LogiCORE IP High Speed SelectIO Wizard v1.0

Product Guide

Vivado Design Suite

PG188 April 2, 2014

XILINX
ALL PROGRAMMABLE™

Table of Contents

Appendix B: Debugging

Appendix C: Additional Resources and Legal Notices

Introduction

The LogiCORE™ IP High Speed SelectIO™ Wizard simplifies the integration of SelectIO technology into high-speed system designs for UltraScale™ architecture-based devices. This wizard creates a Verilog HDL file that instantiates and configures I/O and clocking logic such as RX_BITSLICE, TX_BITSLICE, BITSLICE_CONTROL and PLLE3 blocks present in PHY architecture. Additionally, this core provides pin planning for the selected interface and updates the RTL based on constraints.

Features

- Up to two interfaces with different configurations supported.
- Each interface provides RX, TX, and RX & TX Separate bus configurations with up to 46 bits per bank for single-ended signaling and 23 bits per bank for differential signaling.
- Serialization factor of four and eight are supported.
- Dynamic Phase Alignment (DPA) mode for the RX data capture scheme.
- Delay configuration for each interface.
- Bank selection and I/O planning to generate valid constraints.
- Use pin update in I/O planning to update the required connections among design blocks in RTL.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale™ Architecture
Supported User Interfaces	RIU
Resources	See Table 2-1
Provided with Core	
Design Files	RTL
Example Design	Verilog
Test Bench	Verilog
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver(2)	N/A
Tested Design Flows(3)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from //wiki.xilinx.com.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The High Speed SelectIO™ Wizard provides source HDL that implements I/O circuit for RX, TX and RX TX Separate bus with delay configurations, clocking and required buffers. Users can configure two interfaces with different data speeds. This Wizard provides various capture schemes and modes of operation. Unlike the 7 series SelectIO Interface Wizard, this core generates default pin LOC constraints for each selected bank for the configured bus. In addition, the High Speed SelectIO Wizard updates the connections between design blocks if the constraints are updated.

Feature Summary

- **Interface Selection**: Up to two interfaces with different bus configurations are supported.
- **Interface<k> Settings**: Data, delay, control and clocking attribute settings of the PHY block in native mode. The variable <k> equals the number of interfaces minus one (interfaces-1). Possible values for this core are 0 and 1. Details of these settings are explained in Interface<k> Settings Tab.
- **Data**: Configures bus direction, bus signal type, bus IOSTANDARD, bus data width, serialization factor and RX capture scheme.
- **Delay**: Configures RX delay cascade, TX delay type, TX delay value, RX delay type, RX delay value, clock forward delay type and clock forward delay value.
- **Control**: Configures FIFO sync mode, TX output phase 90, RIU interface, invert RX clock, enable VTC port, serial mode, clock source, RX clk phase P, RX clock phase N, rounding factor, CTRL clock.
- **Clocking**: Configures input clock frequency, Interface<k> data speed, clock signal type, clock IOSTANDARD, clock forward and clock forward signal Type.
- **Interface<k> Pin Assignment**: Provides selection of Interface<k> bank available for the selected device, as well as the default pin LOC of the data and clock pins for the selected bank (can be updated to a new pin LOC from the available list). The port renaming feature updates the data and clock port names. For multiple instances of this core in a design, data and clock ports must be renamed so that conflicting constraints are not generated. With this feature, multiple High Speed SelectIO Wizard cores can be configured and used in a single system.

Applications

This solution is useful for high-speed I/O interface requirements like ASIC Emulation and chip-to-chip interaction.

Unsupported Features

The following features are not currently supported:

- Bi-directional bus configuration
- Bitslip functionality
- RX capture scheme bus (for this core, this bus is the same as BIT)

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

The High Speed SelectIO™ Wizard provides valid configuration options to design an I/O circuit using native components of the PHY block, such as RX_BITSLICE, TX_BITSLICE and BITSLICE_CONTROL for Xilinx UltraScale™ architecture-based devices. This wizard also configures clocking circuitry using PLLE3. Design blocks shown in Figure 2-1 are generated based on the selected configuration.

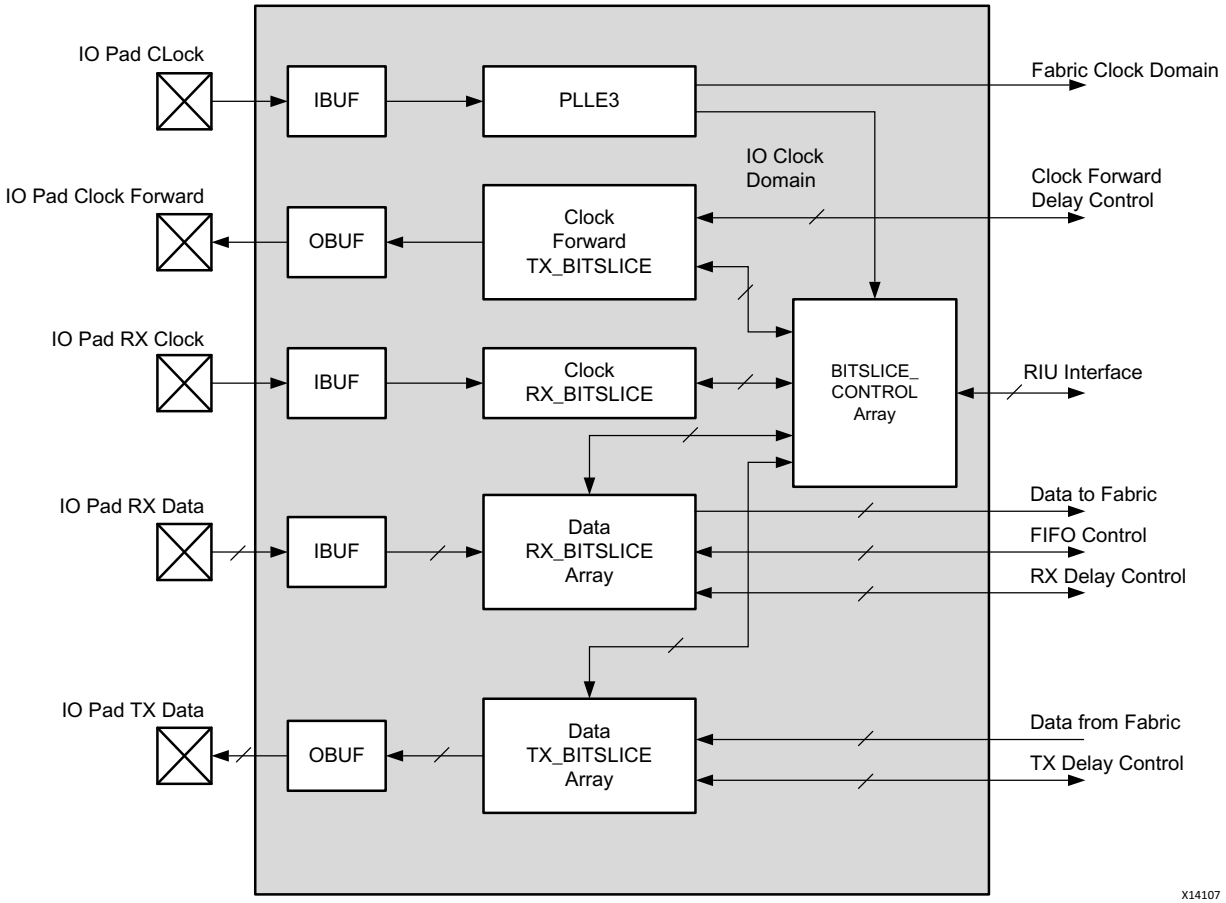

Figure 2-1: **Block Diagram**

SelectIO components in UltraScale architecture-based PHY have dedicated connections among different blocks. These connections should be switched to different ports depending on the pin LOC constraints on the top-level ports. This core generates the HDL and XDC files required.

Performance

This core provides an RX capture scheme as DPA, BUS or BIT. BUS and BIT capture schemes are the same for this version of core. High-performance operations should use DPA mode with delay calibration logic.

Maximum Frequencies

This core supports a maximum data speed of 1600 MHz for I/O. Fabric logic works at 1/8 or 1/4 clock of the I/O clock. If RX is configured in serial mode or the clock source is external, fabric logic works at 1/4 clock (8:1 serialization) or 1/2 clock (4:1 serialization) of the I/O clock.

Resource Utilization

This core uses FF to generate `clkoutphyen` of the PLLE3 for the circuit stabilization before the actual data is sent on the I/O. Other blocks used are FPGA hard blocks.

Kintex UltraScale Devices

Table 2-1 provides approximate resource counts for the various core options on Kintex® UltraScale devices. Resources required for the High Speed SelectIO core have been estimated for the XCKU040-FFVA1 156 devices. These values were generated using Vivado® IP Catalog. They are derived from post-synthesis reports, and might change during implementation.

Table 2-1: **Device Utilization – Kintex UltraScale Devices (XCKU040-FFVA1156)**

Number of Interfaces	Device Resources		Max Core Clock Frequency (MHz)
	LUTs	FFs	
1	14	11	200
2	28	22	200

The maximum clock frequency results were obtained by double-registering input and output ports to reduce dependence on I/O placement. The inner level of registers used a separate clock signal to measure the path from the input registers to the first output register through the core. The results are post-implementation, using tool default settings except for high effort.

The resource usage results do not include the characterization registers and represent the true logic used by the core. LUT counts include SRL16s or SRL32s.

Clock frequency does not take clock jitter into account and should be derated by an amount appropriate to the clock source jitter specification. The maximum achievable clock frequency and the resource counts might also be affected by other tool options, additional logic in the FPGA, using a different version of Xilinx tools, and other factors.

Port Descriptions

Table 2-2 and Table 2-3 describe the input and output ports of the High Speed SelectIO Wizard. Availability of ports is controlled by user-selected parameters. For example, when DPA is selected, only ports associated with DPA are exposed. Any port that is not exposed is tied off or connected to a signal labeled as unused in the delivered source code.

In Table 2-2 and Table 2-3, the following variables are used:

- <k> = Interface Number
- m = Data Width
- n = Serialization Factor
- i = Bitslice Control Position

Table 2-2 lists ports that are top-level ports connected to FPGA I/O. These ports can be renamed. The renaming feature is useful for multiple instances of the core in the design to avoid conflicting pin LOC constraints.

Table 2-2: **Ports Connected to FPGA I/O**

Port	Direction	Description
if<k>_clk_p	Input	Differential clock input P connected to PLLE3 through IBUFDS.
if<k>_clk_n	Input	Differential clock input N connected to PLLE3 through IBUFDS.
if<k>_data_from_pins_p [m-1:0]	Input	Differential serial data input P from FPGA I/O connected to RX_BITSLICE through IBUFDS.
if<k>_data_from_pins_n [m-1:0]	Input	Differential serial data input N from FPGA I/O connected to RX_BITSLICE through IBUFDS.
if<k>_data_to_pins_p [m-1:0]	Output	Differential serial data output from TX_BITSLICE through P side of OBUFDS to FPGA I/O.
if<k>_data_to_pins_n [m-1:0]	Output	Differential serial data output from TX_BITSLICE through N side of OBUFDS to FPGA I/O.
if<k>_clk_fwd_to_pins_p	Output	Differential clock forward output from TX_BITSLICE through P side of OBUFDS to FPGA I/O.
if<k>_clk_fwd_to_pins_n	Output	Differential clock forward output from TX_BITSLICE through N side of OBUFDS to FPGA I/O.

Table 2-2: **Ports Connected to FPGA I/O *(Cont'd)***

Port	Direction	Description
if<k>_clk	Input	Single ended clock input connected to PLLE3 through IBUF.
if<k>_data_from_pins [m-1:0]	Input	Single ended serial data input from FPGA I/O connected to RX_BITSLICE through IBUF.
if<k>_data_to_pins [m-1:0]	Output	Single ended serial data output from TX_BITSLICE through OBUF to FPGA I/O.
if<k>_clk_fwd_to_pins	Output	Single ended clock forward output from TX_BITSLICE through OBUF to FPGA I/O.

Table 2-3: **Ports Connected to FPGA Fabric Logic**

Port	Direction	Description
Data		
if<k>_data_to_fabric[n*m-1:0]	Output	Parallel data output to the fabric from *m* RX bit slices of Interface<k>.
if<k>_dpa_data_to_fabric[n*m-1:0]	Output	Parallel DPA data output to the fabric from *m* RX bit slices of Interface<k>.
if<k>_data_from_fabric[n*m-1:0]	Input	Parallel DPA data input from fabric to *m* TX bit slices of Interface<k>.
Clock		
if<k>_div_clk_to_fabric	Output	Divided version of clock from PLLE3 CLKOUT0 for fabric logic. Frequency of this is data speed divided by the serialization factor.
if<k>_ext_clk_to_fabric	Output	Divided version of clock from PLLE3 CLKOUT1 for fabric logic. Frequency of this is data speed divided by the serialization factor.
RX Delay Control		
if<k>_rx_ce [m-1:0]	Input	Clock enable for the IDELAY register clock for RX.
if<k>_rx_inc [m-1:0]	Input	Increment the current delay tap setting for RX.
if<k>_rx_load [m-1:0]	Input	Load the count value from CNTVALUEIN for RX.
if<k>_rx_cntvaluein [m*9-1:0]	Input	Counter value from the FPGA logic for dynamically loadable tap value for RX.
if<k>_rx_cntvalueout [m*9-1:0]	Output	Counter value going to FPGA logic for monitoring tap value for RX.
if<k>_rx_ce_ext [m-1:0]	Input	Extended clock enable for the DELAY for RX.
if<k>_rx_inc_ext [m-1:0]	Input	Extended increment the current delay tap setting for RX.
if<k>_rx_load_ext [m-1:0]	Input	Extended load the count value from CNTVALUEIN for RX.
if<k>_rx_cntvaluein_ext [m*9-1:0]	Input	Extended counter value from the FPGA logic for dynamically loadable tap value for RX.

Table 2-3: **Ports Connected to FPGA Fabric Logic *(Cont'd)***

Port	Direction	Description
if<k>_rx_cntvalueout_ext [m*9-1:0]	Output	Extended counter value going to the FPGA logic for monitoring tap value for RX.
if<k>_rx_clk	Input	DELAY clock used to sample LOAD, CE INC for RX.
if<k>_rx_rst_dly	Input	Reset the internal IDELAY to value defined in DELAY_VALUE for RX.
if<k>_rx_dpa_ce [m-1:0]	Input	DPA clock enable for the IDELAY register clock for RX.
if<k>_rx_dpa_inc [m-1:0]	Input	DPA increment the current delay tap setting for RX.
if<k>_rx_dpa_load [m-1:0]	Input	DPA load the count value from CNTVALUEIN for RX.
if<k>_rx_dpa_cntvaluein [m*9-1:0]	Input	DPA counter value from the FPGA logic for dynamically loadable tap value for RX.
if<k>_rx_dpa_cntvalueout [m*9-1:0]	Output	DPA counter value going to FPGA logic for monitoring tap value for RX.
if<k>_rx_dpa_ce_ext [m-1:0]	Input	Extended clock enable for the DELAY for RX.
if<k>_rx_dpa_inc_ext [m-1:0]	Input	DPA extended increment the current delay tap setting for RX.
if<k>_rx_dpa_load_ext [m-1:0]	Input	DPA extended load the count value from CNTVALUEIN for RX.
if<k>_rx_dpa_cntvaluein_ext [m*9-1:0]	Input	DPA extended counter value from FPGA logic for dynamically loadable tap value for RX.
if<k>_rx_dpa_cntvalueout_ext [m*9-1:0]	Output	DPA extended counter value going to FPGA logic for monitoring tap value for RX.
if<k>_rx_dpa_clk	Input	DPA DELAY clock used to sample LOAD, CE INC for RX.
TX Delay Control		
if<k>_tx_ce [m-1:0]	Input	Clock enable for the ODELAY register clock for TX.
if<k>_tx_inc [m-1:0]	Input	Increment the current delay tap setting for TX.
if<k>_tx_load [m-1:0]	Input	Load the count value from CNTVALUEIN for TX.
if<k>_tx_cntvaluein [m*9-1:0]	Input	Counter value from FPGA logic for dynamically loadable tap value for TX.
if<k>_tx_cntvalueout [m*9-1:0]	Output	Counter value going to the FPGA logic for monitoring tap value for TX.
if<k>_tx_clk	Input	DELAY Clock used to sample LOAD, CE INC for TX.
if<k>_tx_rst_dly	Input	Reset the internal ODELAY to value defined in DELAY_VALUE for TX.
if<k>_clk_fwd_ce [m-1:0]	Input	Clock enable for the ODELAY register clock for clk.
if<k>_clk_fwd_inc [m-1:0]	Input	Increment the current delay tap setting for clk.
if<k>_clk_fwd_load [m-1:0]	Input	Load the count value from CNTVALUEIN for clk.

Table 2-3: **Ports Connected to FPGA Fabric Logic *(Cont'd)***

Port	Direction	Description
if<k>_clk_fwd_cntvaluein [m*9-1:0]	Input	Counter value from FPGA logic for dynamically loadable tap value for clk forward.
if<k>_clk_fwd_cntvalueout [m*9-1:0]	Output	Counter value going to FPGA logic for monitoring tap value for clk forward.
if<k>_clk_fwd_clk	Input	DELAY Clock used to sample LOAD, CE INC for clk.
RIU		
if<k>_riu_rd_data_bsc<i>	Output	Output read data to the controller.
if<k>_riu_valid_bsc<i>	Output	Output read valid to the controller.
if<k>_riu_addr_bsc<i>	Input	Address of the register.
if<k>_riu_clk_bsc<i>	Input	System clock from fabric.
if<k>_riu_nibble_sel_bsc<i>	Input	Nibble select to enable RIU read/write.
if<k>_riu_wr_data_bsc<i>	Input	Input write data to the register.
if<k>_riu_wr_en_bsc<i>	Input	Register write enable active-High.
Status/Control		
if<k>_rx_fifo_empty [m-1:0]	Output	FIFO empty flag.
if<k>_rx_fifo_wrclk_out [m-1:0]	Output	FIFO source synchronous write clock out to the FPGA logic.
if<k>_rx_fifo_rd_clk [m-1:0]	Input	Read clock for each bit FIFO.
if<k>_rx_fifo_rd_en [m-1:0]	Input	FIFO enable for each bit FIFO.
if<k>_rx_dpa_fifo_empty [m-1:0]	Output	DPA FIFO empty flag.
if<k>_rx_dpa_fifo_wrclk_out [m-1:0]	Output	DPA FIFO source synchronous write clock out to the FPGA logic.
if<k>_rx_dpa_fifo_rd_clk [m-1:0]	Input	DPA read clock for each bit FIFO.
if<k>_rx_dpa_fifo_rd_en [m-1:0]	Input	DPA FIFO enable for each bit FIFO.
if<k>_vtc_rdy_bsc<i>	Output	PHY calibration is complete (VTC is ready – after EN_VTC is enabled).
if<k>_en_vtc_bsc<i>	Input	Active-High to enable DELAYCTRL to keep delay over voltage and temp low to load new delay.
if<k>_dly_rdy_bsc<i>	Output	Indicates fixed delay calibration completion and fabric can start eye-training.
if<k>_locked	Output	Logic High indicates PLLE3 is locked to the desired clock frequency.

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

This core is for high-speed UltraScale™ architecture-based designs and can be configured for a data speed range of 500-1600 MHz. Following steps are recommended for all designs using the High Speed SelectIO Wizard.

1. Instantiate the High Speed SelectIO Wizard in the system as shown in Figure 2-1.
2. Configure the core for required bus direction (Data Settings, page 17).
3. For RX bus direction, choose appropriate capture scheme (Data Settings, page 17).
4. Configure the bus signal type and IOSTANDARD (Data Settings, page 17).
5. Configure the core for required number of inputs and/or outputs (Data Settings, page 17).
6. Configure the core for required delay type and delay value (Delay Settings, page 19).
7. Connect proper clock and reset connection to all the available ports (Clocking Settings, page 21).
8. Select the required bank from a list of available banks (Interface<k> Settings Tab, page 17)
9. Perform pin assignment as required by the hardware.

Clocking

The clock source for all the RX/TX_BITSLICE and BITSLICE_CONTROL is `clkoutphy` from PLLE3 when you select **Clk Source** as PLL in the wizard GUI. In this mode, the data speed is the same as the `clkoutphy` frequency. The reference clock source for the core is always PLL `clkoutphy`. When in serial mode, the data speed is twice the `clkoutphy` frequency. The clock for FPGA fabric logic is a divided clock version from the PLLE3.

When **Clk Source** in the GUI is selected as External, RX/TX_BITSLICE and BITSLICE_CONTROL are clocked by the external clock, and the reference clock source for the core is `clkoutphy`. In this mode, the data speed is twice the external clock frequency.

The signal `clkoutphy` is enabled using a counter running at a divided clock from PLLE3 to settle core logic before actual data is transferred to the FPGA I/O.

Two interfaces can be split across same bank with independent frequencies at a byte group granularity, as shown in Figure 3-1.

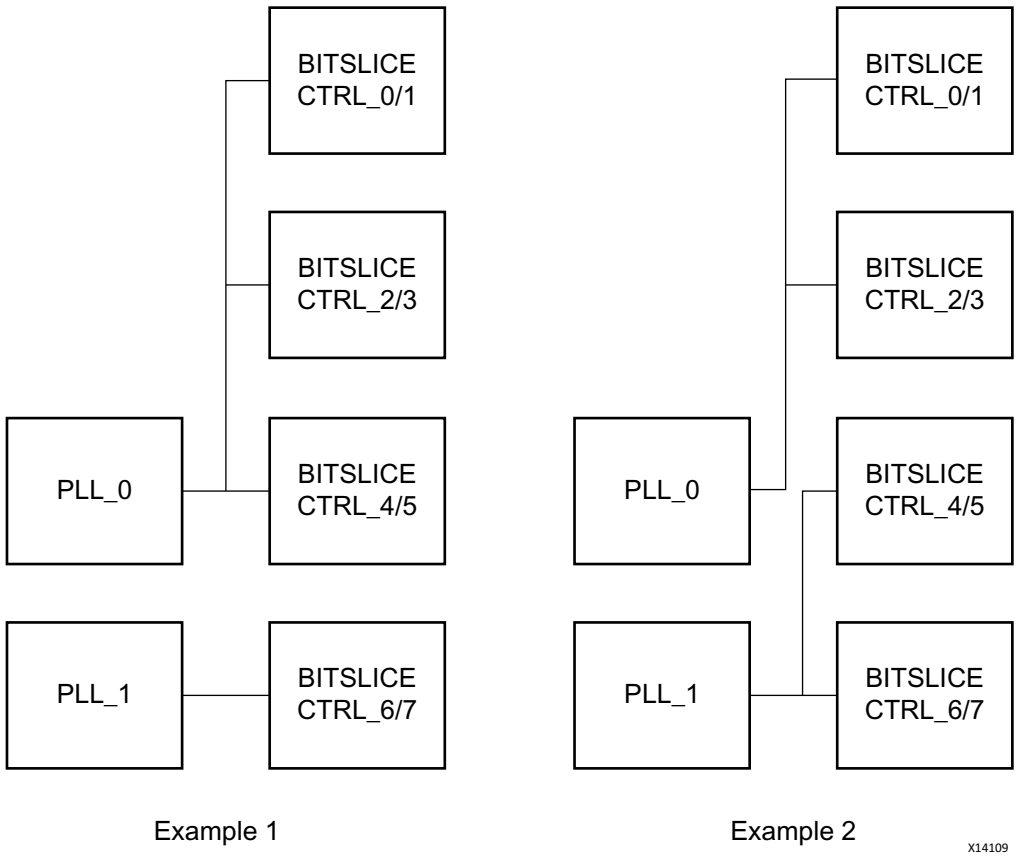

Figure 3-1: **Two Interfaces in Same Bank**

Resets

The reset for this core is an asynchronous reset connected to PLLE3. A locked output is inverted and connected to reset of individual blocks.

Protocol Description

The BITSLICE_CONTROL primitive controls the clocking and characteristics of RX_BITSLICE. Control of BITSLICE_CONTROL is via a register interface unit (RIU), which is a bank of 64 registers each of 16 bits. The RIU register map gives access to all the required delay and control values for the nibble group being programmed. For more details please refer to *UltraScale Architecture SelectIO Resources: Advance Specification User Guide* (UG571) [Ref 9].

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows in the IP Integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite. You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details about starting a Vivado project, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

For details about output generation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Note: Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

General GUI Settings

- **Component Name**: Component name is user defined. Component names must not contain any reserved words in Verilog or VHDL.
- **Number of Interface**: Select a maximum of two interfaces either in separate banks or the same bank with different configuration settings.

IMPORTANT: *When the Number of Interface is set to two, the GUI provides interface and pin settings tabs for both interfaces.*

Interface<k> Settings Tab

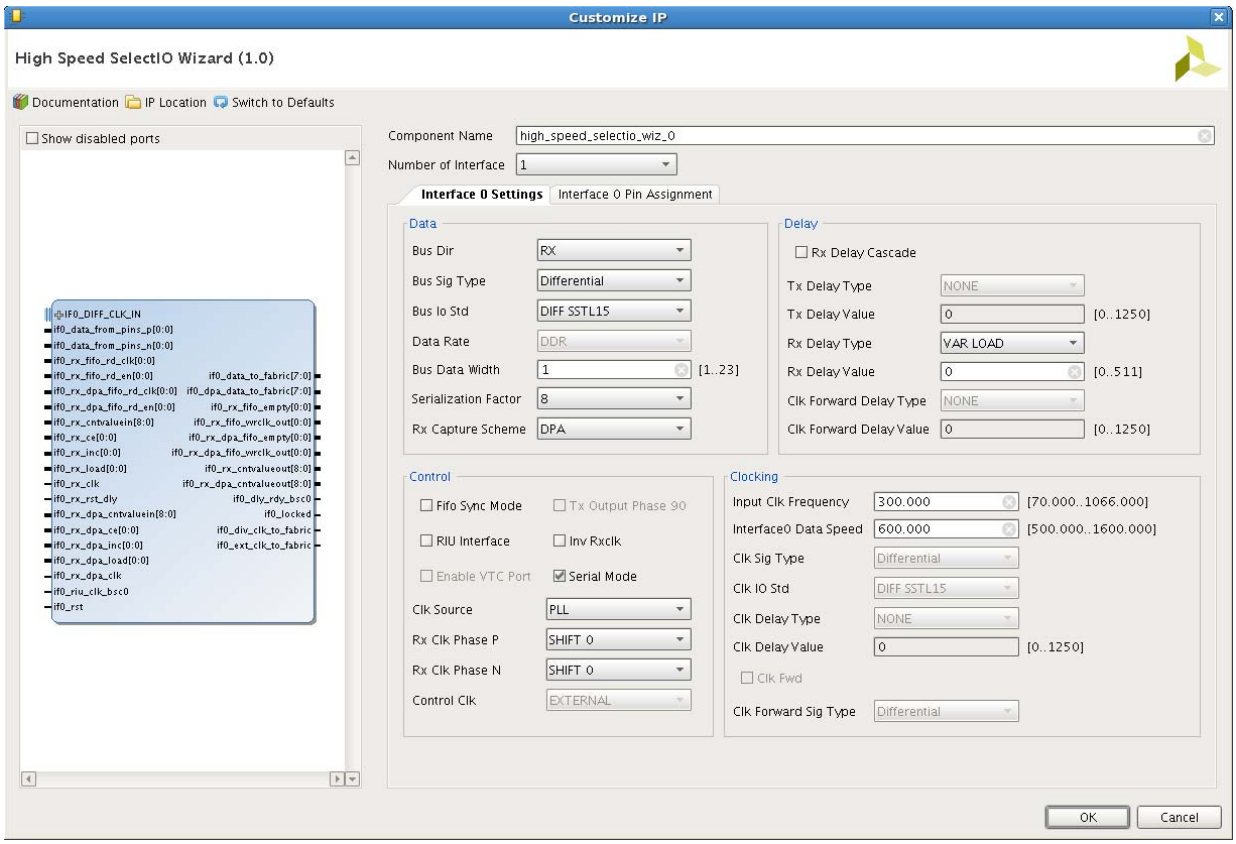

Figure 4-1: **Interface0 Settings for Number of Interface Set to 1**

Data Settings

- **Bus Direction**: Selects RX, TX or RX and TX separate for the interface. Additional bus options are enabled in the wizard GUI after selecting a bus direction. RX is set as default.
- **Bus Sig Type**: Selects differential or single-ended bus signal type. Selecting differential instantiates differential input/output buffers. The single-ended type instantiates

single-ended input/output buffers in the data path. Changes to this parameter also updates the **Clk Sig Type** with the same value.

- **Bus IO Std**: Lists all the available I/O standards of differential or single-ended buffers in the selected bank for the device.
- **Data Rate**: DDR is the only rate available for the RX/TX_BITSLICE. SDR functionality can be achieved by skipping alternate bits from the RX or TX data to/from High Speed SelectIO Wizard.
- **Bus Data Width**: The bus data width range depends on the **Bus Sig Type** selection. For differential signal type, the range is 1-24, and for single-ended signal type, the range is 1-47.

  Note: The range is 1-46 for single-ended signal types for TX when clock forwarding is enabled.

  When two interfaces are in the same bank, the bus data width range is either half or 25 percent of the maximum width. For example, Interface 0 bus data width sets the range for Interface 1 to ensure the appropriate connection among design blocks.

- **Serialization Factor**: Defines the serialization factor for parallel data input width from the fabric. Legal values are 4 and 8. The interface data speed maximum range becomes 800 MHz if a serialization factor of 4 is selected. This parameter is set to 8 by default.
- **Rx Capture Scheme**: Rx Capture schemes are provided to configure the interface for high-speed operations. DPA (Dynamic Phase Alignment) uses differential input differential output buffers to sample P and N data separately and allows calibration on the data. BIT mode allows calibration of the delays on each data input. BUS mode allows calibration of the delay on the clock only and keeps a fixed delay on the data. Currently BUS mode is mapped to BIT mode.
  - **DPA Circuitry**: This mode uses two RX bitslices for a single bit of data. P output of IBUFDS_DIFF_OUT connects to actual data, and N output is connected to eye training data bitslice. This mode can be used with delay calibration logic to compensate for any variation in the input data, such as sampling clock reference or variation (voltage and temperature). Figure 4-2 details the DPA circuitry.

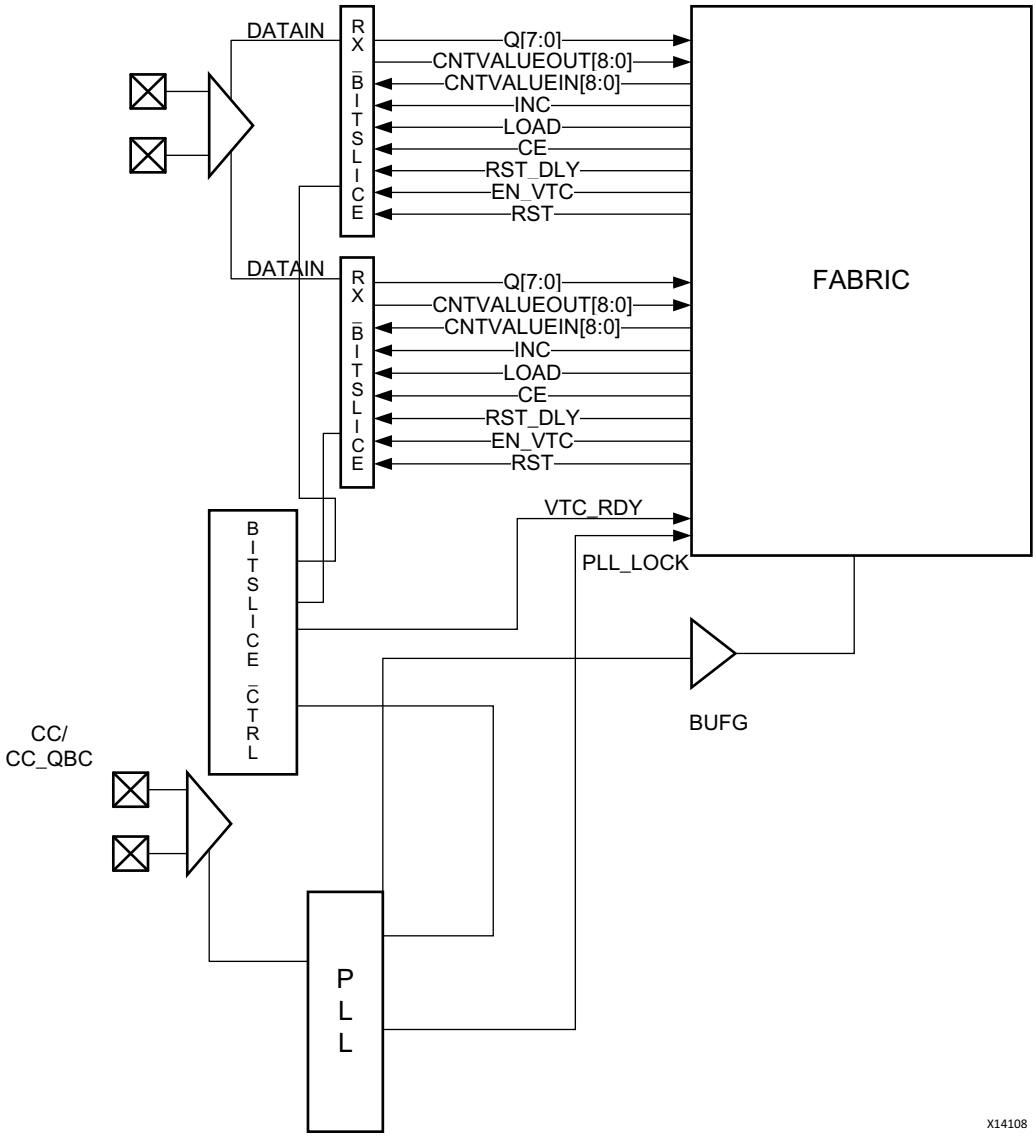

Figure 4-2: **DPA Capture Scheme**

Delay Settings

- **Rx Delay Cascade**: Enables cascading of IDELAY and extended delay lines to get a total of 2.5 ns delay on RX data path.
- **Tx Delay Type**:
  - **FIXED**: Fixed delay value set through **Tx Delay Value** is applied on TX data.
  - **VARIABLE**: Delay on TX data can be incremented or decremented from the default value using delay control inputs CE, CLK, and INC.

  - **VAR_LOAD**: Delay on TX Data can be incremented or decremented from the default set value, or loaded with a new value on CNTVALUEIN using delay control inputs CE, CLK, and INC.

- **Tx Delay Value**: Value of desired TX delay in pico seconds for **FIXED** mode and in COUNT (tap) for **VARIABLE** and **VAR_LOAD** modes. The output delay contains a 512 tap delay line with a maximum value of 1,250 ps. These taps are uncalibrated individually, but the logic to allow a conversion from a fixed value (in ps) to a certain number of taps is built into the I/O control logic. This logic requires a reference clock. No tap-dependent jitter is added by the delay line.
- **Rx Delay Type**:
  - **FIXED**: Fixed delay value set through **Rx Delay Value** is applied on RX data.
  - **VARIABLE**: Delay on RX data can be incremented or decremented from the default value using delay control inputs CE, CLK, and INC.
  - **VAR_LOAD**: Delay on RX data can be incremented or decremented from the default set value or loaded with a new value on CNTVALUEIN using delay control inputs CE, CLK, and INC.
- **Rx Delay Value**: Value of desired RX delay in pico seconds for **FIXED** mode and in COUNT (tap) for **VARIABLE** and **VAR_LOAD** mode. The input delay contains a 512 tap delay line with a maximum value of 1,250 ps when **Rx Delay Cascade** is false. If **Rx Delay Cascade** is enabled, this delay can be configured to 1024 taps or 2500 ps.
- **Clk Fwd Delay Type**:
  - **FIXED**: Fixed delay value set through **Clk Fwd Delay Value** is applied on Clk Forward.
  - **VARIABLE**: Delay on Clk Forward can be incremented or decremented from the default value using delay control inputs CE, CLK, and INC.
  - **VAR_LOAD**: Delay on Clk Forward can be incremented or decremented from the default set value or loaded with a new value on CNTVALUEIN using delay control inputs CE, CLK, and INC.
- **Clk Fwd Delay Value**: Value of desired Clk Forward Delay in pico seconds for **FIXED** mode, and in COUNT (tap) for **VARIABLE** and **VAR_LOAD** modes. The output delay contains a 512 tap delay line with a maximum value of 1,250 ps.

Control Settings

- **Fifo Sync Mode**: Enable this option when the internal write clock and the FIFO_RD_CLK are coming from a common source. Disabling this option causes the internal two-stage flop synchronizers to be used. This adds latency when "not FIFO_EMPTY" asserts after data is first written to the FIFO.
- **Tx Output Phase 90**: Delays TX output data phase by 90 degrees.

- **RIU Interface**: Enables Register Interface Unit (RIU) per bitslice to access internal registers. Every delay element's tap setting can be read with the RIU. Various features, such as clock gating and Voltage Temperature (VT) tracking can be disabled. With this option, you can dynamically change the FIFO usage (for example, from synchronous to asynchronous to full bypass).
- **Inv RX Clk**: Enabling this option inverts clock path from IOB to upper RX bitslice.
- **Enable VTC Port**: Enabling this option provides an option port for Voltage and Temperature Control (VTC).
- **Serial Mode**:
  - Enabled: The input clock for a data receiver comes from an external source via a PLLE3.
  - Disabled: The input clock for a data receiver comes from RX_BITSLICE 0. Used for SGMII.
- **Clk Source**: Selects PLL or EXTERNAL clock as the clock source. When EXTERNAL clock is selected, the input clock for a data receiver comes from RX_BITSLICE 0. For PLL, `clkoutphy` from PLL is used for both data transmitter and receiver.
- **Rx Clk Phase P**: Selecting SHIFT_90 shifts the read clock by 90 degrees relative to read DQ.
- **Rx Clk Phase N**: Selecting SHIFT_90 shifts the read clock by 90 degrees relative to read DQ.
- **Control Clk**: This is always set to EXTERNAL. EXTERNAL uses `riu_clk` for BITSLICE_CONTROL to control the circuitry clock.

Clocking Settings

- **Input Clk Frequency**: For a clock source of PLL, the input clock frequency range is 70-1066 MHz. When EXTERNAL is selected as the clock source:
  - For a serialization factor of eight, the input clock frequency range is 300-800 MHz.
  - For a serialization factor four, the input clock frequency range is 300-400 MHz.
- **Interface<k> Data Speed**:
  - Clk source PLL:
    - For a serialization factor of eight, the Interface<k> data speed range is 500-1600 MHz.
    - For a serialization factor of four, the Interface<k> data speed range is 500-800 MHz.
  - Clk source EXTERNAL:

  - For a serialization factor of eight, the Interface<k> data speed range is 600-1600 MHz.
  - For a serialization factor of four, the Interface<k> data speed range is 600-800 MHz.

- **Clk Sig Type**: Same signal type as **Bus Sig Type**. This option is disabled in the current version of the core.
- **Clk IO Std**: Same I/O standard as **Bus IO Std**. This option is disabled in the current version of the core.
- **Clk Delay Type**: This option is disabled in the current version of the core.
- **Clk Delay Value**: This option is disabled in the current version of the core.
- **Clk Fwd**: When enabled, forwards clock along with the data in TX or RX and TX Separate mode for source synchronous data transfer application.
- **Clk Forward Sig Type**: Same signal type as **Bus Sig Type**. This option is disabled in the current version of the core.

Interface Pin Assignment Tab

This tab provides a bank selection option for the interface. Selections in this tab rename the ports are connected to FPGA pins and pin LOC constraints for data, interface clock source and clk forward ports.

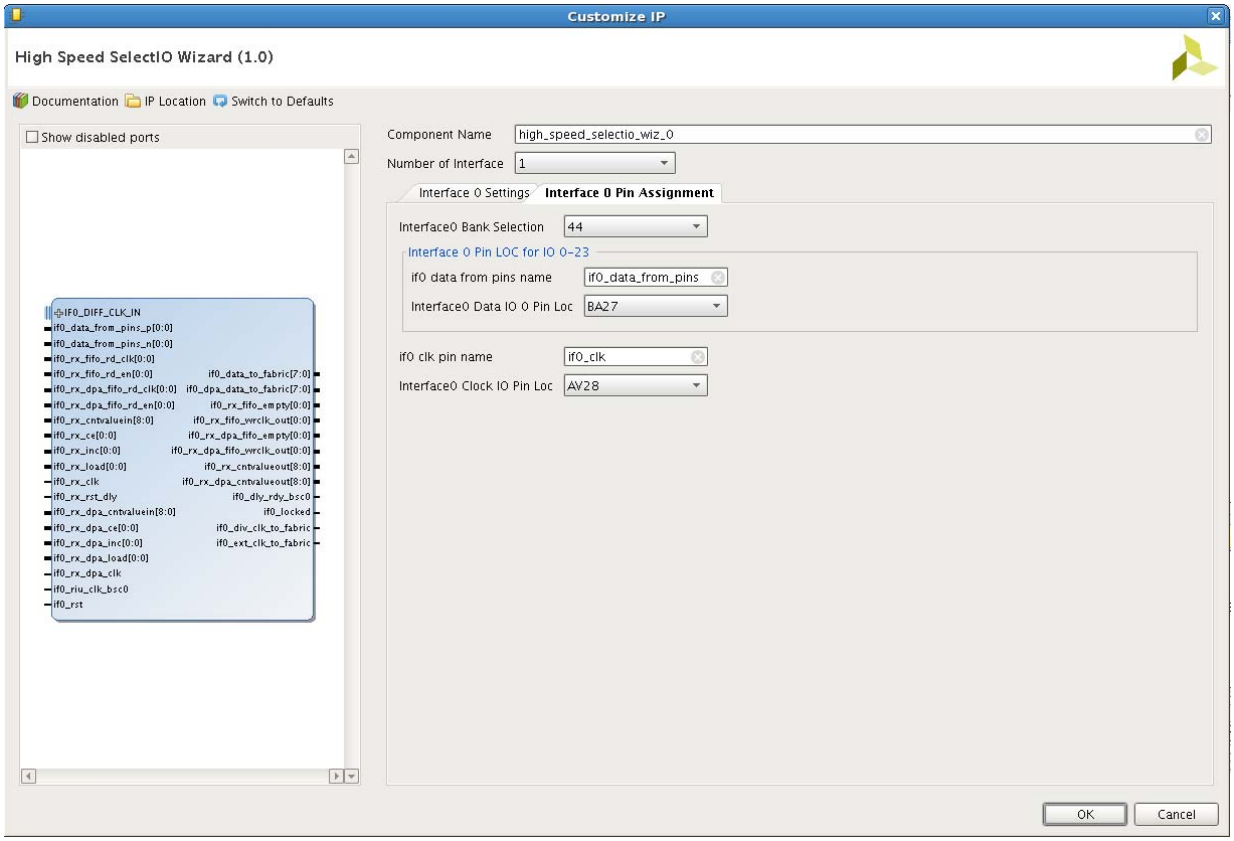

Figure 4-3: **Interface Pin Assignment Tab**

- **Interface<k> Bank Selection**: Lists all available High Performance (HP) and High Range (HR) banks for the selected device.
- **Interface<k> Pin Loc**: Provides port renaming box with a list of available pins for the selected bank.
- **IO Clk Pin Name**: Port renaming for the clock pin along with available global clock capable pins.

Output Generation

The core delivers verilog RTL for the core logic, example design and example test bench. The following files are created when core is configured and output products are generated:

- `<ComponentName>.xci`
- `<ComponentName>.veo`
- `<ComponentName>_if<k>_bitslice_array.v`
- `<ComponentName>_if<k>_bitslice_control_array.v`

- `<ComponentName>_hsio.v`
- `<ComponentName>.v`

If the IP example design project is opened, another core instance with the core name <ip_ex_inst> is instantiated in the `<ComponentName>_exdes.v`. For example design simulation, the `<ComponentName>_tb.v` test bench file is generated.

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

The core creates all the required clock and pin LOC constraints for the selected interfaces in the core-level XDC file.

Device, Package, and Speed Grade Selections

Input clock frequency selection depends on the maximum frequencies supported by the BUFG. Select the device, package and speed grades after referring to the *Kintex UltraScale Architecture Data Sheet* (DS892) for details on supported maximum frequencies [Ref 8].

Clock Frequencies

The I/O logic for this core works in the range 500-1600 MHz. The fabric logic for this core works up to 200 MHz.

Clock Management

All clocks are generated using PLLE3. For one interface per bank, one PLL is used. For two interfaces in a same bank, two PLLs are used.

IMPORTANT: *Core logic should be clocked with the divided version of the PLL output. When different interfaces have different data speeds, you have to synchronize fabric data accordingly.*

Clock Placement

The input clock is placed on global clock pins with the name `*_GC_*`. For an external clock source, the RX clock is placed on the `*_GC_QBC_*` pin.

Banking

This core can be used to configure an I/O circuit for High Range (HR) and High Performance (HP) banks. Available banks for the project part are provided in the GUI for selection. See Customizing and Generating the Core for more details.

Transceiver Placement

There are no transceiver placement constraints for this core.

I/O Standard and Placement

The High Speed SelectIO Wizard supports following I/O standards:

- High Performance Differential I/O

```
DIFF_HSTL_I DIFF_HSTL_I_18 DIFF_SSTL18_I DIFF_SSTL15 DIFF_SSTL135 DIFF_SSTL12
DIFF_HSUL_12 DIFF_HSTL_I_DCI DIFF_HSTL_I_DCI_18 DIFF_SSTL18_I_DCI DIFF_SSTL15_DCI
DIFF_SSTL135_DCI DIFF_SSTL12_DCI DIFF_HSUL_12_DCI LVDS SLVS_400_18 SUB_LVDS
DIFF_HSTL_I_12 DIFF_POD10 DIFF_POD12 DIFF_HSTL_I_DCI_12 DIFF_POD10_DCI
DIFF_POD12_DCI}
```

- High Performance Single Ended I/O

```
LVCMOS18 LVCMOS15 LVCMOS12 HSUL_12 LVDCI_18 LVDCI_15 HSUL_12_DCI HSLVDCI_18
HSLVDCI_15 HSTL_I HSTL_I_DCI HSTL_I_18 HSTL_I_DCI_18 HSTL_I_12  HSTL_I_DCI_12
SSTL18_I SSTL15 SSTL135 SSTL12 SSTL18_I_DCI SSTL15_DCI SSTL135_DCI SSTL12_DCI  POD10
POD12 POD10_DCI POD12_DCI
```

- High Range Differential I/O

```
DIFF_HSTL_I DIFF_HSTL_II DIFF_HSTL_I_18 DIFF_HSTL_II_18 DIFF_SSTL18_I DIFF_SSTL18_II
DIFF_SSTL15 DIFF_SSTL15_R DIFF_SSTL135 DIFF_SSTL135_R DIFF_SSTL12 DIFF_HSUL_12
LVDS_25 RSDS_25 TMDS_33 MINI_LVDS_25 PPDS_25 SUB_LVDS_25 SLVS_400_25
```

- High Range Single Ended I/O

```
LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12 HSUL_12 HSTL_I HSTL_II HSTL_I_18
HSTL_II_18 SSTL18_I SSTL18_II SSTL15 SSTL15_R SSTL135 SSTL135_R SSTL12 LVPECL
```

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

This core provides an example design with one core instance and one example instance. If the core is configured as RX , the example instance is configured as TX with the required settings for data speed, bank selection and other settings. Because data from the RX bitslices are unaligned data, the example design checks for all possible valid data of RX for a known TX data pattern. If a pattern matches, `if<k>_data_check_complete` output is asserted from the example design.

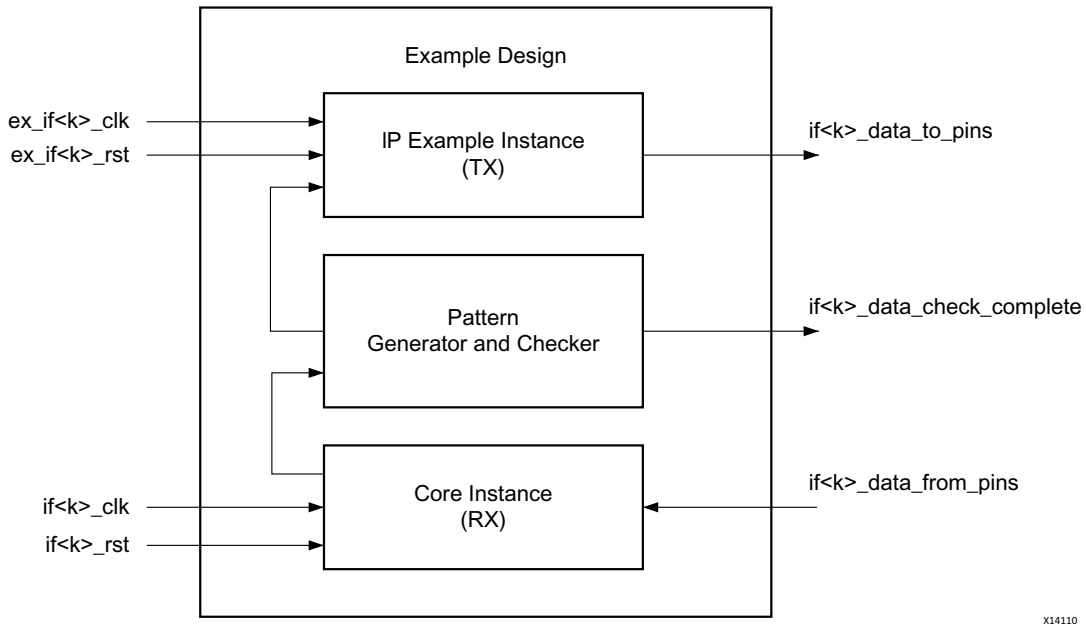

Figure 5-1: **Example Design Block Diagram**

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

The test bench is a simple Verilog code to exercise the example design and the core. This test bench performs following tasks:

- Generates the input clock signals.
- Applies a reset to the example design.
- Example design RX and TX interfaces are looped back.
- If an RX and TX pattern matches, the test bench sends a message for the successful test completion. Otherwise, it waits for 16000 cycles of input clock and sends a test failure message.

Appendix A

Verification, Compliance, and Interoperability

This appendix provides details about how this IP core was tested for compliance with the protocol to which it was designed.

Simulation

This core is verified with IES, VCS, Questa and XSIM simulators.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the High Speed SelectIO Wizard, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the High Speed SelectIO Wizard. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the High Speed SelectIO Wizard

AR: 60295

Contacting Technical Support

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

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Additional Resources.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Note: Access to WebCase is not available in all cases. Log in to the WebCase tool to see your specific support options.

Debug Tools

There are many tools available to address High Speed SelectIO Wizard design issues. It is important to know which tools are useful for debugging various situations.

Vivado Lab Tools

Vivado® lab tools insert logic analyzer and virtual I/O cores directly into your design. Vivado lab tools also allow you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 6].

Reference Boards

Various Xilinx development boards support the High Speed SelectIO Wizard. These boards can be used to prototype designs and establish that the core can communicate with the system.

- UltraScale Architecture evaluation board: KCU105

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado lab tools are a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Vivado lab tools for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- Ensure that all interfaces have obtained lock by monitoring the `if<k>_locked` port.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

For a glossary of technical terms used in Xilinx documentation, see the Xilinx Glossary.

References

These documents provide supplemental material useful with this product guide:

1. *Vivado® Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *ISE® to Vivado Design Suite Migration Guide* (UG911)
6. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
7. *Vivado Design Suite User Guide - Implementation* (UG904)
8. *Kintex UltraScale Architecture Data Sheet* (DS892)
9. *UltraScale Architecture SelectIO Resources: Advance Specification User Guide* (UG571)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
4/2/2014	1.0	Initial Xilinx release.

Please Read: Important Legal Notices