modes that allow data to be presented from the FPGA fabric at the same clock edge. This feature allows designers to avoid additional timing complexities and CLB usage. In addition, ODDR will work in conjunction with SelectIO features of Virtex-4 architecture.

ODDR Ports (Detailed Description)

Q – Data Output (DDR)

This pin connects to the IOB pad.

C – Clock Input Port

The C pin represents the clock input pin.

CE – Clock Enable Port

When asserted LOW, this port disables the output clock at port O.

D1 – D2 – Data Input

This pin is where the DDR data is presented into the ODDR module.

R - Reset

Depends on how SRTYPE is set.

ODDR Modes

The following section describes the functionality of various modes of ODDR. These modes are set by the DDR_CLK_EDGE attribute.

OPPOSITE_EDGE

In the OPPOSITE_EDGE mode, data transmit interface uses the classic DDR methodology. Given a data and clock at pin D1-2 and C respectively, D1 will be sampled at every positive edge of clock C, and D2 will be sampled at every negative edge of clock C. Q changes every clock edge.

SAME_EDGE

In the SAME_EDGE mode, data is still transmitted by opposite edges of clock C. However, both register are clocked with positive clock edge C and an extra register has been placed in front of the D2 input data register. The extra register is clocked with negative clock edge of clock signal C. Using this feature, DDR data can now be presented into the ODDR at the same clock edge.

Port List and Definitions

Name	Type	Width	Function
Q	Output	1	Data Output (DDR)
C	Input	1	Clock Input
CE	Input	1	Clock Enable Input

Name	Type	Width	Function
D1 – D2	Input	1 (each)	Data Input
R	Input	1	Reset
S	Input	1	Set

Available Attributes

DDR_CLK_EDGE	STRING	"OPPOSITE_EDGE", "SAME_EDGE", "SAME_EDGE_PIPELINED"	"OPPOSITE_EDGE"	DDR clock mode recovery mode selection
INIT	INTEGER	0 or 1	1	Q initialization value
SRTYPE	STRING	"SYNC" or "ASYNC"	"SYNC"	Set/Reset type selection

Usage

This design element is available for schematics and instantiation only.

VHDL Instantiation Template

```
--      ODDR       : 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 (ODDR_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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- ODDR: Output Double Data Rate Output Register with Set, Reset
   --       and Clock Enable. Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   ODDR_inst : ODDR
   generic map(
      DDR_CLK_EDGE => "OPPOSITE_EDGE", -- "OPPOSITE_EDGE" or "SAME_EDGE"
      INIT => '0',   -- Initial value for Q port ('1' or '0')
      SRTYPE => "SYNC") -- Reset Type ("ASYNC" or "SYNC")
   port map (
      Q => Q,   -- 1-bit DDR output
      C => C,    -- 1-bit clock input
      CE => CE,  -- 1-bit clock enable input
      D1 => D1,  -- 1-bit data input (positive edge)
      D2 => D2,  -- 1-bit data input (negative edge)
      R => R,    -- 1-bit reset input
      S => S     -- 1-bit set input
   );

   -- End of ODDR_inst instantiation
```

Verilog Instantiation Code

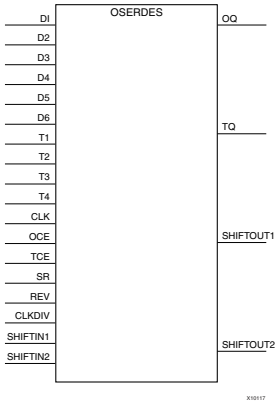
```
//      ODDR     : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (ODDR_inst) and/or the port declarations within the
//      code     : 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---->

   // ODDR: Output Double Data Rate Output Register with Set, Reset
   //       and Clock Enable. Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   ODDR #(
      .DDR_CLK_EDGE("OPPOSITE_EDGE"), // "OPPOSITE_EDGE" or "SAME_EDGE"
      .INIT(1'b0),    // Initial value of Q: 1'b0 or 1'b1
      .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
   ) ODDR_inst (
      .Q(Q),   // 1-bit DDR output
      .C(C),   // 1-bit clock input
      .CE(CE), // 1-bit clock enable input
      .D1(D1), // 1-bit data input (positive edge)
      .D2(D2), // 1-bit data input (negative edge)
      .R(R),   // 1-bit reset
      .S(S)    // 1-bit set
   );

   // End of ODDR_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

OSERDES

Primitive: Dedicated IOB output serializer



The Virtex-4 architecture provides a way for the user to easily implement source synchronous interface by using the OSERDES module. Unlike previous generations of FPGAs, OSERDES is an output architecture that contains a parallel to serial converter resources for both data and tristate. This module helps user by saving logic resources that is otherwise implemented in the FPGA fabric. Furthermore, OSERDES also avoids additional timing complexities that may be encountered when designing such circuitry in the FPGA fabric. OSERDES module is present in all Virtex-4 family of FPGA. In addition, OSERDES contains multiple clock inputs to accommodate various applications and will work in conjunction with the SelectIO features of Virtex-4 family.

OSERDES Ports (Detailed Description)

OQ – Data Path Output

This port is the data output of the OSERDES module. This port connects the output of the data serial to parallel converter to the data input of the IOB pad. In addition, this output port can also be configured to bypass all the submodules within OSERDES module.

SHIFTOUT 1-2 – Data input expansion (slave)

Carry out for data input expansion. Connect to SHIFTIN1/2 of master.

TQ – 3-state Path Output

This port is the 3-state output of the OSERDES module. This port connects the output of the 3-state serial to parallel converter to the control input of the IOB pad.

CLK – High Speed Clock Input

This clock input is used to drive the parallel-to-serial converters. The possible source for the CLK port is from one of the following clock resources:

1. Eight global clock lines in a clock region

2. Two regional clock lines

3. Six clock capable I/Os (within adjacent clock region)

4. Fabric (through bypass)

CLKDIV – Divided High Speed Clock Input

This clock input is used to drive the parallel-to-serial converters. This clock has to have slower frequency than the clock connected to the CLK port. The possible source for the CLKDIV port is from one of the following clock resources:

1. Eight global clock lines in a clock region

2. Two regional clock lines

D1-D6 – Parallel Data Inputs

Ports D1 to D6 are the location in which all incoming parallel data enters the OSERDES module. This port is connected to the FPGA fabric, and can be configured from 2 to 6 bits. In the extended width mode, this port can be expanded up to 10 bits.

OCE – Output Data Clock Enable

This port is used to enables the output of the data serial to parallel converter when asserted HIGH.

REV - Reverse SR pin

When SR is used, a second input, REV forces the storage element into the opposite state. The reset condition predominates over the set condition. See “Set/Reset Input – SR” for the truth table that describes the REV operation with respect to SR.

SR - Set/Reset Input

The set/reset pin, SR forces the storage element into the state specified by the SRVAL attribute. SRVAL = “1” forces a logic 1. SRVAL =”0” forces a logic "0." When SR is used, a second input (REV) forces the storage element into the opposite state. The reset condition predominates over the set condition. The following truth tables describe the operation of SR in conjunction with REV.

The Truth Table when SRVAL = ”0” (Default Condition)

SR	REV	Function
0	0	NOP
0	1	Set
1	0	Reset
1	1	Reset

The Truth Table when SRVAL = “1”

SR	REV	Function
0	0	NOP
0	1	Reset
1	0	Set
1	1	Reset

SHIFTIN 1-2 – Data input expansion (master)

Carry input for data input expansion. Connect to SHIFTOUT1/2 of slave.

T1-T4 – Parallel 3-state Inputs

Ports T1 to T4 are the location in which all parallel tristate signals enters the OSERDES module. This port is connected to the FPGA fabric, and can be configured from 1 to 4 bits. This feature is not supported in the extended width mode.

TCE – 3-state Signal Clock Enable

This port is used to enables the output of the tristate signal serial to parallel converter when asserted HIGH.

Usage

Parallel-to-Serial Converter (Data)

The data parallel to serial converter in the OSERDES module takes in 2 to 6 bit of parallel data and convert them into serial data. Data input widths larger than 6 (7,8, and 10) is achievable by cascading two OSERDES modules for data width expansion. In order to do this, one OSERDES must be set into a MASTER mode, while another is set into SLAVE mode. The user will also need to connect the SHIFTOUT of "slave" and

SHIFTIN of "master" ports together. The "slave" will only use D3 to D6 ports as its input. The parallel to serial converter is available for both SDR and DDR modes.

This module is designed such that the data input at D1 port is the first output bit. This module is controlled by CLK and CLKDIV clocks. The following table describes the relationship between CLK and CLKDIV for both SDR and DDR mode.

SDR Data Width	DDR Data Width	CLK	CLKDIV
2	4	2X	X
3	6	3X	X
4	8	4X	X
5	10	5X	X
6	-	6X	X
7	-	7X	X
8	-	8X	X

Output of this block is connected to the data input of an IOB pad of the FPGA. This IOB pad can be configured to a desired standard using SelectIO.

Parallel-to-Serial Converter (3-state)

The 3-state parallel-to-serial converter in the OSERDES module takes in up to 4 bits of parallel 3-state signals and converts them into serial3-state signals. Unlike data parallel-to-serial converters, the 3-state parallel-to-serial converters are not extendable to more than 4-bit 3-state signals. This module is primarily controlled by CLK and CLKDIV clocks. In order to use this module, the following attributes must be declared: DATA_RATE_TQ and TRISTATE_WIDTH. In certain cases, the user may also need to declare DATA_RATE_OQ and DATA_WIDTH. The following table lists the attributes needed for the desired functionality.

Mode of Operation	DATA_RATE_TQ	TRISTATE_WIDTH
4-bit DDR*	DDR	4
2-bit DDR	DDR	2
1-bit SDR	SDR	1
Buffer	BUF	1

Note: If 4-bit DDR is chosen, then the constraints DATA_RATE_OQ and DATA_WIDTH must be {SDR,2} or {DDR,4} for proper operation.

Output of this block is connected to the tristate input of an IOB pad of the FPGA. This IOB pad can be configured to a desired standard using SelectIO.

Width Expansion

It is possible to use the OSERDES modules to transmit parallel data widths larger than six. However, the tristate output is not expandable. In order to use this feature, two OSERDES modules need to be instantiated. Both the OSERDES must be an adjacent master and slave pair. The attribute MODE must be set to either "MASTER" or "SLAVE" in order to differentiate the modes of the ISERDES pair. In addition, the user must connect the SHIFTIN ports of the MASTER to the SHIFTOUT ports of the SLAVE. This feature supports data widths of 7, 8, and 10 for SDR and DDR mode. The table below lists the data width availability for SDR and DDR mode.

.

Mode	Widths
SDR Data Widths	2,3,4,5,6,7,8
DDR Data Widths	4,6,8,10

Port List and Definitions

Name	Type	Width	Function
OQ	Output	1	Data path output
SHIFTOUT1-2	Output	1 (each)	Carry out for data input expansion. Connect to SHIFTIN1/2 of master.
TQ	Output	1	Tristate path output
CLK	Input	1	High speed clock input
CLKDIV	Input	1	Divided high speed clock input
D1-D6	Input	1	Data path inputs
OCE	Input	1	Parallel to serial converter (data) clock enable
REV	Input	1	Reverse SR
SR	Input	1	Set/Reset
SHIFTIN1-2	Input	1 (each)	Carry input for data input expansion. Connect to SHIFTOUT1/2 of slave.
T1 – T4	Input	1 (each)	Tristate inputs
TCE	Input	1	Parallel to serial converter (tristate) clock enable

Available Attributes

DATA_RATE_OQ	STRING	"SDR" or "DDR"	"DDR"	Defines whether the data changes at every clock edge or every positive clock edge with respect to CLK.
DATA_RATE_TQ	STRING	"BUF", "SDR", "DDR"	"DDR"	Defines whether the 3-state changes at every clock edge, every positive clock edge, or buffer configuration with respect to CLK.
DATA_WIDTH	STRING	If DATA_RATE_OQ = "DDR", value is limited to 4,6,8, or 10. If DATA_RATE_OQ = "SDR", value is limited to 2,3,4,5,6,7, or 8.	4	Defines the parallel to serial data converter width. This value also depends on the DATA_RATE_OQ value of the OSERDES
INIT_OQ	1-Bit Binary	1-Bit Binary	1'b0	Defines the initial value of OQ output
INIT_TQ	1-Bit Binary	1-Bit Binary	1'b0	Defines the initial value of TQ output
SERDES_MODE	STRING	"MASTER" or "SLAVE"	"MASTER"	Defines whether the OSERDES module is a master or slave when width expansion is used
SRVAL_OQ	1-Bit Binary	1-Bit Binary	1'b0	Defines the value of OQ output when reset is invoked

SRVAL_TQ	1-Bit Binary	1-Bit Binary	1'b0	Defines the value of TQ output when reset is invoked
TRISTATE_WIDTH	STRING	If DATA_RATE_TQ = "DDR", value is limited to 2 and 4. If DATA_RATE_TQ = "SDR" or "BUF", value is limited 1.	4	Specify parallel to serial converter width. When DATA_RATE_TQ = DDR: 2 or 4. When DATA_RATE_TQ = SDR or BUF: 1.

VHDL Instantiation Template

```
--     OSERDES   : 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 (OSERDES_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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- OSERDES: Output SERDES
   --          Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i

   OSERDES_inst : OSERDES
   generic map (
      DATA_RATE_OQ => "DDR", -- Specify data rate to "DDR" or "SDR"
      DATA_RATE_TQ => "DDR", -- Specify data rate to "DDR", "SDR", or "BUF"
      DATA_WIDTH => 4, -- Specify data width - For DDR: 4,6,8, or 10
                       -- For SDR or BUF: 2,3,4,5,6,7, or 8
      INIT_OQ => '0',  -- INIT for Q1 register - '1' or '0'
      INIT_TQ => '0',  -- INIT for Q2 register - '1' or '0'
      SERDES_MODE => "MASTER", --Set SERDES mode to "MASTER" or "SLAVE"
      SRVAL_OQ => '0', -- Define Q1 output value upon SR assertion - '1' or '0'
      SRVAL_TQ => '0', -- Define Q1 output value upon SR assertion - '1' or '0'
      TRISTATE_WIDTH => 4) -- Specify parallel to serial converter width
                           -- When DATA_RATE_TQ = DDR: 2 or 4
                           -- When DATA_RATE_TQ = SDR or BUF: 1 "
   port map (
      OQ => OQ,     -- 1-bit output
      SHIFTOUT1 => SHIFTOUT1, -- 1-bit output
      SHIFTOUT2 => SHIFTOUT2, -- 1-bit output
      TQ => TQ,     -- 1-bit onput
      CLK => CLK,   -- 1-bit input
      CLKDIV => CLKDIV,  -- 1-bit input
      D1 => D1,     -- 1-bit input
      D2 => D2,     -- 1-bit input
      D3 => D3,     -- 1-bit input
      D4 => D4,     -- 1-bit input
      D5 => D5,     -- 1-bit input
      D6 => D6,     -- 1-bit input
      OCE => OCE,   -- 1-bit input
      REV => REV,   -- 1-bit input
      SHIFTIN1 => SHIFTIN1, -- 1-bit input
      SHIFTIN2 => SHIFTIN2, -- 1-bit input
      SR => SR,     -- 1-bit input
```

```
      T1 => T1,    -- 1-bit input
      T2 => T2,    -- 1-bit input
      T3 => T3,    -- 1-bit input
      T4 => T4,    -- 1-bit input
      TCE => TCE   -- 1-bit input
   );

   -- End of OSERDES_inst instantiation
```

Verilog Instantiation Code

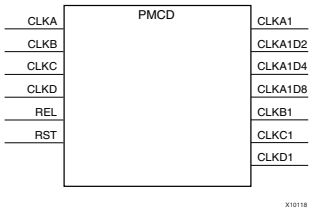
```
//     OSERDES   : In order to incorporate this function into the design,
//     Verilog   : the following instance declaration needs to be placed
//    instance   : in the body of the design code.  The instance name
//   declaration : (OSERDES_inst) and/or the port declarations within the
//      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---->

   // OSERDES: Source Synchronous Output Serializer
   //          Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   OSERDES #(
      .DATA_RATE_OQ("DDR"), // Specify data rate to "DDR" or "SDR"
      .DATA_RATE_TQ("DDR"), // Specify data rate to "DDR", "SDR", or "BUF"
      .DATA_WIDTH(4), // Specify data width - For DDR: 4,6,8, or 10
                      //     For SDR or BUF: 2,3,4,5,6,7, or 8
      .INIT_OQ(1'b0), // INIT for OQ register - 1'b1 or 1'b0
      .INIT_TQ(1'b0), // INIT for OQ register - 1'b1 or 1'b0
      .SERDES_MODE("MASTER"), // Set SERDES mode to "MASTER" or "SLAVE"
      .SRVAL_OQ(1'b0), // Define OQ output value upon SR assertion - 1'b1 or 1'b0
      .SRVAL_TQ(1'b0), // Define TQ output value upon SR assertion - 1'b1 or 1'b0
      .TRISTATE_WIDTH(4)  // Specify parallel to serial converter width
                          //    When DATA_RATE_TQ = DDR: 2 or 4
                          //    When DATA_RATE_TQ = SDR or BUF: 1
   ) OSERDES_inst (
      .OQ(OQ),    // 1-bit data path output
      .SHIFTOUT1(SHIFTOUT1), // 1-bit data expansion output
      .SHIFTOUT2(SHIFTOUT2), // 1-bit data expansion output
      .TQ(TQ),    // 1-bit 3-state control output
      .CLK(CLK),        // 1-bit clock input
      .CLKDIV(CLKDIV),  // 1-bit divided clock input
      .D1(D1),   // 1-bit parallel data input
      .D2(D2),   // 1-bit parallel data input
      .D3(D3),   // 1-bit parallel data input
      .D4(D4),   // 1-bit parallel data input
      .D5(D5),   // 1-bit parallel data input
      .D6(D6),   // 1-bit parallel data input
      .OCE(OCE), // 1-bit clock enable input
      .REV(REV), // 1-bit reverse SR input
      .SHIFTIN1(SHIFTIN1),  // 1-bit data expansion input
      .SHIFTIN2(SHIFTIN2), // 1-bit data expansion input
      .SR(SR),     // 1-bit set/reset input
      .T1(DLYINC), // 1-bit parallel 3-state input
      .T2(DLYRST), // 1-bit parallel 3-state input
      .T3(OCLK),   // 1-bit parallel 3-state input
      .T4(REV),    // 1-bit parallel 3-state input
      .TCE(TCE)    // 1-bit 3-state signal clock enable input
   );

   // End of OSERDES_inst instantiation
);
```

For More Information

Consult the *Virtex-4 User Guide*.

PMCD

Primitive: Phase-Matched Clock Divider



The Phase-Matched Clock Dividers (PMCDs) are one of the clock resources available in the Virtex-4 architecture. PMCDs provide the following clock management features:

- ***Phase-Aligned Divided Clocks***

  The phase-aligned divided clocks create up to four frequency-divided and phse-aligned versions of an input clock, CLKA. The output clocks are a function of the input clock frequency: divided-by-1 (CLKA1), divided-by-2 (CLKAD2), divided-by-4 (CLKA1D4), and divided-by-8 (CLKA1D8). CLKA1, CLKA1D2, CLKA1D4, CLKA1D8 output clocks are rising-edge aligned.

- ***Matched-Clock Phase***

  The matched-clock phase preserves edge alignments, phase relations, or skews between the input clock CLKA and other PMCD linput clocks. Three additional input clocks (CLKB, CLKC, and CLKD) and three corresponding delayed output clocks (CLKB1, CLKC1, and CLKD1) are available. The same delay is inserted to CLKA, CLKB, CLKC, and CLKD; thus, the delayed CLKA1, CLKB1, CLKC1, and CLKD1 clock outputs maintain edge alignments, phase relations, and the skews of the respective inputs.

A PMCD can be used with other clock reosurces, including global buffers and the digital clock management feature. Together, these clock resources provide flexibility in managing complex clock networks in designs.

Port Descriptions

PMCD Port Description

Port Name	Direction	Description
CLKA	Input	CLKA is a clock input to the PMCD. The CLKA frequency can be divided by 1, 2, 4, and 8.
CLKB CLKC CLKD	Input	CLKB, CLKC, and CLKD are clock inputs to the PMCD. These clock are not divided by PMCD, however, they are delayed by the PMCD to maintain the phase alignment and phase relationship to CLKA.
RST	Input	RST is the reset input to the PMCD. Asserting the RST signal asynchronously forces all outputs Low. Deasserting RST synchronously allows all outputs to toggle.
REL	Input	REL is the release input to the PMCD. Asserting the REL signal releases the divided output synchronous to CLKA.
CLKA1	Output	The CLKA1 output has the same frequency as the CLKA input. It is a delayed version of CLKA.
CLKA1D2	Output	The CLKA1D2 output has the frequency of CLKA divided by two. CLKA1D2 is rising-edge aligned to CLKA1.

Port Name	Direction	Description
CLKA1D4	Output	The CLKA1D4 output has the frequency of CLKA divided by four. CLKA1D4 is rising-edge aligned to CLKA1.
CLKA1D8	Output	The CLKA1D8 output has the frequency of CLKA divided by eight, CLKA1D8 is rising-edge aligned to CLKA1.
CLKB1 CLKC1 CLKD1	Output	The CLKB1 output is has the same frequency as the CLKB input, a delayed version of CLKB. The skew between CLKB1 and CLKA1 is the same as the skew between CLKB and CLKA inputs. Similarly, CLKC1 is a delayed version of CLKC, and CLKD1 is a delayed version of CLKD.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
EN_REL	BOOLEAN	FALSE, TRUE	0	This attribute allows for CLKA1D2, CLKA1D4, and CLKA1D8 outputs to be released at REL signal assertion. Note: REL is synchronous to CLKA input.
RST_DEASSERT_CLK	STRING	"CLKA", "CLKB", "CLKC", or "CLKD"	"CLKA"	This attribute allows the deassertion of the RST signal to be synchronous to a selected PMCD input clock.

VHDL Instantiation Template

```
--    PMCD     : 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 (PMCD_inst) and/or the port declarations
--    code     : after the "=>" assignment maybe changed to properly
--             : connect this function to the design.  Unused inputs
--             : and outputs may 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 will be used
--             : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- PMCD: Phase-Matched Clock Divider Circuit for Virtex-4
   -- Xilinx HDL Libraries Guide Version 8.1i

   PMCD_inst : PMCD
   generic map (
      EN_REL => FALSE,             -- TRUE/FALSE to allow synchronous deassertion of RST
      RST_DEASSERT_CLK => "CLKA")  -- Reset syncronization to which clock: CLKA, CLKB, CLKC or CLKD
   port map (
      CLKA1 => CLKA1, -- Output CLKA divided by 1
      CLKA1D2 => CLKA1D2, -- Output CLKA divided by 2
      CLKA1D4 => CLKA1D4, -- Output CLKA divided by 4
      CLKA1D8 => CLKA1D8, -- Output CLKA divided by 8
      CLKB1 => CLKB1, -- Output phase matched CLKB
      CLKC1 => CLKC1, -- Output phase matched CLKC
      CLKD1 => CLKD1, -- Output phase matched CLKD
      CLKA => CLKA,   -- Input CLKA
      CLKB => CLKB,   -- Input CLKB
```

```
      CLKC => CLKC,    -- Input CLKC
      CLKD => CLKD,    -- Input CLKD
      REL => REL,      -- PCMD release input
      RST => RST       -- Active high reset input
   );

   -- End of PMCD_inst instantiation
```

Verilog Instantiation Code

```
//    PMCD     : 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 : (PMCD_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 may be removed or commented out.

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

   // PMCD: Phase-Matched Clock Divider Circuit for Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   PMCD #(
      .EN_REL("FALSE"),  // TRUE/FALSE to allow synchronous deassertion of RST
      .RST_DEASSERT_CLK("CLKA") // Reset syncronization to which clock: CLKA, CLKB, CLKC or CLKD
   ) PMCD_inst (
      .CLKA1(CLKA1),     // Output CLKA divided by 1
      .CLKA1D2(CLKA1D2), // Output CLKA divided by 2
      .CLKA1D4(CLKA1D4), // Output CLKA divided by 4
      .CLKA1D8(CLKA1D8), // Output CLKA divided by 8
      .CLKB1(CLKB1),     // Output phase matched CLKB
      .CLKC1(CLKC1),     // Output phase matched CLKC
      .CLKD1(CLKD1),     // Output phase matched CLKD
      .CLKA(CLKA),       // Input CLKA
      .CLKB(CLKB),       // Input CLKB
      .CLKC(CLKC),       // Input CLKC
      .CLKD(CLKD),       // Input CLKD
      .REL(REL),         // PCMD release input
      .RST(RST)          // Active high reset input
   );

   // End of PMCD_inst instantiation
```

For More Information

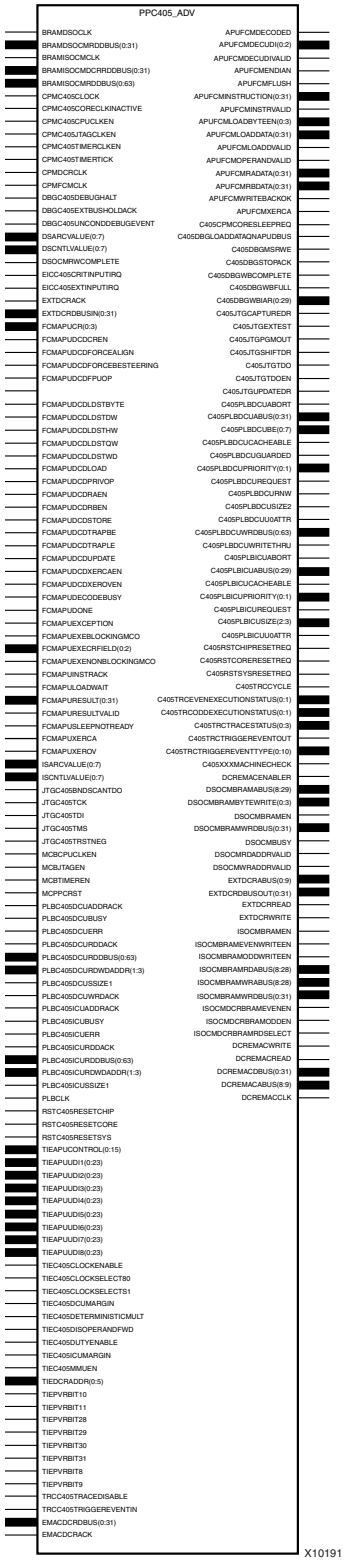
Consult the *Virtex-4 User Guide*.

PPC405_ADV

Primitive: For the Power PC Core

The PowerPC 405 is a 32-bit implementation of the PowerPC embedded environment architecture that is derived from the PowerPC architecture. Specifically, the PowerPC 405 is an embedded PowerPC 405F6, for Virtex-4 devices, processor core. The processor core also contains on-chip memory logic (OCM), an APU controller (Virtex-4 only), and the gasket logic and interface.

The PowerPC architecture provides a software model that ensures compatibility between implementations of the PowerPC family of microprocessors. The PowerPC architecture defines parameters that guarantee compatible processor implementations at the application-program level, allowing broad flexibility in the development derivative PowerPC implementations that meet specific market requirements.



PPC405_ADV Schematic

Inputs and Outputs

Inputs	Outputs
BRAMDSOCMCLK	APUFCMDECODED
BRAMDSOCMRDDBUS [0:31]	APUFCMDECUDI [0:2]
BRAMISOCMCLK	APUFCMDECUDIVALID
BRAMISOCMDCRRDDBUS [0:31]	APUFCMENDIAN
BRAMISOCMRDDBUS [0:63]	APUFCMFLUSH
CPMC405CLOCK	APUFCMINSTRUCTION [0:31]
CPMC405CORECLKINACTIVE	APUFCMINSTRVALID
CPMC405CPUCLKEN	APUFCMLOADBYTEEN [0:3]
CPMC405JTAGCLKEN	APUFCMLOADDATA [0:31]
CPMC405SYNCBYPASS	APUFCMLOADDVALID
CPMC405TIMERCLKEN	APUFCMOPERANDVALID
CPMC405TIMERTICK	APUFCMRADATA [0:31]
CPMDCRCLK	APUFCMRBDATA [0:31]
CPMFCMCLK	APUFCMWRITEBACKOK
DBGC405DEBUGHALT	APUFCMXERCA
DBGC405EXTBUSHOLDACK	C405CPMCORESLEEPREQ
DBGC405UNCONDDEBUGEVENT	C405CPMMSRCE
DSARCVALUE [0:7]	C405CPMMSREE
DSCNTLVALUE [0:7]	C405CPMTIMERIRQ
DSOCMRWCOMPLETE	C405CPMTIMERRESETREQ
EICC405CRITINPUTIRQ	C405DBGLOADDATAONAPUDBUS
EICC405EXTINPUTIRQ	C405DBGMSRWE
EMACDCRACK	C405DBGSTOPACK
EMACDCRDBUS [0:31]	C405DBGWBCOMPLETE
EXTDCRACK	C405DBGWBFULL
EXTDCRDBUSIN [0:31]	C405DBGWBIAR [0:29]
FCMAPUCR [0:3]	C405JTGCAPTUREDR
FCMAPUDCDCREN	C405JTGEXTEST
FCMAPUDCDFORCEALIGN	C405JTGPGMOUT
FCMAPUDCDFORCEBESTEERING	C405JTGSHIFTDR
FCMAPUDCDFPUOP	C405JTGTDO
FCMAPUDCDGPRWRITE	C405JTGTDOEN
FCMAPUDCDLDSTBYTE	C405JTGUPDATEDR
FCMAPUDCDLDSTDW	C405PLBDCUABORT
FCMAPUDCDLDSTHW	C405PLBDCUABUS [0:31]
FCMAPUDCDLDSTQW	C405PLBDCUBE [0:7]
FCMAPUDCDLDSTWD	C405PLBDCUCACHEABLE
FCMAPUDCDLOAD	C405PLBDCUGUARDED
FCMAPUDCDPRIVOP	C405PLBDCUPRIORITY [0:1]
FCMAPUDCDRAEN	C405PLBDCUREQUEST

Inputs	Outputs
FCMAPUDCDRBEN	C405PLBDCURNW
FCMAPUDCDSTORE	C405PLBDCUSIZE2
FCMAPUDCDTRAPBE	C405PLBDCUU0ATTR
FCMAPUDCDTRAPLE	C405PLBDCUWRDBUS [0:63]
FCMAPUDCDUPDATE	C405PLBDCUWRITETHRU
FCMAPUDCDXERCAEN	C405PLBICUABORT
FCMAPUDCDXEROVEN	C405PLBICUABUS [0:29]
FCMAPUDECODEBUSY	C405PLBICUCACHEABLE
FCMAPUDONE	C405PLBICUPRIORITY [0:1]
FCMAPUEXCEPTION	C405PLBICUREQUEST
FCMAPUEXEBLOCKINGMCO	C405PLBICUSIZE [2:3]
FCMAPUEXECRFIELD [0:2]	C405PLBICUU0ATTR
FCMAPUEXENONBLOCKINGMCO	C405RSTCHIPRESETREQ
FCMAPUINSTRACK	C405RSTCORERESETREQ
FCMAPULOADWAIT	C405RSTSYSRESETREQ
FCMAPURESULT [0:31]	C405TRCCYCLE
FCMAPURESULTVALID	C405TRCEVENEXECUTIONSTATUS [0:1]
FCMAPUSLEEPNOTREADY	C405TRCODDEXECUTIONSTATUS [0:1]
FCMAPUXERCA	C405TRCTRACESTATUS [0:3]
FCMAPUXEROV	C405TRCTRIGGEREVENTOUT
ISARCVALUE [0:7]	C405TRCTRIGGEREVENTTYPE [0:10]
ISCNTLVALUE [0:7]	C405XXXMACHINECHECK
JTGC405BNDSCANTDO	DCREMACABUS [8:9]
JTGC405TCK	DCREMACCLK
JTGC405TDI	DCREMACDBUS [0:31]
JTGC405TMS	DCREMACENABLER
JTGC405TRSTNEG	DCREMACREAD
MCBCPUCLKEN	DCREMACWRITE
MCBJTAGEN	DSOCMBRAMABUS [8:29]
MCBTIMEREN	DSOCMBRAMBYTEWRITE [0:3]
MCPPCRST	DSOCMBRAMEN
PLBC405DCUADDRACK	DSOCMBRAMWRDBUS [0:31]
PLBC405DCUBUSY	DSOCMBUSY
PLBC405DCUERR	DSOCMRDADDRVALID
PLBC405DCURDDACK	DSOCMWRADDRVALID
PLBC405DCURDDBUS [0:63]	EXTDCRABUS [0:9]
PLBC405DCURDWDADDR [1:3]	EXTDCRDBUSOUT [0:31]
PLBC405DCUSSIZE1	EXTDCRREAD
PLBC405DCUWRDACK	EXTDCRWRITE
PLBC405ICUADDRACK	ISOCMBRAMEN
PLBC405ICUBUSY	ISOCMBRAMEVENWRITEEN

Inputs	Outputs
PLBC405ICUERR	ISOCMBRAMODDWRITEEN
PLBC405ICURDDACK	ISOCMBRAMRDABUS [8:28]
PLBC405ICURDDBUS [0:63]	ISOCMBRAMWRABUS [8:28]
PLBC405ICURDWDADDR [1:3]	ISOCMBRAMWRDBUS [0:31]
PLBC405ICUSSIZE1	ISOCMDCRBRAMEVENEN
PLBCLK	ISOCMDCRBRAMODDEN
RSTC405RESETCHIP	ISOCMDCRBRAMRDSELECT
RSTC405RESETCORE	
RSTC405RESETSYS	
TIEAPUCONTROL [0:15]	
TIEAPUUDI1 [0:23]	
TIEAPUUDI2 [0:23]	
TIEAPUUDI3 [0:23]	
TIEAPUUDI4 [0:23]	
TIEAPUUDI5 [0:23]	
TIEAPUUDI6 [0:23]	
TIEAPUUDI7 [0:23]	
TIEAPUUDI8 [0:23]	
TIEC405DETERMINISTICMULT	
TIEC405DISOPERANDFWD	
TIEC405MMUEN	
TIEDCRADDR [0:5]	
TIEPVRBIT10	
TIEPVRBIT11	
TIEPVRBIT28	
TIEPVRBIT29	
TIEPVRBIT30	
TIEPVRBIT31	
TIEPVRBIT8	
TIEPVRBIT9	
TRCC405TRACEDISABLE	
TRCC405TRIGGEREVENTIN	

Usage

Refer to the EDK software for information regarding the use of this component.

VHDL/Verilog Instantiation

Use the Embedded Development Kit (EDK) in order to generate and instantiate these components.

For More Information

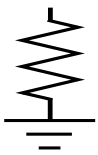
For complete information about the PowerPC 405 in Virtex-4 devices, see the following documents:

- Virtex-4 Data Sheet
- PowerPC 405 Processor Block Reference Guide (for Virtex-4).

More information about this element can be found in the *Virtex-4 User Guide*.

PULLDOWN

Primitive: Resistor to GND for Input Pads



X3860

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

Usage

For HDL, the PULLDOWN 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 will be used
--             : for simulation.

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

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, CoolRunner-II
   -- 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 Code

```
//    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, CoolRunner-II
   // 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 *Virtex-4 User Guide*.

PULLUP

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

X3861

The pull-up elements establish a High logic level for open-drain elements and macros (DECODE, WAND, WORAND) or 3-state nodes (TBUF) when all the drivers are set to off.

The buffer outputs are connected together as a wired-AND to form the output (O). When all the inputs are High, the output is off. To establish an output High level, a PULLUP resistor is tied to output (O). One PULLUP resistor uses the least power, two pull-up resistors achieve the fastest Low-to-High speed.

To indicate two PULLUP resistors, append a DOUBLE parameter to the pull-up symbol attached to the output (O) node.

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 will be used
--             : for simulation.

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

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, CoolRunner-II
   -- 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 Code

```
//     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---->
```

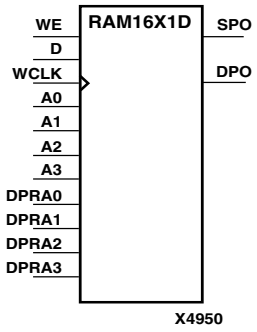
```
// PULLUP: I/O Buffer Weak Pull-up
//          All FPGA, CoolRunner-II
// Xilinx HDL Libraries Guide Version 8.1i

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

// End of PULLUP_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

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 Virtex-4 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	64-Bit Hexadecimal	64-Bit Hexadecimal	16'h0000000000000000	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 will be used
--             : for simulation.

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

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 Code

```
//   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 (
```

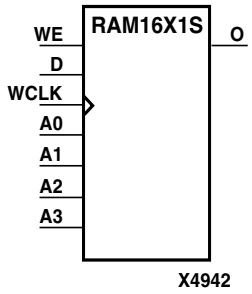
```
    .DPO(DPO),      // Port A 1-bit data output
    .SPO(SPO),      // Port B 1-bit data output
    .A0(A0),        // Port A address[0] input bit
    .A1(A1),        // Port A address[1] input bit
    .A2(A2),        // Port A address[2] input bit
    .A3(A3),        // Port A address[3] input bit
    .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 *Virtex-4 User Guide*.

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 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 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	64-Bit Hexadecimal	64-Bit Hexadecimal	16'h0000000000000000	Initializes ROMs, RAMs, registers, and look-up tables.

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 will be used
--             : for simulation.
```

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

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 Code

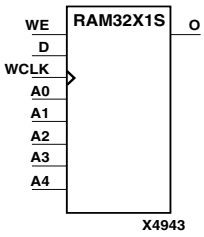
```
//  RAM16X1S   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (RAM16X1S_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

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 Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 5-bit address (A4 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM32X1S during configuration using the INIT attribute. 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

For HDL, 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_00 To INIT_07	INTEGER	0, 1, 2, 3, 4, 5, 6, or 7	0	Initializes ROMs, RAMs, registers, and look-up tables.

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 will be used
--             : for simulation.
```

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

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 Code

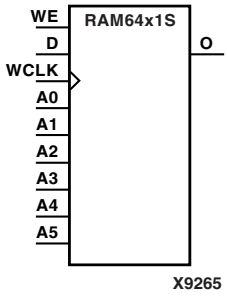
```
//  RAM32X1S   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (RAM32X1S_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

RAM64X1S

Primitive: 64-Deep by 1-Wide Static Synchronous RAM



RAM64X1S is a 64-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any 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

For HDL, 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 Hexadecimal	64-Bit Hexadecimal	64'h0000000000000000	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 will be used
--             : for simulation.
```

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

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
   --           Virtex-II/II-Pro, Spartan-3/3E
   -- 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 Code

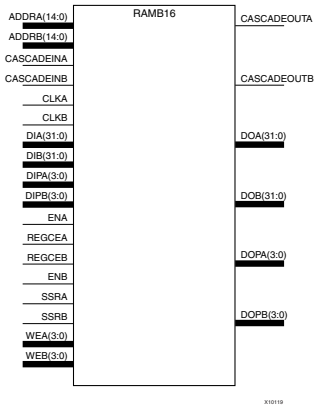
```
//  RAM64X1S   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (RAM64X1S_inst) and/or the port declarations within the
//    code     : 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
   //           Virtex-II/II-Pro/4, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

RAMB16

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



In addition to distributed RAM memory, Virtex-4 devices feature a large numberof 18 Kb block RAM memories. The block RAM memory is a True Dual-Port™ RAM, offering fast, discrete, and large blocks of memory in the device. The memory is organized in columns, and the total amount of block RAM memory depends on the size of the Virtex-4 device. The 18 Kb blocks are cascadable to enable a deeper and wider memory implementation, with a minimal timing penalty incurred through specialized routing resources.

Available Attributes

Table 4-4: Block RAM Initialization Attributes		Memory Cell	
	from		to
INIT_00	255		0
INIT_01	511		256
INIT_02	767		512
…	…		…
INIT_0E	3839		3584
INIT_0F	4095		3840
INIT_10	4351		4096
…	…		…
INIT_1F	8191		7936
INIT_20	8447		8192
…	…		…
INIT_2F	12287		12032
INIT_30	12543		12288
..	…		…
INIT_3F	16383		16128

Content Initialization - INIT_xx

INIT_xx attributes define the initial memory contents. By default block RAM memory is initialized with all zeros during the device configuration sequence. The 64 initialization attributes from INIT_00 through INIT_3F represent the regular memory contents. Each INIT_xx is a 64-digit hex-encoded bit vector. The memory contents can be partially initialized and are automatically completed with zeros.

Content Initialization - INITP_xx

INITP_xx attributes define the initial contents of the memory cells corresponding to DIP/DOP buses (parity bits). By default these memory cells are also initialized to all zeros. The eight initialization attributes from INITP_00 through INITP_07 represent the memory contents of parity bits. Each INITP_xx is a 64-digit hex-encoded bit vector

with a regular INIT_xx attribute behavior. The same formula can be used to calculate the bit positions initialized by a particular INITP_xx attribute.

Output Latches Initialization - INIT (INIT_A & INIT_B)

The INIT_A and INIT_B (dual-port) attributes define the output latches values after configuration. The width of the INIT (INIT_A & INIT_B) attribute is the port width, as shown in Table 4-5. These attributes are hex-encoded bit vectors and the default value is 0.

Output Latches Synchronous Set/Reset - SRVAL (SRVAL_A & SRVAL_B)

The SRVAL_A and SRVAL_B (dual-port) attributes define output latch values when the SSR input is asserted. The width of the SRVAL (SRVAL_A and SRVAL_B) attribute is the port width, as shown in the following table:

Table 4-5: Port Width Values Port Data Width	DOP Bus	DO Bus	INIT / SRVAL
1	NA	<0>	1
2	NA	<1:0>	2
4	NA	<3:0>	4
9	<0>	<7:0>	(1 + 8) = 9
18	<1:0>	<15:0>	(2 + 16) = 18
36	<3:0>	<31:0>	(4 + 32) = 36

Optional Output Register On/Off Switch - DO[A/B]_REG

This attribute sets the number of pipeline register at A/B output of RAMB16. The valid values are 0 (default) or 1.

Clock Inversion at Output Register Switch - INVERT_CLK_DO[A/B]_REG

When set to TRUE, the clock input to the pipeline register at A/B output of RAMB16 is inverted. The default value is FALSE.

Extended Mode Address Determinant - RAM_EXTENSION_[A/B]

This attribute determines whether the block RAM of interest has its A/B port as UPPER/LOWER address when using the cascade mode. When the block RAM is not used in cascade mode, the default value is NONE.

Read Width - READ_WIDTH_[A/B]

This attribute determines the A/B read port width of the block RAM. The valid values are: 0 (default), 1, 2, 4, 9, 18, and 36.

Write Width - WRITE_WIDTH_[A/B]

This attribute determines the A/B write port width of the block RAM. The valid values are: 0 (default), 1, 2, 4, 9, 18, and 36.

Write Mode - WRITE_MODE_[A/B]

This attribute determines the write mode of the A/B input ports. The possible values are WRITE_FIRST (default), READ_FIRST, and NO_CHANGE.

RAMB16 Location Constraints

Block RAM instances can have LOC properties attached to them to constrain placement. Block RAM placement locations differ from the convention used for naming CLB locations, allowing LOC properties to transfer easily from array to array.

The LOC properties use the following form:

LOC = RAMB16_X#Y#

The RAMB16_X0Y0 is the bottom-left block RAM location on the device.

Usage

Read Operation

The read operation uses one clock edge. The read address is registered on the read port, and the stored data is loaded into the output latches after the RAM access interval passes.

Write Operation

A write operation is a single clock-edge operation. The write address is registered on the write port, and the data input is stored in memory.

Operating Modes

There are three options for the behavior of the data output during a write operation on its port. The "read during write" mode offers the flexibility of using the data output bus during a write operation on the same port. Output behavior is determined by the configuration. This choice increases the efficiency of block RAM memory at each clock cycle and allows designs that use maximum bandwidth.

Three different modes are used to determine data available on the output latches after a write clock edge.

WRITE_FIRST or Transparent Mode (Default)

In WRITE_FIRST mode, the input data is simultaneously written into memory and stored in the data output (transparent write).

READ_FIRST or Read-Before-Write Mode

In READ_FIRST mode, data previously stored at the write address appears on the output latches, while the input data is being stored in memory (read before write).

NO_CHANGE Mode

In NO_CHANGE mode, the output latches remain unchanged during a write operation.

Mode selection is set by configuration. One of these three modes is set individually for each port by an attribute. The default mode is WRITE_FIRST.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DOA_REG	INTEGER	0 or 1	0	Optional output registers on A port .
DOB_REG	INTEGER	0 or 1	0	Optional output registers on B port.
INIT_00 to INIT_39	256-Bit Hexadecimal	256-Bit Hexadecimal	256'h0000000000000000000000000000000000000000000000000000000000000000	To change the initial contents of the RAM to anything other than all zero's.
INIT_0A to INIT_0F	256-Bit Hexadecimal	256-Bit Hexadecimal	256'h0000000000000000000000000000000000000000000000000000000000000000	To change the initial contents of the RAM to anything other than all zero's.
INIT_1A to INIT_1F	256-Bit Hexadecimal	256-Bit Hexadecimal	256'h0000000000000000000000000000000000000000000000000000000000000000	To change the initial contents of the RAM to anything other than all zero's.
INIT_2A to INIT_2F	256-Bit Hexadecimal	256-Bit Hexadecimal	256'h0000000000000000000000000000000000000000000000000000000000000000	To change the initial contents of the RAM to anything other than all zero's.
INIT_3A to INIT_3F	256-Bit Hexadecimal	256-Bit Hexadecimal	256'h0000000000000000000000000000000000000000000000000000000000000000	To change the initial contents of the RAM to anything other than all zero's.
INIT_A	36-Bit Hexadecimal	36-Bit Hexadecimal	36'h0	Initial values on A output port.
INIT_B	36-Bit Hexadecimal	36-Bit Hexadecimal	36'h0	Initial values on B output port.
INITP_00 to INITP_07	256-Bit Hexadecimal	256-Bit Hexadecimal	256'h0000000000000000000000000000000000000000000000000000000000000000	Applied for the parity bits.
INVERT_CLK_DOA_REG	BOOLEAN	FALSE, TRUE	FALSE	Invert clock on A port output registers.
INVERT_CLK_DOB_REG	BOOLEAN	FALSE, TRUE	FALSE	Invert clock on A port output registers.
RAM_EXTENSION_A	STRING	"LOWER", "NONE" or "UPPER"	"NONE"	Allowed value when cascaded.
RAM_EXTENSION_B	STRING	"LOWER", "NONE" or "UPPER"	"NONE"	Allowed value when cascaded.
READ_WIDTH_A	INTEGER	0, 1, 2, 4, 9, 18 or 36	0	Set/Reset for the allowed value.
READ_WIDTH_B	INTEGER	0, 1, 2, 4, 9, 18 or 36	0	Set/Reset for the allowed value.
SIM_COLLISION_CHECK	STRING	"ALL", "NONE", "WARNING_ONLY" or "GENERATE_X_ONLY"	"ALL"	Collision check enable for the allowed value.
SRVAL_A	36-Bit Hexadecimal	36-Bit Hexadecimal	36'h0	Use to set/reset value for A port output.
SRVAL_B	36-Bit Hexadecimal	36-Bit Hexadecimal	36'h0	Use to set/reset value for B port output.
WRITE_MODE_A	STRING	"WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"	"WRITE_FIRST"	Configures port A (Sm) of a dual port RAMB16 to support one of three write modes.
WRITE_MODE_B	STRING	"WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"	"WRITE_FIRST"	Configures port B (Sn) of a dual-port RAMB16 to support one of three write modes.

Attribute	Type	Allowed Values	Default	Description
WRITE_WIDTH_A	INTEGER	0, 1, 2, 4, 9, 18 or 36	0	Set/Reset for the allowed value.
WRITE_WIDTH_B	INTEGER	0, 1, 2, 4, 9, 18 or 36	0	Set/Reset for the allowed value.

VHDL Instantiation Template

```
--    RAMB16     : 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  : (RAMB16_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 will be used
--               : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- RAMB16: Virtex-4 16k+2k Parity Paramatizable BlockRAM
   -- Xilinx  HDL Libraries Guide Version 8.1i

   RAMB16_inst : RAMB16
   generic map (
      DOA_REG => 0, -- Optional output registers on the A port (0 or 1)
      DOB_REG => 0, -- Optional output registers on the B port (0 or 1)
      INIT_A => X"000000000", --  Initial values on A output port
      INIT_B => X"000000000", --  Initial values on B output port
      INVERT_CLK_DOA_REG => FALSE, -- Invert clock on A port output registers (TRUE or FALSE)
      INVERT_CLK_DOB_REG => FALSE, -- Invert clock on B port output registers (TRUE or FALSE)
      RAM_EXTENSION_A => "NONE", -- "UPPER", "LOWER" or "NONE" when cascaded
      RAM_EXTENSION_B => "NONE", -- "UPPER", "LOWER" or "NONE" when cascaded
      READ_WIDTH_A => 0, -- Valid values are 1,2,4,9,18 or 36
      READ_WIDTH_B => 0, -- Valid values are 1,2,4,9,18 or 36
      SIM_COLLISION_CHECK => "ALL", -- Collision check enable "ALL", "WARNING_ONLY", "GENERATE_X_ONLY"
                                    -- or "NONE"
      SRVAL_A => X"000000000", --  Port A ouput value upon SSR assertion
      SRVAL_B => X"000000000", --  Port B ouput value upon SSR assertion
      WRITE_MODE_A => "WRITE_FIRST", --  WRITE_FIRST, READ_FIRST or NO_CHANGE
      WRITE_MODE_B => "WRITE_FIRST", --  WRITE_FIRST, READ_FIRST or NO_CHANGE
      WRITE_WIDTH_A => 0, -- Valid values are 1,2,4,9,18 or 36
      WRITE_WIDTH_B => 0, -- Valid values are 1,2,4,9,18 or 36
      INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
      INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
   INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000")
port map (
   CASCADEOUTA => CASCADEOUTA, -- 1-bit cascade output
   CASCADEOUTB => CASCADEOUTB, -- 1-bit cascade output
   DOA => DOA,       -- 32-bit A port Data Output
   DOB => DOB,       -- 32-bit B port Data Output
   DOPA => DOPA,     -- 4-bit  A port Parity Output
   DOPB => DOPB,     -- 4-bit  B port Parity Output
   ADDRA => ADDRA,   -- 15-bit A port Address Input
   ADDRB => ADDRB,   -- 15-bit B port Address Input
   CASCADEINA => CASCADEINA, -- 1-bit cascade A input
   CASCADEINB => CASCADEINB, -- 1-bit cascade B input
   CLKA => CLKA,     -- Port A Clock
   CLKB => CLKB,     -- Port B Clock
   DIA => DIA,       -- 32-bit A port Data Input
   DIB => DIB,       -- 32-bit B port Data Input
   DIPA => DIPA,     -- 4-bit  A port parity Input
   DIPB => DIPB,     -- 4-bit  B port parity Input
   ENA => ENA,       -- 1-bit  A port Enable Input
   ENB => ENB,       -- 1-bit  B port Enable Input
   REGCEA => REGCEA, -- 1-bit A port register enable input
   REGCEB => REGCEB, -- 1-bit B port register enable input
   SSRA => SSRA,     -- 1-bit  A port Synchronous Set/Reset Input
   SSRB => SSRB,     -- 1-bit  B port Synchronous Set/Reset Input
   WEA => WEA,       -- 4-bit  A port Write Enable Input
   WEB => WEB        -- 4-bit  B port Write Enable Input
);

-- End of RAMB16_inst instantiation
```

Verilog Instantiation Code

```
//     RAMB16    : 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  : (RAMB_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---->

   // RAMB16: Virtex-4 16k+2k Parity Paramatizable BlockRAM
   // Xilinx HDL Libraries Guide Version 8.1i

   RAMB16 #(
      .DOA_REG(0),  // Optional output registers on A port (0 or 1)
      .DOB_REG(0),  // Optional output registers on B port (0 or 1)
      .INIT_A(36'h000000000),  // Initial values on A output port
      .INIT_B(36'h000000000),  // Initial values on B output port
      .INVERT_CLK_DOA_REG("FALSE"),  // Invert clock on A port output registers ("TRUE" or "FALSE")
      .INVERT_CLK_DOB_REG("FALSE"),  // Invert clock on A port output registers ("TRUE" or "FALSE")
      .RAM_EXTENSION_A("NONE"),  // "UPPER", "LOWER" or "NONE" when cascaded
      .RAM_EXTENSION_B("NONE"),  // "UPPER", "LOWER" or "NONE" when cascaded
      .READ_WIDTH_A(0),  // Valid values are 1, 2, 4, 9, 18, or 36
      .READ_WIDTH_B(0),  // Valid values are 1, 2, 4, 9, 18, or 36
      .SIM_COLLISION_CHECK("ALL"),  // Collision check enable "ALL", "WARNING_ONLY",
                                    //   "GENERATE_X_ONLY" or "NONE"
      .SRVAL_A(36'h000000000), // Set/Reset value for A port output
      .SRVAL_B(36'h000000000),  // Set/Reset value for B port output
      .WRITE_MODE_A("WRITE_FIRST"),  // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
      .WRITE_MODE_B("WRITE_FIRST"),  // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
      .WRITE_WIDTH_A(2),  // Valid values are 1, 2, 4, 9, 18, or 36
      .WRITE_WIDTH_B(0),  // Valid values are 1, 2, 4, 9, 18, or 36

      // The following INIT_xx declarations specify the initial contents of the RAM
      .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_13(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_14(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_15(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_16(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_17(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_18(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_19(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_1A(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_1B(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_1C(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_1D(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_1E(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_1F(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_20(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_21(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_22(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_23(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_24(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_25(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_26(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_27(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_28(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_29(256'h0000000000000000000000000000000000000000000000000000000000000000),
      .INIT_2A(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

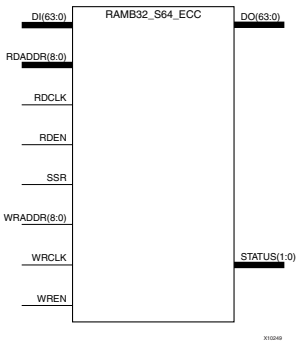
```
    .INIT_2B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_30(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_31(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_32(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_33(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_34(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_35(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_36(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_37(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_38(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_39(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3F(256'h0000000000000000000000000000000000000000000000000000000000000000),

    // The next set of INITP_xx are for the parity bits
    .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000)
) RAMB16_inst (
    .CASCADEOUTA(CASCADEOUTA),  // 1-bit cascade output
    .CASCADEOUTB(CASCADEOUTB),  // 1-bit cascade output
    .DOA(DOA),       // 32-bit A port data output
    .DOB(DOB),       // 32-bit B port data output
    .DOPA(DOPA),     // 4-bit A port parity data output
    .DOPB(DOPB),     // 4-bit B port parity data output
    .ADDRA(ADDRA),   // 15-bit A port address input
    .ADDRB(ADDRB),   // 15-bit B port address input
    .CASCADEINA(CASCADEINA), // 1-bit cascade A input
    .CASCADEINB(CASCADEINB), // 1-bit cascade B input
    .CLKA(CLKA),     // 1-bit A port clock input
    .CLKB(CLKB),     // 1-bit B port clock input
    .DIA(DIA),       // 32-bit A port data input
    .DIB(DIB),       // 32-bit B port data input
    .DIPA(DIPA),     // 4-bit A port parity data input
    .DIPB(DIPB),     // 4-bit B port parity data input
    .ENA(ENA),       // 1-bit A port enable input
    .ENB(ENB),       // 1-bit B port enable input
    .REGCEA(REGCEA), // 1-bit A port register enable input
    .REGCEB(REGCEB), // 1-bit B port register enable input
    .SSRA(SSRA),     // 1-bit A port set/reset input
    .SSRB(SSRB),     // 1-bit B port set/reset input
    .WEA(WEA),       // 4-bit A port write enable input
    .WEB(WEB)        // 4-bit B port write enable input
);

// End of RAMB16_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

RAMB32_S64_ECC

Primitive: 512 Deep by 64-Bit Wide Synchronous, Two-Port, Block RAM with Built-In Error Correction



Two vertically adjacent block RAMs can be configured as a single 512 x 64 RAM with built in Hamming error correction, using the extra eight bits in the 72-bit wide RAM. The operation is transparent to the user. The eight protection bits are generated during each write operation, and are used during each read operation to correct any single error, or to detect (but not correct) any double error. Two status outputs indicate the three possible read results: No error, single error corrected, double error detected. The read operation does not correct the error in the memory array, it only presents corrected data on DOUT.

This error correction code (ECC) configuration option is available with any block RAM pair, but cannot use the one block RAM immediately above or below the Virtex-4 PowerPC™ blocks.

Port Names and Descriptions

Port Name	Direction	Signal Description
DIN<63:0>	Input	Data input bus
WRADDR<8:0>	Input	Write address bus
RDADDR<8:0>	Input	Read address bus
WREN	Input	Write enable. When WREN = 1, data will be written into memory. When WREN = 0, write is disabled.
RDEN	Input	Read enable. When RDEN = 1, data will be read from memory. When RDEN = 0, read is disabled
SSR	Input	Set/Reset output registers (not the memory content)
WRCLK	Input	Clock for write operations
RDCLK	Input	Clock for read operations
DOUT<63:0>	Output	Data output bus
STATUS<1:0>(1)	Output	Error status bus

Note: Hamming code implemented in the block RAM ECC logic detects one of three conditions: no detectable error, single-bit error detected and corrected on DOUT (but not corrected in the memory), and double-bit error detected without correction. The result of STATUS<1:0> indicates these three conditions.

Status Bit Truth Table

STATUS[1:0]	Description
00	No bit error.
01	Single-bit error. The block RAM ECC macro detects and automatically corrects a single-bit error.
10	Double-bit error. The block RAM ECC macro detects a double-bit error.
11	Indeterminate state. The Hamming code implemented in the block RAM ECC cannot generate a predictable status if STATUS<1:0> is equal to three. Designers must ensure that the data has at most double-bit errors for the STATUS<1:0> to generate the proper indicator.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DO_REG	INTEGER	0 or 1	0	Optional output registers on A port .
SIM_COLLISION_CHECK	STRING	“ALL”, "NONE", "WARNING_ONLY" or "GENERATE_X_ONLY”	"ALL”	Collision check enable for the allowed value.

VHDL Instantiation Template

```
-- RAMB32_S64_ECC  : 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 (RAMB32_S64_ECC_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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- RAMB32_S64_ECC: Virtex-4 512 x 64 Error Correction BlockRAM
   -- Xilinx HDL Libraries Guide Version 8.1i

   RAMB32_S64_ECC_inst: RAMB32_S64_ECC_inst (
      port map (
         DO => DO,          -- 64-bit output data
         STATUS => STATUS,  -- 2-bit status output
         DI => DI,          -- 64-bit data input
         RDADDR => RDADDR,  -- 9-bit data address input
         RDCLK => RDCLK,    -- 1-bit read clock input
         RDEN => RDEN,      -- 1-bit read enable input
         SSR => SSR,        -- 1-bit synchronous reset
         WRADDR =>WRADDR,   -- 9-bit write address input
         WRCLK => WRCLK,    -- 1-bit write clock input
         WREN => WREN       -- 1-bit write enable input
      );

   -- End of RAMB32_S64_ECC_inst instantiation
```

Verilog Instantiation Code

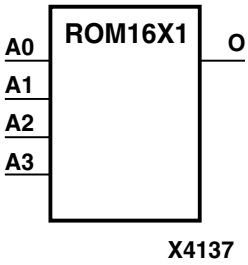
```
// RAMB32_S64_ECC: In order to incorporate this function into the design,
//    Verilog    : the following instance declaration needs to be placed
//   instance    : in the body of the design code.  The instance name
//  declaration  : (RAMB32_S64_ECC_inst) and/or the port declarations within the
//     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---->

   // RAMB32_S64_ECC: Virtex-4 512 x 64 Error Correction BlockRAM
   // Xilinx HDL Libraries Guide Version 8.1i

   RAMB32_S64_ECC RAMB32_S64_ECC_inst (
      .DO(DO),          // 64-bit output data
      .STATUS(STATUS), // 2-bit status output
      .DI(DI),          // 64-bit data input
      .RDADDR(RDADDR), // 9-bit data address input
      .RDCLK(RDCLK),    // 1-bit read clock input
      .RDEN(RDEN),      // 1-bit read enable input
      .SSR(SSR),        // 1-bit synchronous reset
      .WRADDR(WRADDR), // 9-bit write address input
      .WRCLK(WRCLK),    // 1-bit write clock input
      .WREN(WREN)       // 1-bit write enable input
   );

   // End of RAMB32_S64_ECC_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

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 to a known value during configuration with the INIT=value parameter. 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	16-Bit Hexadecimal	16-Bit Hexadecimal	16'h0000	Specifies the contents after configuration.

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 will be used
--             : for simulation.

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

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

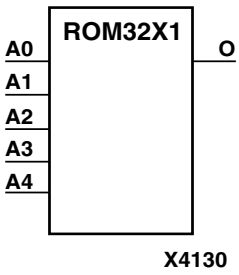
```
//   ROM16X1   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (ROM16X1_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

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 to a known value during configuration with the INIT=value parameter. The value consists of eight hexadecimal digits that are written into the ROM from the most-significant digit A=1FH to the least-significant digit A=00h. For example, the INIT=10A78F39 parameter produces the data stream:

```
0001 0000 1010 0111 1000 1111 0011 1001
```

An error occurs if the INIT=*value* is not specified.

Usage

This design element should be instantiated rather than inferred.

Available Attributes.

Attribute	Type	Allowed Values	Default	Description
INIT	32-Bit Hexadecimal	32-Bit Hexadecimal	32'h00000000	Specifies the contents after configuration.

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 will be used
--             : for simulation.

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

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

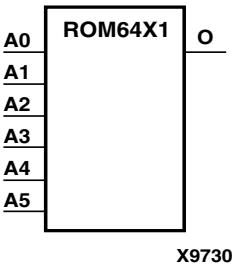
```
//   ROM32X1   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (ROM32X1_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 User Guide*.

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 to a known value during configuration with the INIT=value parameter. The value consists of 16 hexadecimal digits that are written into the ROM from the most-significant digit A=Fh to the least-significant digit A=0h.

An error occurs if the INIT=*value* is not specified.

Usage

This design element should be instantiated rather than inferred.

Available Attributes.

Attribute	Type	Allowed Values	Default	Description
INIT	64-Bit Hexadecimal	64-Bit Hexadecimal	64'h0000000000000000	Specifies the contents after configuration.

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 will be used
--             : for simulation.

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

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
   --          Virtex-II/II-Pro, Spartan-3/3E
   -- 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

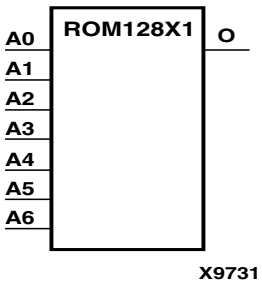
```
//   ROM64X1   : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (ROM64X1_inst) and/or the port declarations within the
//    code     : 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
   //          Virtex-II/II-Pro/4, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

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 to a known value during configuration with the INIT=value parameter. The value consists of 32 hexadecimal digits that are written into the ROM from the most-significant digit A=Fh to the least-significant digit A=0h.

An error occurs if the INIT=*value* is not specified.

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	128-Bit Hexadecimal	128-Bit Hexadecimal	128'h0000000000000000000000000000000	Specifies the contents after configuration.

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 will be used
--             : for simulation.

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

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
   --           Virtex-II/II-Pro, Spartan-3/3E
   -- 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

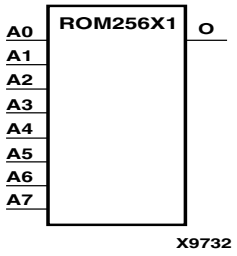
```
//   ROM128X1   : In order to incorporate this function into the design,
//   Verilog    : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration : (ROM128X1_inst) and/or the port declarations within the
//    code      : 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
   //           Virtex-II/II-Pro/4, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

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 to a known value during configuration with the INIT=value parameter. The value consists of 64 hexadecimal digits that are written into the ROM from the most-significant digit A=Fh to the least-significant digit A=0h.

An error occurs if the INIT=*value* is not specified.

Usage

This design element should be instantiated rather than inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	256-Bit Hexadecimal	256-Bit Hexadecimal	256'h0000000000000000000000000000000000000000000000000000000000000000	Specifies the contents after configuration.

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 will be used
--              : for simulation.

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

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
   --           Virtex-II/II-Pro, Spartan-3/3E
   -- 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

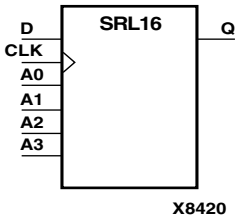
```
//   ROM256X1   : In order to incorporate this function into the design,
//   Verilog    : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration : (ROM256X1_inst) and/or the port declarations within the
//    code      : 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
   //           Virtex-II/II-Pro, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

SRL16

Primitive: 16-Bit Shift Register Look-Up Table (LUT)



SRL16 is a shift register look-up table (LUT). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may 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, it 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. During subsequent Low-to-High clock transitions data shifts to the next highest bit position while new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

Static Length Mode

To get a fixed-length shift register, drive the A3 through A0 inputs with static values. The length of the shift register can vary from 1 bit to 16 bits, as determined by the following formula:

Length = (8 x A3) +(4 x A2) + (2 x A1) + A0 +1

If A3, A2, A1, and A0 are all zeros (0000), the shift register is one bit long. If they are all ones (1111), it is 16 bits long.

Dynamic Length Mode

The length of the shift register can be changed dynamically by changing the values driving the A3 through A0 inputs. For example, if A2, A1, and A0 are all ones (111) and A3 toggles between a one (1) and a zero (0), the length of the shift register changes from 16 bits to 8 bits.

Internally, the length of the shift register is always 16 bits and the input lines A3 through A0 select which of the 16 bits reach the output.

Inputs			Output
Am	**CLK**	**D**	**Q**
Am	X	X	Q(Am)
Am	↑	D	Q(Am - 1)

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 Hexadecimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of Q output after configuration

VHDL Instantiation Templates

```
--    SRL16    : 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 (SRL16_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 will be used
--             : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- SRL16: 16-bit shift register LUT operating on posedge of clock
   --        All FPGAs
   -- Xilinx  HDL Libraries Guide Version 8.1i

   SRL16_inst : SRL16
   generic map (
      INIT => X"0000")
   port map (
      Q => Q,       -- SRL data output
      A0 => A0,     -- Select[0] input
      A1 => A1,     -- Select[1] input
      A2 => A2,     -- Select[2] input
      A3 => A3,     -- Select[3] input
      CLK => CLK,   -- Clock input
      D => D        -- SRL data input
   );

   -- End of SRL16_inst instantiation
```

Verilog Instantiation Code

```
//   SRL16     : 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 : (SRL16_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---->

   // SRL16: 16-bit shift register LUT operating on posedge of clock
   //        All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   SRL16 #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRL16_inst (
      .Q(Q),       // SRL data output
      .A0(A0),     // Select[0] input
      .A1(A1),     // Select[1] input
      .A2(A2),     // Select[2] input
      .A3(A3),     // Select[3] input
      .CLK(CLK),   // Clock input
```

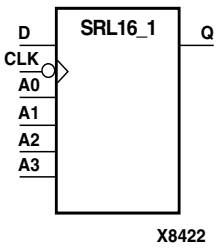
```
    .D(D)         // SRL data input
);

// End of SRL16_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

SRL16_1

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Negative-Edge Clock



SRL16_1 is a shift register look-up table (LUT). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may be dynamically adjusted. See "Static Length Mode" and "Dynamic Length Mode" in "SRL16".

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions data shifts to the next highest bit position as new data is loaded. The data appears on theQ output when the shift register length determined by the address inputs is reached.

Inputs			Output
Am	**CLK**	**D**	**Q**
Am	X	X	Q(Am)
Am	↓	D	Q(Am - 1)

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 Hexadecimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of Q output after configuration

VHDL Instantiation Template

```
--   SRL16_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 (SRL16_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 will be used
--             : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

```
   -- SRL16_1: 16-bit shift register LUT operating on negedge of clock
   --          All FPGAs
   -- Xilinx  HDL Libraries Guide Version 8.1i

   SRL16_1_inst : SRL16_1
   generic map (
      INIT => X"0000")
   port map (
      Q => Q,        -- SRL data output
      A0 => A0,      -- Select[0] input
      A1 => A1,      -- Select[1] input
      A2 => A2,      -- Select[2] input
      A3 => A3,      -- Select[3] input
      CLK => CLK,    -- Clock input
      D => D         -- SRL data input
   );

   -- End of SRL16_1_inst instantiation
```

Verilog Instantiation Code

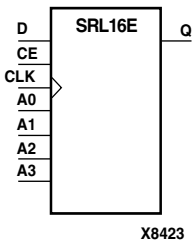
```
//  SRL16_1    : In order to incorporate this function into the design,
//  Verilog    : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (SRL16_1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

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

   // SRL16_1: 16-bit shift register LUT operating on negedge of clock
   //          All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   SRL16_1 #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRL16_1_inst (
      .Q(Q),       // SRL data output
      .A0(A0),     // Select[0] input
      .A1(A1),     // Select[1] input
      .A2(A2),     // Select[2] input
      .A3(A3),     // Select[3] input
      .CLK(CLK),   // Clock input
      .D(D)        // SRL data input
   );

   // End of SRL16_1_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

SRL16E

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Clock Enable



SRL16E is a shift register look-up table (LUT). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or dynamically adjusted. See "Static Length Mode" and "Dynamic Length Mode" in "SRL16".

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the Low-to-High clock (CLK) transition. During subsequent Low-to-High clock transitions, when CE is High, data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

When CE is Low, the register ignores clock transitions.

Inputs				Output
Am	**CE**	**CLK**	**D**	**Q**
Am	0	X	X	Q(Am)
Am	1	↑	D	Q(Am - 1)

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 Hexadecimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of content and output of shift register after configuration

VHDL Instantiation Template

```
--   SRL16E    : 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 will be used
--             : for simulation.

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

Library UNISIM;
```

```
use UNISIM.vcomponents.all;

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

   -- SRL16E: 16-bit shift register LUT with clock enable operating on posedge of clock
   --         All FPGAs
   -- Xilinx HDL Libraries Guide Version 8.1i

   SRL16E_inst : SRL16E
   generic map (
      INIT => X"0000")
   port map (
      Q => Q,       -- SRL data output
      A0 => A0,     -- Select[0] input
      A1 => A1,     -- Select[1] input
      A2 => A2,     -- Select[2] input
      A3 => A3,     -- Select[3] input
      CE => CE,     -- Clock enable input
      CLK => CLK,   -- Clock input
      D => D        -- SRL data input
   );

   -- End of SRL16E_inst instantiation
```

Verilog Instantiation Code

```
//  SRL16E     : 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---->

   // SRL16E: 16-bit shift register LUT with clock enable operating on posedge of clock
   //         All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   SRL16E #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRL16E_inst (
      .Q(Q),       // SRL data output
      .A0(A0),     // Select[0] input
      .A1(A1),     // Select[1] input
      .A2(A2),     // Select[2] input
      .A3(A3),     // Select[3] input
      .CE(CE),     // Clock enable input
      .CLK(CLK),   // Clock input
      .D(D)        // SRL data input
```

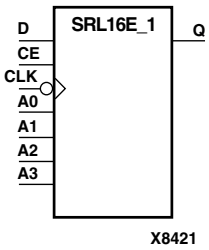
```
);
// End of SRL16E_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

SRL16E_1

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Negative-Edge Clock and Clock Enable



SRL16E_1 is a shift register look-up table (LUT) with clock enable (CE). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or dynamically adjusted. See "Static Length Mode" and "Dynamic Length Mode" in the "SRL16".

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions, when CE is High, data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

When CE is Low, the register ignores clock transitions.

Inputs				Output
Am	**CE**	**CLK**	**D**	**Q**
Am	0	X	X	Q(Am)
Am	1	↓	D	Q(Am - 1)

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 Hexadecimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of content and output of shift register after configuration

VHDL Instantiation Template

```
--  SRL16E_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 (SRL16E_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 will be used
--             : for simulation.

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

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

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

   -- SRL16E_1: 16-bit shift register LUT with clock enable operating on negedge of clock
   --           All FPGAs
   -- Xilinx  HDL Libraries Guide Version 8.1i

   SRL16E_1_inst : SRL16E_1
   generic map (
      INIT => X"0000")
   port map (
      Q => Q,       -- SRL data output
      A0 => A0,     -- Select[0] input
      A1 => A1,     -- Select[1] input
      A2 => A2,     -- Select[2] input
      A3 => A3,     -- Select[3] input
      CE => CE,     -- Clock enable input
      CLK => CLK,   -- Clock input
      D => D        -- SRL data input
   );

   -- End of SRL16E_1_inst instantiation
```

Verilog Instantiation Code

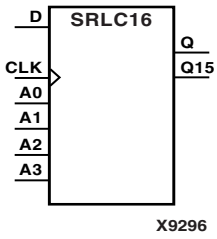
```
//  SRL16E_1   : In order to incorporate this function into the design,
//  Verilog    : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (SRL16E_1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

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

   // SRL16E_1: 16-bit shift register LUT with clock enable operating on negedge of clock
   //           All FPGAs
   // Xilinx HDL Libraries Guide Version 8.1i

   SRL16E_1 #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRL16E_1_inst (
      .Q(Q),       // SRL data output
      .A0(A0),     // Select[0] input
      .A1(A1),     // Select[1] input
      .A2(A2),     // Select[2] input
      .A3(A3),     // Select[3] input
      .CE(CE),     // Clock enable input
      .CLK(CLK),   // Clock input
      .D(D)        // SRL data input
   );

   // End of SRL16E_1_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

SRLC16

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Carry



SRLC16 is a shift register look-up table (LUT) with Carry. The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length, or it may 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, it 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. During subsequent Low-to-High clock transitions data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

For information about the static length mode, see "Static Length Mode" in "SRL16".

For information about the dynamic length mode, see "Dynamic Length Mode" in "SRL16".

Inputs			Output
Am	**CLK**	**D**	**Q**
Am	X	X	Q(Am)
Am	↑	D	Q(Am - 1)

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 Hexadecimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of content and output of shift register after configuration

VHDL Instantiation Template

```
--   SRLC16    : 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 (SRLC16_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 will be used
--             : for simulation.
```

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- SRLC16: 16-bit cascadable shift register LUT operating on posedge of clock
   --         Virtex-II/II-Pro, Spartan-3/3E
   -- Xilinx  HDL Libraries Guide Version 8.1i

   SRLC16_inst : SRLC16
   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
      CLK => CLK,   -- Clock input
      D => D        -- SRL data input
   );

   -- End of SRLC16_inst instantiation
```

Verilog Instantiation Code

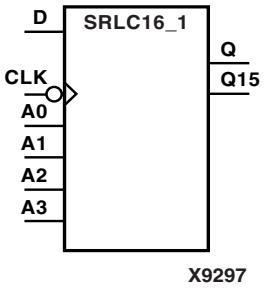
```
//   SRLC16    : In order to incorporate this function into the design,
//  Verilog    : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (SRLC16_inst) and/or the port declarations within the
//    code     : 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---->

   // SRLC16: 16-bit cascadable shift register LUT operating on posedge of clock
   //         Virtex-II/II-Pro/4, Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   SRLC16 #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRLC16_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
      .CLK(CLK),   // Clock input
      .D(D)        // SRL data input
   );

   // End of SRLC16_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

SRLC16_1

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Carry and Negative-Edge Clock



SRLC16_1 is a shift register look-up table (LUT) with carry and a negative-edge clock. The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be a fixed-length, static length, or it may be dynamically adjustable. See "Static Length Mode" and "Dynamic Length Mode" in "SRL16".

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

Inputs			Output	
Am	**CLK**	**D**	**Q**	**Q15**
Am	X	X	Q(Am)	No Change
Am	↓	D	Q(Am - 1)	Q14

m= 0, 1, 2, 3

Usage

This design element can also be inferred.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	16-Bit Hexadecimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of content and output of shift register after configuration

VHDL Instantiation Template

```
--  SRLC16_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 (SRLC16_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 will be used
--             : for simulation.

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

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

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

   -- SRLC16_1: 16-bit cascadable shift register LUT operating on negedge of clock
   --           Virtex-II/II-Pro, Spartan-3/3E
   -- Xilinx  HDL Libraries Guide Version 8.1i

   SRLC16_1_inst : SRLC16_1
   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
      CLK => CLK,   -- Clock input
      D => D        -- SRL data input
   );

   -- End of SRLC16_1_inst instantiation
```

Verilog Instantiation Code

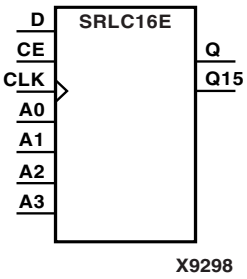
```
//  SRLC16_1   : In order to incorporate this function into the design,
//  Verilog    : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (SRLC16_1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

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

   // SRLC16_1: 16-bit cascadable shift register LUT operating on negedge of clock
   //           Virtex-II/II-Pro/4, Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   SRLC16_1 #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRLC16_1_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
      .CLK(CLK),   // Clock input
      .D(D)        // SRL data input
   );

   // End of SRLC16_1_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

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 may be of a fixed, static length or it may 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, it 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 shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

For information about the static length mode, see "Static Length Mode" in "SRL16".

For information about the dynamic length mode, see "Dynamic Length Mode" in "SRL16".

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 Hexadecimal	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 will be used
--             : for simulation.

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

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
   --          Virtex-II/II-Pro, Spartan-3/3E
   -- 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

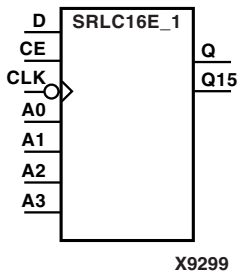
```
//  SRLC16E     : In order to incorporate this function into the design,
//  Verilog     : the following instance declaration needs to be placed
//  instance    : in the body of the design code.  The instance name
// declaration : (SRL16E_inst) and/or the port declarations within the
//    code      : 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
   //          Virtex-II/II-Pro/4, Spartan-3/3E
   // 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 *Virtex-4 User Guide*.

SRLC16E_1

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Carry, Negative-Edge Clock, and Clock Enable



SRLC16E_1 is a shift register look-up table (LUT) with carry, clock enable, and negative-edge clock. The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may be dynamically adjusted. See "SRLC16" and "Dynamic Length Mode" in "SRL16".

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions data shifts to the next highest bit position as new data is loaded when CE is HIgh. 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	**CE**	**CLK**	**D**	**Q**	**Q15**
Am	0	X	X	Q(Am)	No Change
Am	1	X	X	Q(Am)	No Change
Am	1	↓	D	Q(Am -1)	Q14

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 Hexadecimal	16-Bit Hexadecimal	16'h0000	Sets the initial value of content and output of shift register after configuration

VHDL Instantiation Template

```
--  SRLC16E_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 (SRLC16E_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 will be used
--             : for simulation.
```

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- SRLC16E_1: 16-bit shift register LUT with clock enable operating on negedge of clock
   --            Virtex-II/II-Pro, Spartan-3/3E
   -- Xilinx  HDL Libraries Guide Version 8.1i

   SRLC16E_1_inst : SRLC16E_1
   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_1_inst instantiation
```

Verilog Instantiation Code

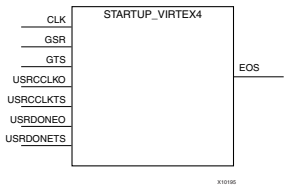
```
//  SRLC16E_1  : In order to incorporate this function into the design,
//  Verilog    : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (SRLC16E_1_inst) and/or the port declarations within the
//    code     : parenthesis maybe changed to properly reference and
//             : connect this function to the design.  All inputs
//             : and outputs must be connected.

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

   // SRLC16E_1: 16-bit shift register LUT with clock enable operating on negedge of clock
   //            Virtex-II/II-Pro/4, Spartan-3/3E
   // Xilinx HDL Libraries Guide Version 8.1i

   SRLC16E_1 #(
      .INIT(16'h0000) // Initial Value of Shift Register
   ) SRLC16E_1_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_1_inst instantiation
```

For More Information

Consult the *Virtex-4 User Guide*.

STARTUP_VIRTEX4

Primitive: Virtex-4 User Interface to Configuration Clock, Global Reset, Global 3-State Controls, and Other Configuration Signals



The STARTUP_VIRTEX4 primitive is used for Global Set/Reset, global 3-state control, and the user configuration clock. The Global Set/Reset (GSR) input, when High, sets or resets all flip-flops, all latches, and every block RAM output register in the device, depending on the initialization state (INIT=1 or 0) of the component.

Note: Block RAM content, LUT RAMs, the Digital Clock Manager (DCM), and shift register LUTs (SRL16, SRL16_1, SRL16E, SRL16E_1, SRLC16, SRLC16_1, SRLC16E, and SRLC16E_1) are not set/reset.

Following configuration, the global 3-state control (GTS), when High—and BSCAN is not enabled and executing an EXTEST instruction—forces all the IOB outputs into high impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

Port List and Definition

Name	Type	Width	Function
EOS	Output	1	EOS signal
CLK	Input	1	Clock input
GTS	Input	1	Global 3-state control
GSR	Input	1	Global Set/Reset
USRCCLKO	Input	1	
USRCCLKTS	Input	1	
USRDONEO	Input	1	
USRDONETS	Input	1	

Usage

Including the STARTUP_VIRTEX4 primitive in a design is optional. You must include the primitive under the following conditions.

To exert external control over global set/reset, connect the GSR pin to a top level port and an IBUF, as shown below.

To exert external control over global 3-state, connect the GTS pin to a top level port and IBUF, as shown below.

To synchronize startup to a user clock, connect the user clock signal to the CLK input, as shown below. Furthermore, "user clock" must be selected in the BitGen program.

You can use location constraints to specify the pin from which GSR or GTS (or both) is accessed.

VHDLInstantiation Template

```
-- STARTUP_VIRTEX4 : 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_VIRTEX4_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 will be used
--                 : for simulation.

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

Library UNISIM;
use UNISIM.vcomponents.all;

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

   -- STARTUP_VIRTEX4: Startup primitive for GSR, GTS or startup sequence
   --                  control. Virtex-4
   -- Xilinx  HDL Libraries Guide Version 8.1i
   STARTUP_VIRTEX4_inst : STARTUP_VIRTEX4
   port map (
      EOS => EOS,       -- End of Startup 1-bit output
      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)
      USRCCLKO => USRCCLKO,    -- USRCCLKO 1-bit input
      USRCCLKTS => USRCCLKTS, -- USRCCLKTS 1-bit input
      USRDONEO => USRDONEO,    -- USRDONEO 1-bit input
      USRDONETS => USRDONETS  -- USRDONETS 1-bit input
   );

   -- End of STARTUP_VIRTEX4_inst instantiation
```

Verilog Instantiation Code

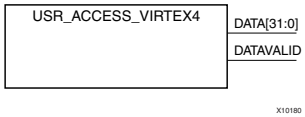
```
//  STARTUP_VIRTEX4 : In order to incorporate this function into the design,
//     Verilog      : the following instance declaration needs to be placed
//    instance      : in the body of the design code.  The instance name
//   declaration    : (STARTUP_VIRTEX4_inst) and/or the port declarations 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_VIRTEX4: Startup primitive for GSR, GTS or startup sequence
   //                  control. Virtex-4
   // Xilinx HDL Libraries Guide Version 8.1i

   STARTUP_VIRTEX4 STARTUP_VIRTEX4_inst (
      .EOS(EOS), // End Of Startup 1-bit output
      .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)
      .USRCCLKO(USRCCLKO), // USERCLK0 1-bit input
      .USRCCLKTS(USRCCLKTS), // USERCLKTS 1-bit input
      .USRDONEO(USRDONEO), // USRDONE0 1-bit input
      .USRDONETS(USRDONETS) // USRDONETS 1-bit input
   );

   // End of STARTUP_VIRTEX4_inst instantiation
```

For More Information

Consult the *Virtex-4 Configuration Guide*.

USR_ACCESS_VIRTEX4

Primitive: 32-Bit Register with a 32-Bit DATA Bus and a DATAVALID Port



The User Access Register (USR_ACCESS_VIRTEX4) module is a 32-bit register that allows data from the bitstream to be directly accessible by the FPGA fabric. This module has two outputs: the 32-bit DATA bus and DATAVALID.

The configuration data source clock can be CCLK or TCK. The use model for this block is that it allows data from a bitstream data storage source (e.g., PROM) to be accessed by the fabric after the FPGA has been configured. To accomplish this the STARTUP_VIRTEX4 block should also be instantiated. The STARTUP_VIRTEX4 block has inputs that allow the user to take over the CCLK and DONE pins after the EOS (End-Of-Startup) signal has been asserted. These pins are USR_CCLK_O, USR_CCLK_TS, USR_DONE_O, and USR_DONE_TS. The bitgen option –g DONE_cycle: 7 should be used to prevent the DONE pin from going high since that would reset the PROM. The USR_CCLK_O pin should be connected to a controlled clock in the fabric. The PROM should contain a packet of data with the USR_ACCESS register as the target. After EOS has been asserted, the data packet can be loaded by clocking the USR_CCLK_O pin while keeping USR_CCLK_TS low (it can be tied low in this usage).

Alternatively, the USR_ACCESS register can be used to provide a single 32-bit constant value to the fabric as an alternative to using a BRAM or LUTRAM to hold the constant.

Name	Type	Width	Function
DATA	Output	32	32-bit data bus
DATAVALID	Output	1	Indicates whether the value at the DATA bus is valid or new

DATA – Output

DATA output port is the 32-bit register that allows the FPGA fabric to access data from bitstream data storage source.

DATAVALID – Output

DATAVALID port indicates whether the value in the DATA bus is new or valid. When this condition is true, this port is asserted HIGH for one cycle of the configuration data source clock.

USR_ACCESS_VIRTEX4 Usage

Whenever a new value accessed by USR_ACCESS_VIRTEX4 appeared in the DATA bus, the DATAVALID signal is asserted for one cycle of the configuration data source clock. There are many sources for the configuration data source clock. They can be either CCLK or TCK.

When using this module to access data from bitstream data storage source (e.g., PROM) to FPGA fabric after configuration, the STARTUP_VIRTEX4 block should also be instantiated. The STARTUP_VIRTEX4 module contains inputs that allow the designer to utilize the CCLK and DONE pins after the EOS (End-Of-Startup) signal have been asserted. These pins are USR_CCLK_O, USR_CCLK_TS, USR_DONE_O, and USR_DONE_TS. The USR_CCLK_O pin should be connected to a controlled

clock in the fabric. The data storage source should contain a packet of data with the USR_ACCESS_VIRTEX4 register as the target. After EOS has been asserted, the data packet can be loaded by clocking the USR_CCLK_O pin while keeping USR_CCLK_TS to logic Low. The USR_CCLK_TS can be tied to logic LOW when using this application.

In addition, when using this module, the bitgen option –g DONE_cycle: 7 should be used to prevent the HIGH assertion of DONE pin. Should the DONE pin be asserted HIGH, the PROM will be reset.

VHDL Instantiation Template

```
-- USR_ACCESS_VIRTEX4: 32-bit register that allows data from the
-- bitstream
-- to be directly accessible by the FPGA fabric.
-- Virtex-4
-- Xilinx HDL Libraries Guide version 8.1i

USR_ACCESS_VIRTEX4_inst : USR_ACCESS_VIRTEX4
port map (
DATA => DATA, -- 32-bit data output
DATAVALID => DATAVALID -- 1-bit data valid signal
);

// End of USR_ACCESS_VIRTEX4_inst instantiation
```

Verilog Template

```
// USR_ACCESS_VIRTEX4: 32-bit register that allows data from the
// bitstream
// to be directly accessible by the FPGA fabric.
// Virtex-4
// Xilinx HDL Libraries Guide version 8.1i

USR_ACCESS_VIRTEX4 USR_ACCESS_VIRTEX4_inst (
.DATA(DATA), // 32-bit data output
.DATAVALID(DATAVALID) // 1-bit data valid signal
);

// End of USR_ACCESS_VIRTEX4_inst instantiation
```

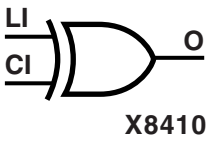
For More Information

Consult the *Virtex-4 Configuration Guide*.

XORCY

Primitive: XOR for Carry Logic with General Output

XORCY is a special XOR with general O output that generates 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 will be used
--             : for simulation.

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

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

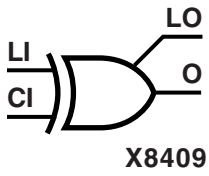
```
//    XORCY    : In order to incorporate this function into the design,
//   Verilog   : the following instance declaration needs to be placed
//  instance   : in the body of the design code.  The instance name
// declaration : (XORCY_inst) and/or the port declarations within the
//    code     : 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 *Virtex-4 Configuration Guide*.

XORCY_D

Primitive: XOR for Carry Logic with Dual Output



XORCY_D is a special XOR that generates 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.

See also "XORCY" and "XORCY_L."

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 will be used
--             : for simulation.

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

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
```

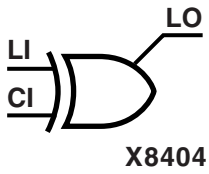
```
   .CI(CI), // Carry input signal
   .LI(LI)  // LUT4 input signal
);

// End of XORCY_D_inst instantiation.
```

For More Information

Consult the *Virtex-4 Configuration Guide*.

XORCY_L

Primitive: XOR for Carry Logic with Local Output



XORCY_L is a special XOR with local LO output that generates faster and smaller arithmetic functions.

Usage

The LO output connects to another output within the same CLB slice.

See also "XORCY" and "XORCY_D."

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 will be used
--             : for simulation.

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

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 *Virtex-4 Configuration Guide*.