XAPP1330 (v1.1) August 3, 2018

Asynchronous Data Capture Using the High Speed SelectIO Wizard

Author: Jim Tatsukawa

Summary

This application note and associated reference design show how to use the High Speed SelectIO™ Wizard to generate an asynchronous receiver using the native mode I/O in UltraScale™ and UltraScale+™ devices.

A three-channel transmitter and receiver design can be implemented on a board such as the KCU105 board [Ref 1]. The reference design has been modified to provide easier pinout changes by using the High Speed SelectIO Wizard. The reference design targets the KCU105 board, but the design can be modified for other boards by following the guidelines described in this application note.

Download the reference design files for this application note from the Xilinx website. For detailed information about the design files, see Asynchronous Reference Design.

Introduction

Asynchronous data capture allows data to be captured when using differential inputs like LVDS without a forwarded clock as commonly used in synchronous interfaces. Instead, a state machine uses a reference clock and the variable delays within the RX_BITSLICE to recapture the clock from the data stream. The clock recovery algorithm requires the RX_BITSLICE to be directly controlled, which requires the High Speed SelectIO Wizard to be configured and connected to the clock recovery algorithms.

These are the restrictions in asynchronous data capture:

- 1300 Mb/s (-2 and 3 speed grades) or 1250 Mb/s (-1 speed grade)
- Inputs must use differential inputs

These are the restrictions in the reference design:

- For porting purposes, the receiver (RX) and transmitter (TX) must be placed in different byte groups
- The receiver must use ASYNC mode DATA_WIDTH = 4
- The transmitter must use DATA_WIDTH = 8
- Changes to pins might require changes to VHDL

- A differential 125 MHz clock is used to control the virtual I/O (VIO) interface for general-purpose I/Os

RECOMMENDED: *Drive clocks directly from global clock inputs that connect to the Quad byte clock strobe pins. The pin name contains the text string "GC_QBC." Directly connect global clock inputs to the PLL within the bank. Using the PLL with a cascaded MMCM clock source or sharing clocks across multiple banks might limit the performance.*

This application note is organized by the following topics:

- Asynchronous Data Capture
- Asynchronous Reference Design
- Simulation
- Hardware Operation

Asynchronous Data Capture

Synchronous interfaces transfer data with a clock that has a known phase and frequency related to the data being transferred. For example in a DDR interface, the clock is normally phase-aligned with data being captured on both the rising and falling edges of the clock.

For asynchronous interfaces, sometimes only data is transmitted. In other cases, a clock is transmitted with data, but the exact clock/data phase relationship is unknown. To recapture the asynchronous data in these cases, the receiver's reference clock must match the frequency of the data within ±100 ppm. For example, if the transmitter transmits data at 1250 Mb/s and the receiver requires a clock of 625 MHz, the reference clock can vary by ±0.125 MHz. For this reference design, the receiver's circuitry uses the RX_BITSLICE (native mode) along with control circuitry to track the phase alignment of the asynchronous data. This data recovery method is called phase tracking.

The phase tracking is loosely based on the Alexander Bang-Bang Phase Detector. As shown in Figure 1, a monitor is adjusted to be aligned to the edge of the data transitions based on using the active and monitor data. When monitor is at the edge of data, active is centered due to the separation of half a bit time (BtVal/2).

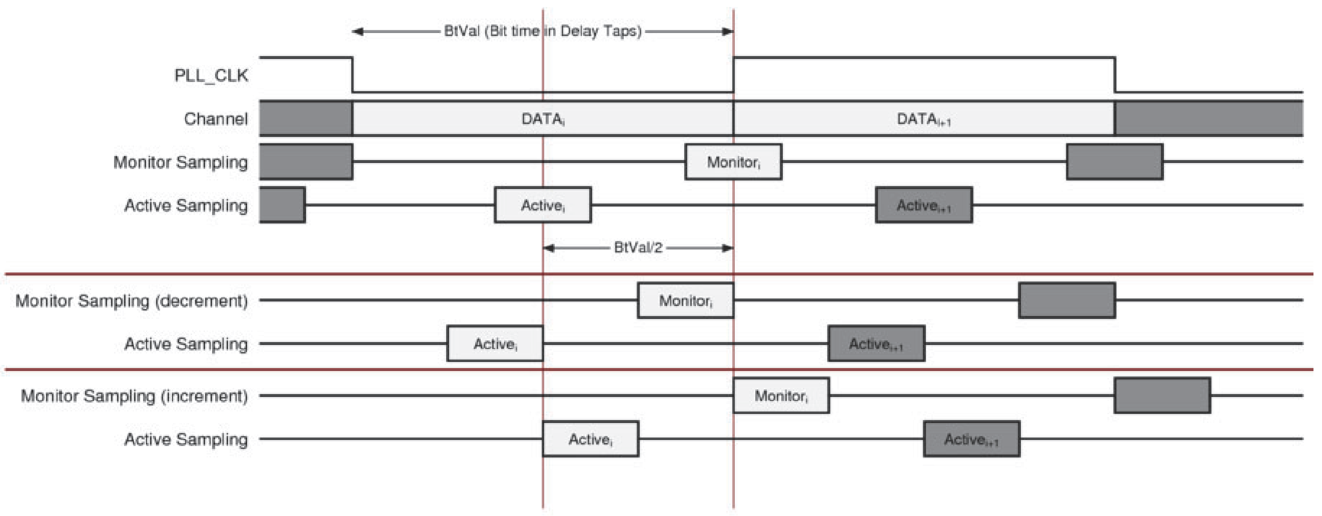


Figure 1: **Monitor Phase Tracking Adjustments (Active Delay > Monitor Delay)**

For this phase tracking to work, the delay tap value for a single bit time must be read from the BITSLICE_CONTROL. For a design operating at 1250 Mb/s, the bit time is 800 ps. A register interface unit (RIU) state machine records this delay value as BtVal, which is then used by the phase tracking algorithm. For example, if the average tap delay is 5 [ps/tap], then BtVal = 160 for this example calculation.

In UltraScale and UltraScale+ devices, differential pairs can use both RX_BITSLICEs, as shown in Figure 2. A state machine is used to control the delays for both the master and slave RX_BITSLICEs so that one of the bitslices samples the data at the center of a bit time (active) while the other bitslice samples data at the edge of the bit time (monitor).

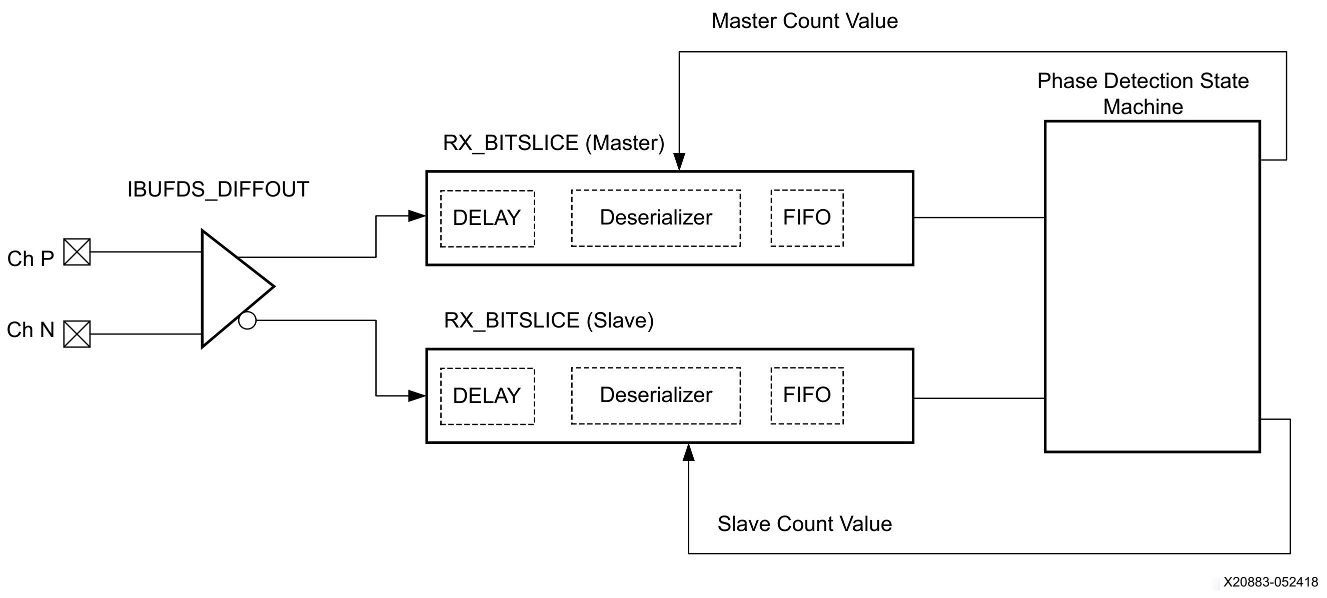


Figure 2: **Differential Channel For Asynchronous Phase Tracking**

In UltraScale devices, the phases are controlled by adjusting the CNTVALUEIN for the different RX_BITSLICEs, which inserts the delays into the datapath. Increasing the delay value has the effect of moving the sampling window negatively, as shown in Figure 3.

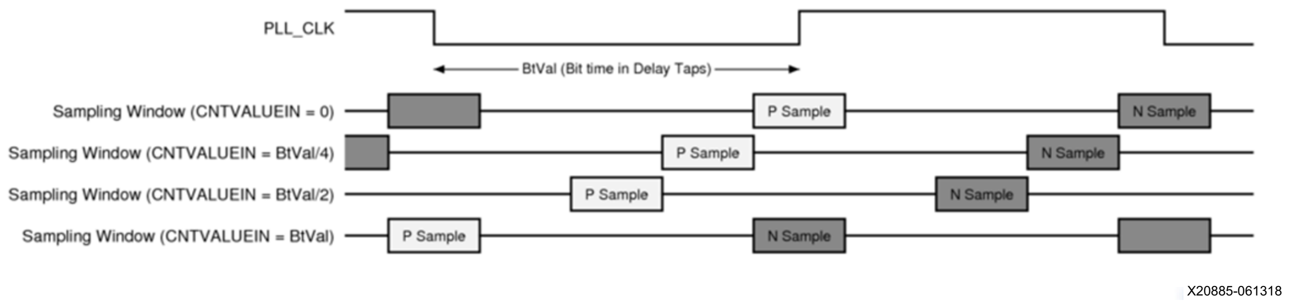

PLL_CLK
BtVal (Bit time in Delay Taps)
Sampling Window (CNTVALUEIN = 0)
Sampling Window (CNTVALUEIN = BtVal/4)
Sampling Window (CNTVALUEIN = BtVal/2)
Sampling Window (CNTVALUEIN = BtVal)
P Sample
N Sample
X20885-061318

Figure 3: **Data Sampling Window With RX_BITSLICE Delays**

Communication protocols use 8B/10B encoding to transmit data requiring a gearbox to convert the 4-bit patterns to 10-bit patterns. The state machine uses the 4-bit to 10-bit gearbox to additionally ensure data is not lost due to the phase tracking.

This gearbox also provides the additional purpose of compensating up to the 200 ppm difference between the data rate and reference clock. The gearbox is implemented as a 15-bit shift register and can gradually insert or remove an extra bit of data using oversampling and consequently ensuring no data is lost. For the gearbox, both the write and read side are clocked by a fabric clock running at the frequency of the parallel data (1250 Mb/s ÷ 4 = 312.5 MHz).

Asynchronous Reference Design

Download the reference design files for this application note from the Xilinx website.

Table 1 shows the reference design matrix.

Table 1: **Reference Design Matrix**

Parameter	Description
General	
Developer name	Jim Tatsukawa and Marc Defossez
Target devices	UltraScale FPGAs
Source code provided	Yes
Source code format	VHDL and Verilog
Design uses code and IP from existing Xilinx application note and reference designs or third party	Yes
Simulation	
Functional simulation performed	Yes
Timing simulation performed	Yes
Test bench used for functional and timing simulations	Yes

Table 1: **Reference Design Matrix *(Cont'd)***

Parameter	Description
Test bench format	Verilog
Simulator software/version used	Vivado® simulator
SPICE/IBIS simulations	No
Implementation	
Synthesis software tools/versions used	Vivado tools 2017.4 and later
Implementation software tools/versions used	Vivado tools 2017.4 and later
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	KCU105 board

The reference design accompanying this application note uses the High Speed SelectIO Wizard to generate the native mode interfaces for the TX and RX as shown in Figure 4.

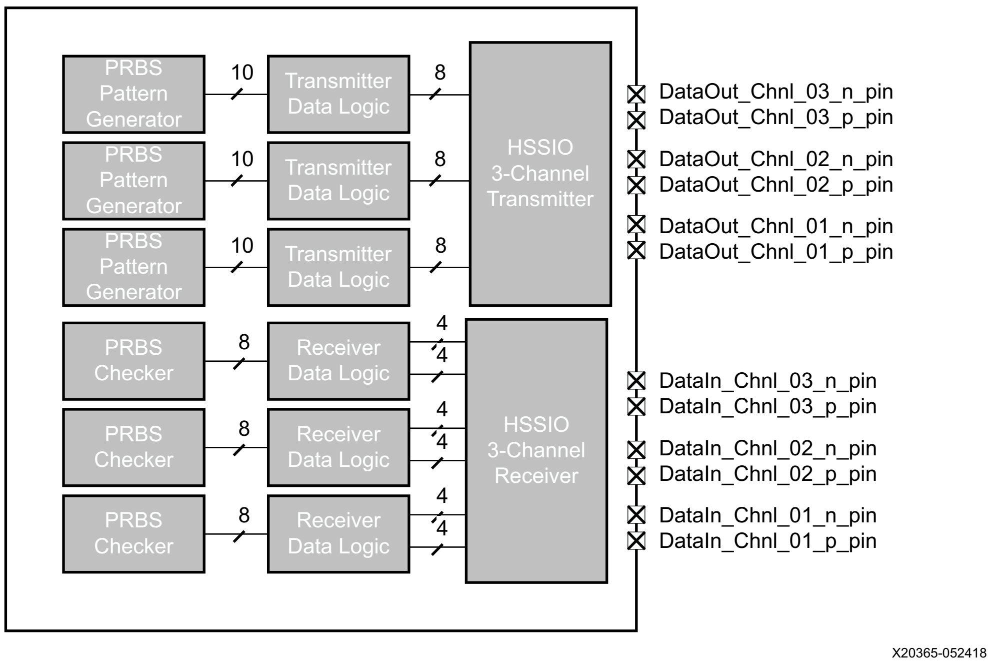
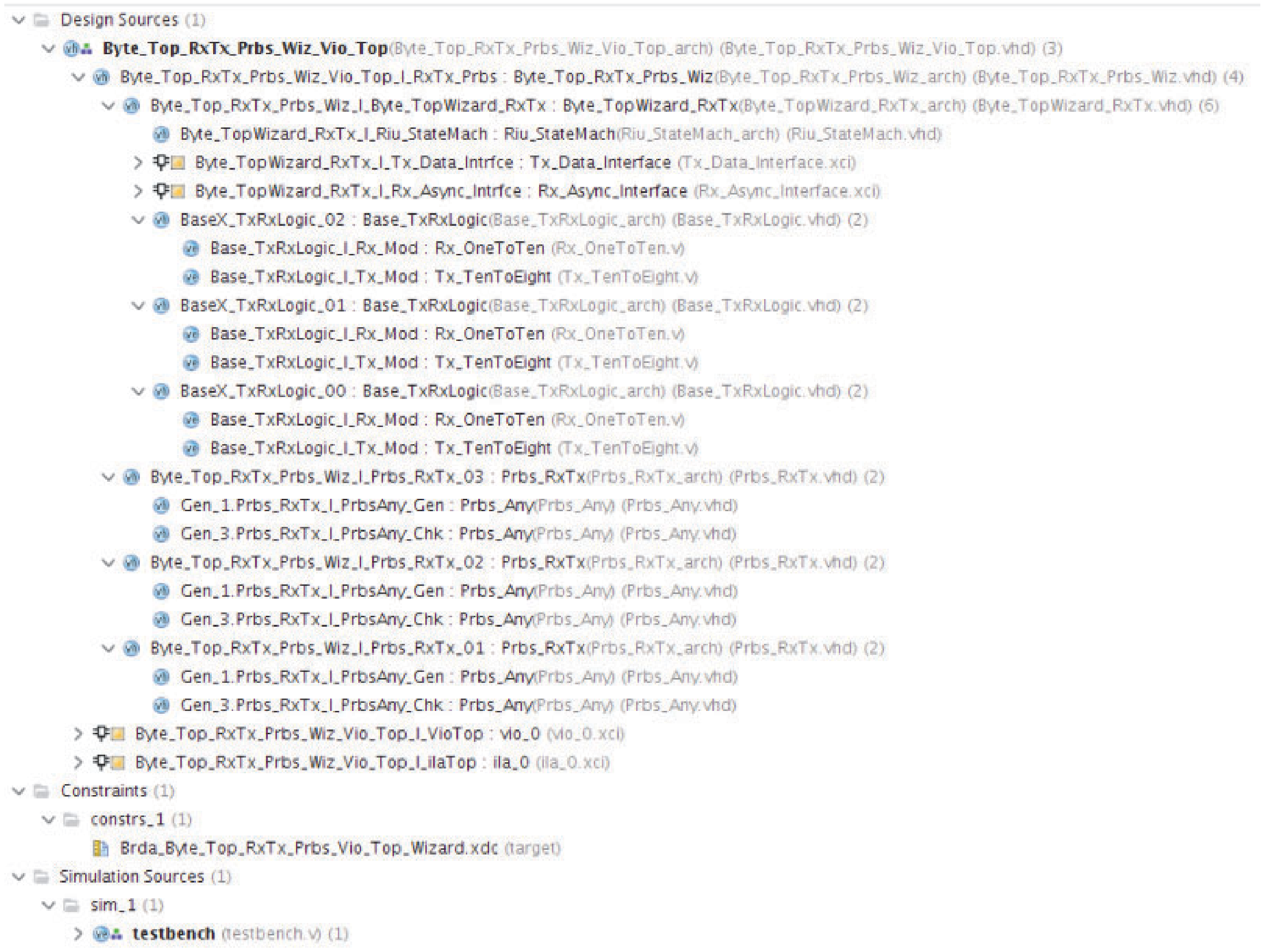


Figure 4: **Asynchronous Block Diagram**

The design files are organized as shown in Figure 5. In the download files, a TCL script (`top.tcl`) has been added to rebuild the project.



```
v Design Sources (1)
  v Byte_Top_RxTx_Prbs_Wiz_Vio_Top(Byte_Top_RxTx_Prbs_Wiz_Vio_Top_arch) (Byte_Top_RxTx_Prbs_Wiz_Vio_Top.vhd) (3)
    v Byte_Top_RxTx_Prbs_Wiz_Vio_Top_I_RxTx_Prbs : Byte_Top_RxTx_Prbs_Wiz(Byte_Top_RxTx_Prbs_Wiz_arch) (Byte_Top_RxTx_Prbs_Wiz.vhd) (4)
      v Byte_Top_RxTx_Prbs_Wiz_I_Byte_TopWizard_RxTx : Byte_TopWizard_RxTx(Byte_TopWizard_RxTx_arch) (Byte_TopWizard_RxTx.vhd) (6)
          Byte_TopWizard_RxTx_I_Riu_StateMach : Riu_StateMach(Riu_StateMach_arch) (Riu_StateMach.vhd)
        > Byte_TopWizard_RxTx_I_Tx_Data_Intrfce : Tx_Data_Interface (Tx_Data_Interface.xci)
        > Byte_TopWizard_RxTx_I_Rx_Async_Intrfce : Rx_Async_Interface (Rx_Async_Interface.xci)
        v BaseX_TxRxLogic_02 : Base_TxRxLogic(Base_TxRxLogic_arch) (Base_TxRxLogic.vhd) (2)
            Base_TxRxLogic_I_Rx_Mod : Rx_OneToTen (Rx_OneToTen.v)
            Base_TxRxLogic_I_Tx_Mod : Tx_TenToEight (Tx_TenToEight.v)
        v BaseX_TxRxLogic_01 : Base_TxRxLogic(Base_TxRxLogic_arch) (Base_TxRxLogic.vhd) (2)
            Base_TxRxLogic_I_Rx_Mod : Rx_OneToTen (Rx_OneToTen.v)
            Base_TxRxLogic_I_Tx_Mod : Tx_TenToEight (Tx_TenToEight.v)
        v BaseX_TxRxLogic_00 : Base_TxRxLogic(Base_TxRxLogic_arch) (Base_TxRxLogic.vhd) (2)
            Base_TxRxLogic_I_Rx_Mod : Rx_OneToTen (Rx_OneToTen.v)
            Base_TxRxLogic_I_Tx_Mod : Tx_TenToEight (Tx_TenToEight.v)
      v Byte_Top_RxTx_Prbs_Wiz_I_Prbs_RxTx_03 : Prbs_RxTx(Prbs_RxTx_arch) (Prbs_RxTx.vhd) (2)
          Gen_1.Prbs_RxTx_I_PrbsAny_Gen : Prbs_Any(Prbs_Any) (Prbs_Any.vhd)
          Gen_3.Prbs_RxTx_I_PrbsAny_Chk : Prbs_Any(Prbs_Any) (Prbs_Any.vhd)
      v Byte_Top_RxTx_Prbs_Wiz_I_Prbs_RxTx_02 : Prbs_RxTx(Prbs_RxTx_arch) (Prbs_RxTx.vhd) (2)
          Gen_1.Prbs_RxTx_I_PrbsAny_Gen : Prbs_Any(Prbs_Any) (Prbs_Any.vhd)
          Gen_3.Prbs_RxTx_I_PrbsAny_Chk : Prbs_Any(Prbs_Any) (Prbs_Any.vhd)
      v Byte_Top_RxTx_Prbs_Wiz_I_Prbs_RxTx_01 : Prbs_RxTx(Prbs_RxTx_arch) (Prbs_RxTx.vhd) (2)
          Gen_1.Prbs_RxTx_I_PrbsAny_Gen : Prbs_Any(Prbs_Any) (Prbs_Any.vhd)
          Gen_3.Prbs_RxTx_I_PrbsAny_Chk : Prbs_Any(Prbs_Any) (Prbs_Any.vhd)
    > Byte_Top_RxTx_Prbs_Wiz_Vio_Top_I_VioTop : vio_0 (vio_0.xci)
    > Byte_Top_RxTx_Prbs_Wiz_Vio_Top_I_ilaTop : ila_0 (ila_0.xci)
v Constraints (1)
  v constrs_1 (1)
      Brda_Byte_Top_RxTx_Prbs_Vio_Top_Wizard.xdc (target)
v Simulation Sources (1)
  v sim_1 (1)
    > testbench (testbench.v) (1)
```

X20366-030318

Figure 5: **Design Files**

The reference design uses four cores:

- Tx_Data_Interface - High Speed SelectIO Wizard for transmitter setup of TX_BITSLICE
- Rx_Async_Interface - High Speed SelectIO Wizard for receiver setup of RX_BITSLICE
- vio_0 - Hardware interface to control resets and for getting status of reference design
- ila_0 - Hardware debug interface for capturing data during operation

The constraint file has additional settings for the specific demonstration board being targeted and might not be needed for all boards. The High Speed SelectIO Wizard provides additional constraints such as placement and select I/O settings.

The phase alignment algorithm requires RIU access to the BITSLICE_CONTROL, which is why the RX and TX interfaces must be kept in different byte groups and the design can be used without any changes. For designs that must place the RX and TX interfaces within the same byte group, the RIU_OR must be used within the `Byte_TopWizard_RxTx.vhd` design file. The RIU_OR has been commented out in the `Byte_TopWizard_RxTx.vhd` design file. Because the RIU_OR has been removed, the transmitter and receiver can be placed in different banks.

While changing pinouts for any device is simplified by using the wizard and using the same signal names for the ports, a number of interface signals need to be updated based on the instantiation template. The port names can be seen in the instantiation template for the given IP source. For VHDL, the component definition and the instantiation need to be updated.

These are the RX port names based on byte group:

```
riu_addr_bg<#>          => IntRiu_Addr,                  -- in [5:0]
riu_wr_data_bg<#>       => IntRiu_WrData,                -- in [15:0]
riu_rd_data_bg<#>       => IntBase_Riu_Rd_Data_Rx,       -- out [15:0]
riu_valid_bg<#>         => IntBase_Riu_Valid_Rx,         -- out
riu_wr_en_bg<#>         => IntRiu_Wr_En,                 -- in
riu_nibble_sel_bg<#>    => IntRiu_Nibble_Sel,            -- in [1:0]
```

These are the RX port names based on nibble (bitslice control):

```
dly_rdy_bsc<#>          => IntBase_Rx_Dly_Rdy,           -- out
```

These are the RX port names based on bitslice:

```
rx_en_vtc<#>            => IntBase_Rx_Bs_En_Vtc,         -- in
fifo_rd_clk<#>          => IntBase_Rx_Fifo_Rd_Clk,       -- in
fifo_rd_en<#>           => IntBase_Rx_Fifo_Rd_En_0(1),   -- in
fifo_empty<#>           => IntBase_Rx_Fifo_Empty_0(1),   -- out
rx_ce<#>                => Low,                          -- in
rx_inc<#>               => Low,                          -- in
rx_load<#>              => IntBase_Idly_Load_0(1),       -- in
rx_cntvaluein<#>        => IntBase_Idly_CntValueIn_0(17 downto 9),    -- in [8:0]
rx_cntvalueout<#>       => IntBase_Idly_CntValueOut_0(17 downto 9),   -- out [8:0]
```

These are the TX port names based on nibble (bitslice control)

```
vtc_rdy_bsc<#>          => open,                         --: OUT STD_LOGIC;
en_vtc_bsc<#>           => IntBase_Tx_Bsc_En_Vtc,        --: IN STD_LOGIC;
dly_rdy_bsc<#>          => IntBase_Tx_Dly_Rdy_bsc4,      --: OUT STD_LOGIC;
```

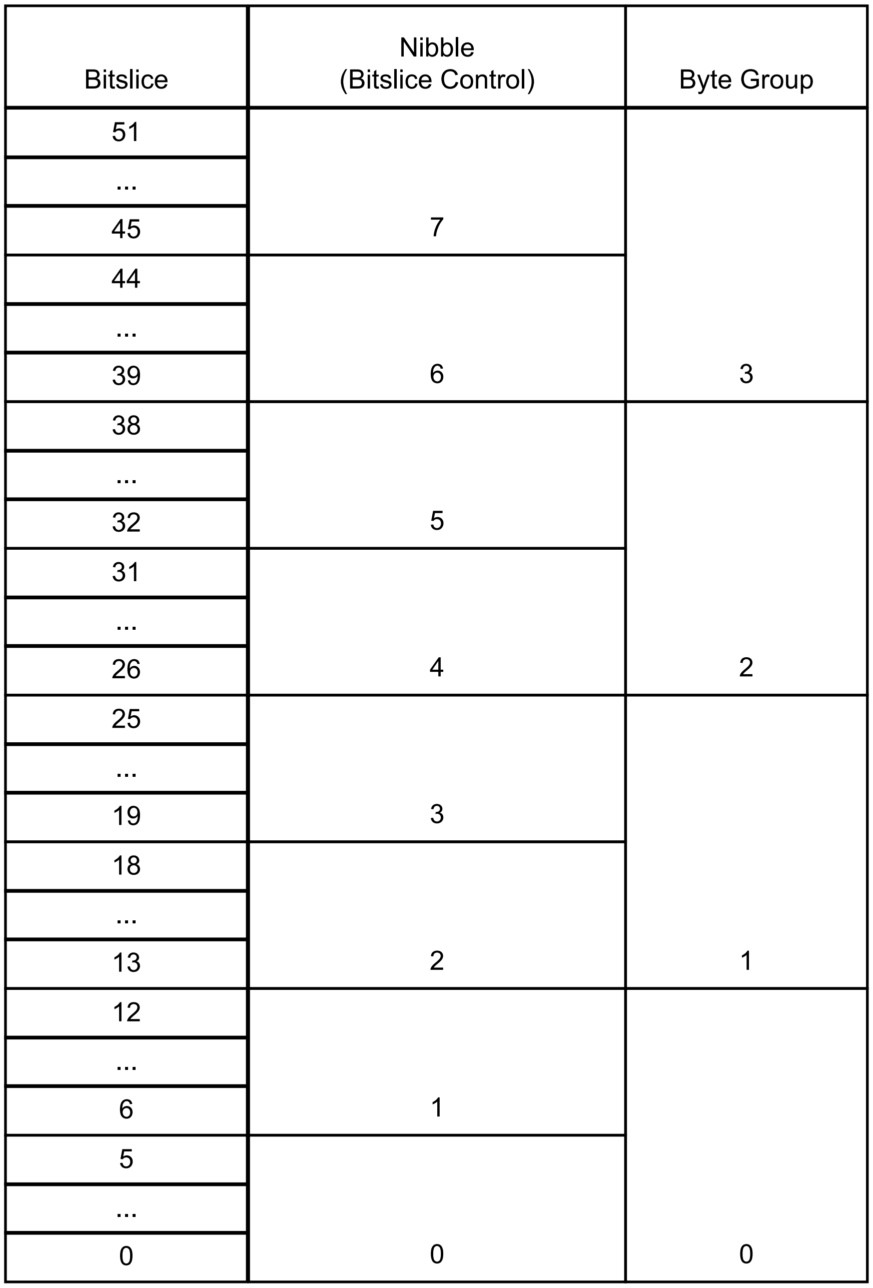
For the TX ports, if the design spans multiple nibbles, the dly_rdy_bsc<#> for each of the nibbles should be combined using the Stat_Tx_Dly_Rdy (`Byte_TopWizard_RxTx.vhd`). For example, if the TX interface spans nibble 0 and nibble 1, the Tx_Data_Interface should be adjusted as follows for dly_rdy_bsc# connections:

```
Stat_Tx_Dly_Rdy     <= IntBase_Tx_Dly_Rdy_bsc0 and IntBase_Tx_Dly_Rdy_bsc1;
Byte_TopWizard_RxTx_I_Tx_Data_Intrfce : Tx_Data_Interface
   port map (
       Tx_ dly_rdy_bsc0 => IntBase_Tx_Dly_Rdy_bsc0,
       Tx_ dly_rdy_bsc1 => IntBase_Tx_Dly_Rdy_bsc1,
       …
   )
```

The following lists pin selection guidelines for simplifying RX and TX designs:

- Each byte group must be either RX or TX.
- Empty byte groups are allowed. For example, an interface can use byte group 0 and 2.
- Byte groups spanning each interface must be either RX or TX. If byte groups 0 and 2 are used for the RX, byte group 1 must either be an RX or left empty.

Figure 6 shows how the bitslices are organized within a bank.

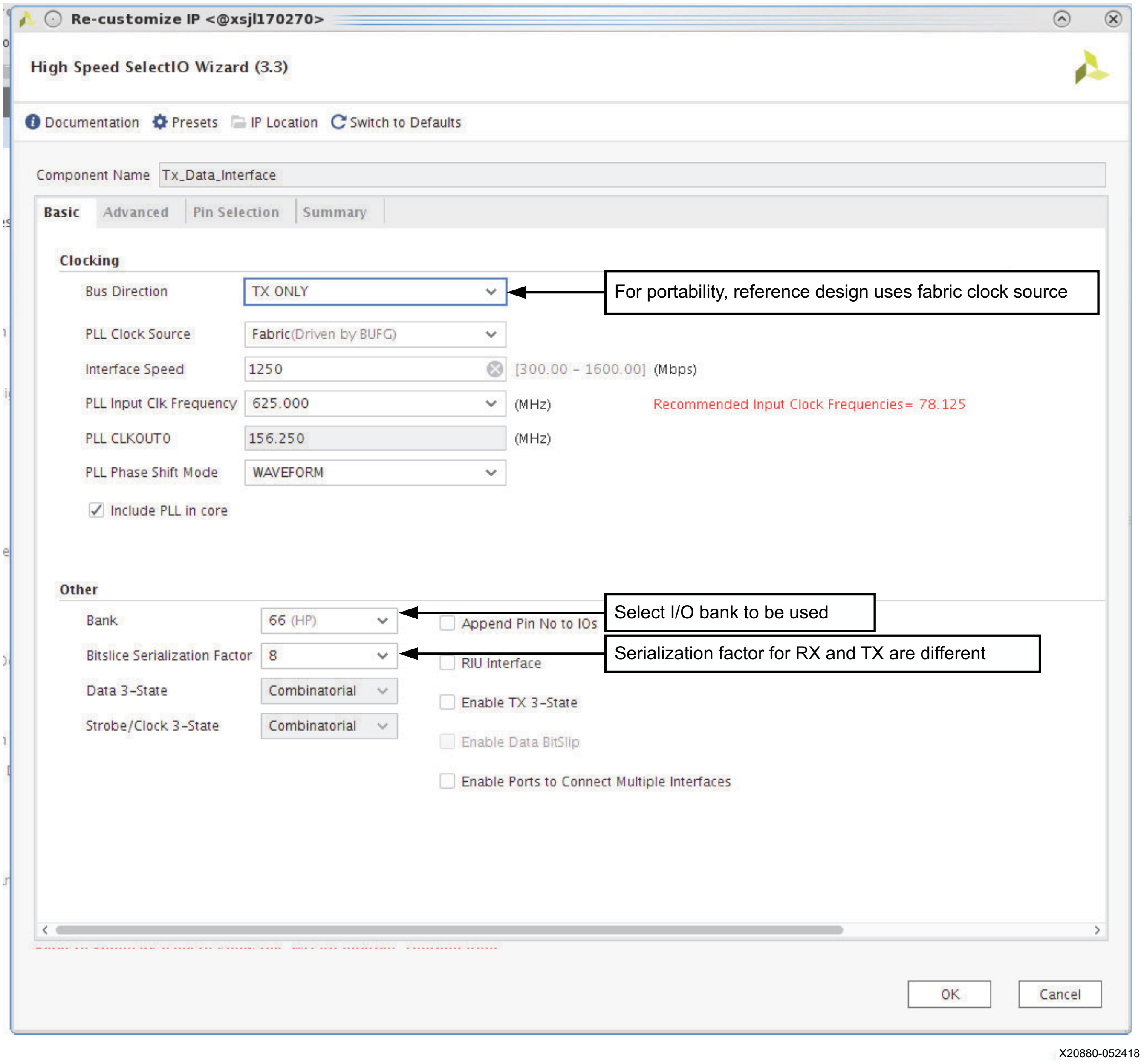


Bitslice	Nibble (Bitslice Control)	Byte Group
51		
...		
45	7	
44		
...		
39	6	3
38		
...		
32	5	
31		
...		
26	4	2
25		
...		
19	3	
18		
...		
13	2	1
12		
...		
6	1	
5		
...		
0	0	0

X20368-030318

Figure 6: **Bitslice, Nibble, and Byte Group Number within a Bank**

For the reference design to work, the TX and RX wizard settings are shown in Figure 7 to Figure 11.


Re-customize IP <@xsjl170270>

High Speed SelectIO Wizard (3.3)

Component Name: Tx_Data_Interface

Clocking

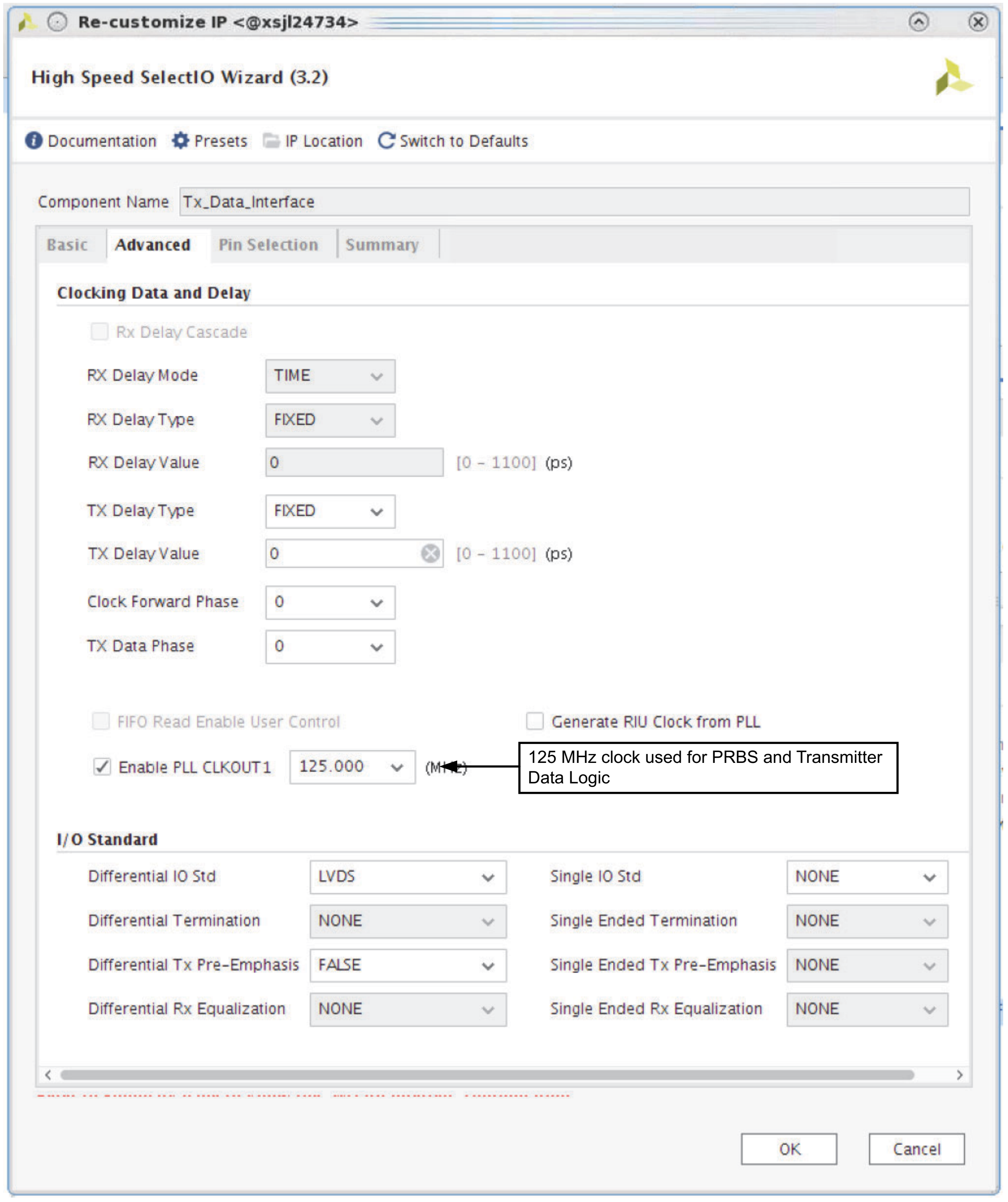
- Bus Direction: TX ONLY
- PLL Clock Source: Fabric(Driven by BUFG)
- Interface Speed: 1250 [300.00 - 1600.00] (Mbps)
- PLL Input Clk Frequency: 625.000 (MHz) Recommended Input Clock Frequencies= 78.125
- PLL CLKOUT0: 156.250 (MHz)
- PLL Phase Shift Mode: WAVEFORM
- [✓] Include PLL in core

Other

- Bank: 66 (HP)
- Bitslice Serialization Factor: 8
- Data 3-State: Combinatorial
- Strobe/Clock 3-State: Combinatorial
- [ ] Append Pin No to IOs
- [ ] RIU Interface
- [ ] Enable TX 3-State
- [ ] Enable Data BitSlip
- [ ] Enable Ports to Connect Multiple Interfaces

Documentation
Presets
IP Location
Switch to Defaults
Basic
Advanced
Pin Selection
Summary
For portability, reference design uses fabric clock source
Select I/O bank to be used
Serialization factor for RX and TX are different
OK
Cancel
X20880-052418

Figure 7: **High Speed SelectIO Wizard TX Setup - Basic Tab**


Re-customize IP <@xsjl24734>

High Speed SelectIO Wizard (3.2)

Documentation | Presets | IP Location | Switch to Defaults

Component Name: Tx_Data_Interface

Basic | **Advanced** | Pin Selection | Summary

Clocking Data and Delay

☐ Rx Delay Cascade

RX Delay Mode	TIME	
RX Delay Type	FIXED	
RX Delay Value	0	[0 - 1100] (ps)
TX Delay Type	FIXED	
TX Delay Value	0	[0 - 1100] (ps)
Clock Forward Phase	0	
TX Data Phase	0	

☐ FIFO Read Enable User Control

☐ Generate RIU Clock from PLL

✓ Enable PLL CLKOUT1 125.000 (MHz)

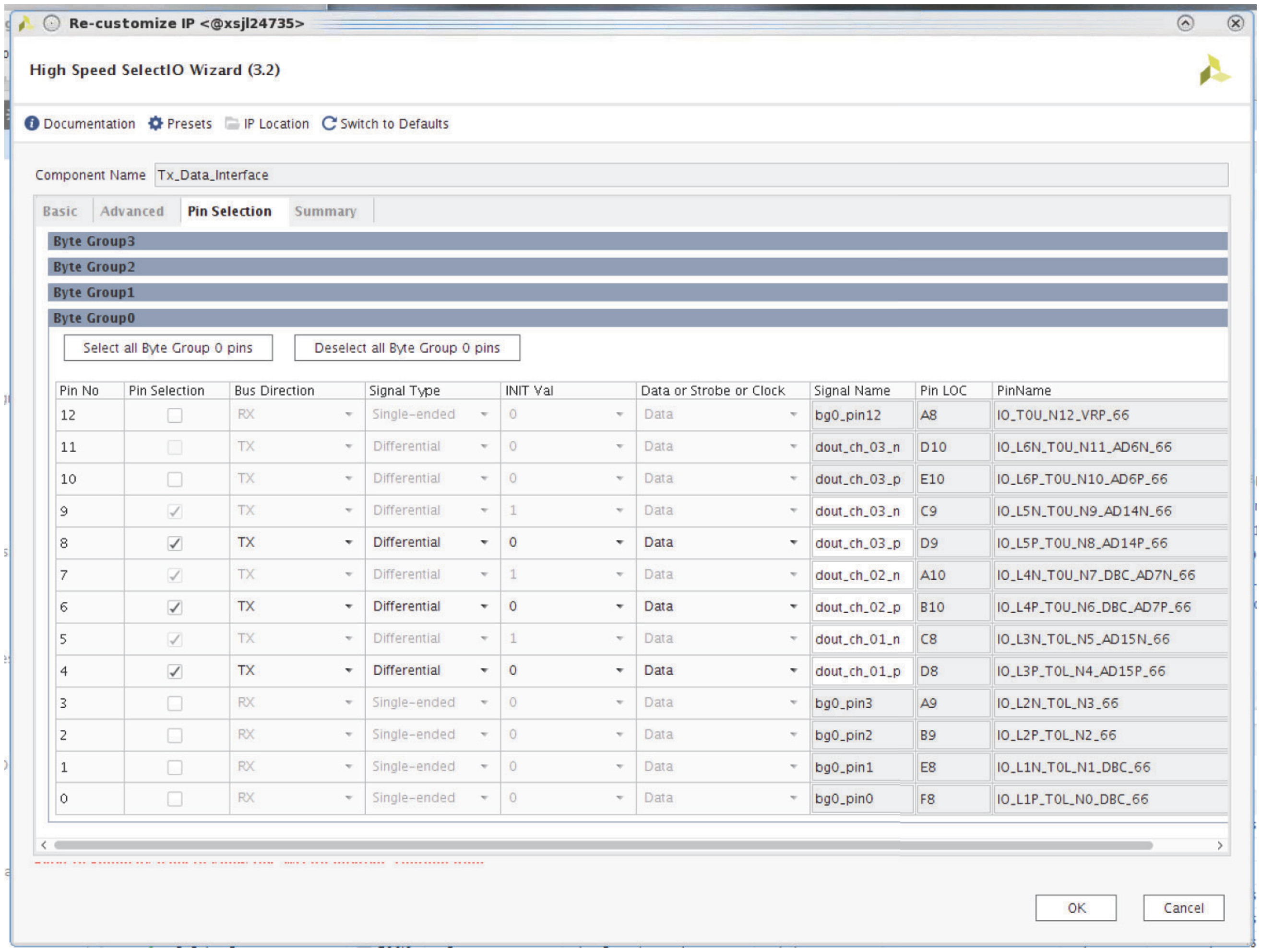
125 MHz clock used for PRBS and Transmitter Data Logic

I/O Standard

Differential IO Std	LVDS	Single IO Std	NONE
Differential Termination	NONE	Single Ended Termination	NONE
Differential Tx Pre-Emphasis	FALSE	Single Ended Tx Pre-Emphasis	NONE
Differential Rx Equalization	NONE	Single Ended Rx Equalization	NONE

X20370-052418

Figure 8: **High Speed SelectIO Wizard TX Setup - Advanced Tab**



Re-customize IP <@xsjl24735>

High Speed SelectIO Wizard (3.2)

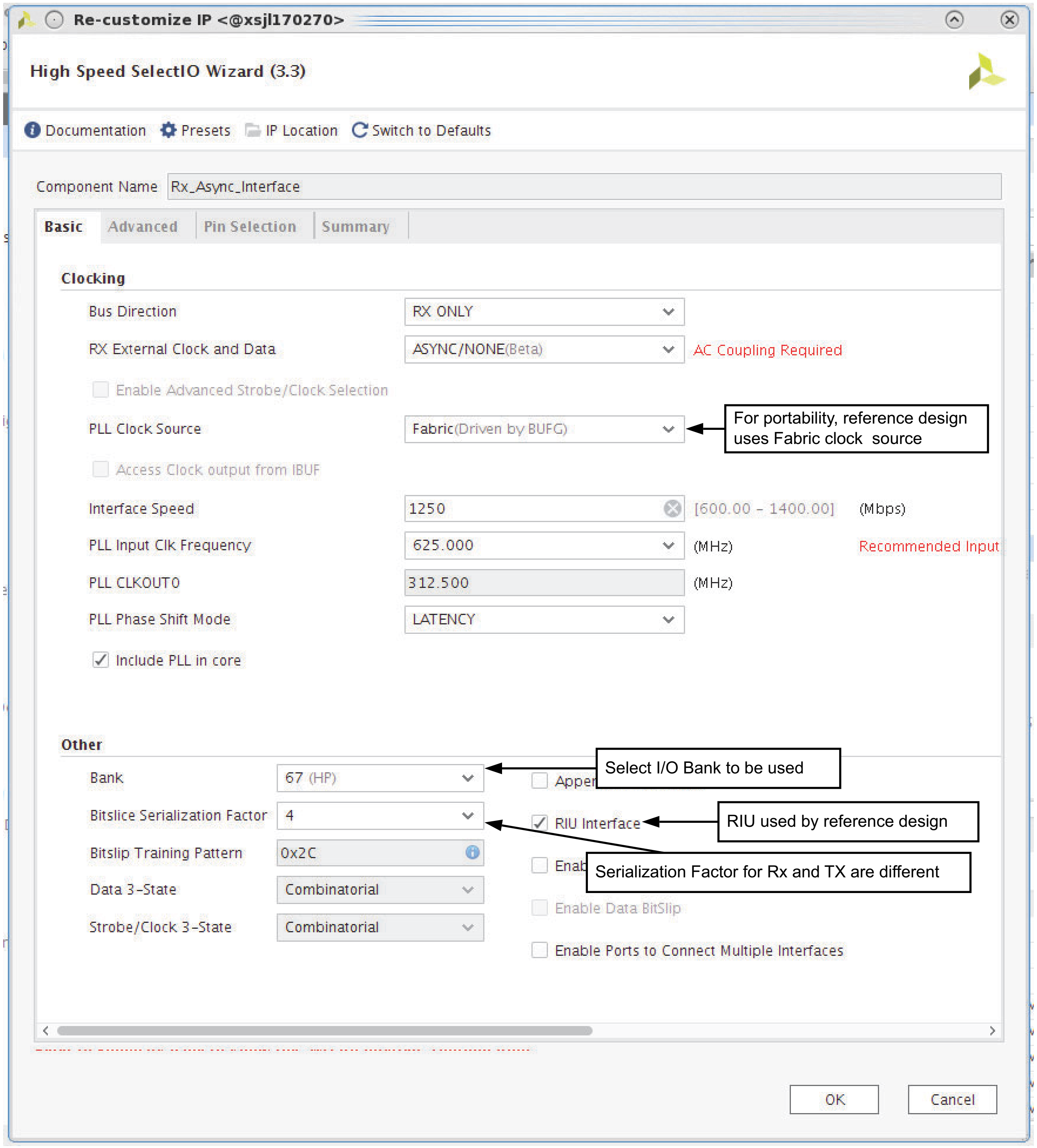
Component Name Tx_Data_Interface

Pin No	Pin Selection	Bus Direction	Signal Type	INIT Val	Data or Strobe or Clock	Signal Name	Pin LOC	PinName
12	☐	RX	Single-ended	0	Data	bg0_pin12	A8	IO_T0U_N12_VRP_66
11	☐	TX	Differential	0	Data	dout_ch_03_n	D10	IO_L6N_T0U_N11_AD6N_66
10	☐	TX	Differential	0	Data	dout_ch_03_p	E10	IO_L6P_T0U_N10_AD6P_66
9	✓	TX	Differential	1	Data	dout_ch_03_n	C9	IO_L5N_T0U_N9_AD14N_66
8	✓	TX	Differential	0	Data	dout_ch_03_p	D9	IO_L5P_T0U_N8_AD14P_66
7	✓	TX	Differential	1	Data	dout_ch_02_n	A10	IO_L4N_T0U_N7_DBC_AD7N_66
6	✓	TX	Differential	0	Data	dout_ch_02_p	B10	IO_L4P_T0U_N6_DBC_AD7P_66
5	✓	TX	Differential	1	Data	dout_ch_01_n	C8	IO_L3N_T0L_N5_AD15N_66
4	✓	TX	Differential	0	Data	dout_ch_01_p	D8	IO_L3P_T0L_N4_AD15P_66
3	☐	RX	Single-ended	0	Data	bg0_pin3	A9	IO_L2N_T0L_N3_66
2	☐	RX	Single-ended	0	Data	bg0_pin2	B9	IO_L2P_T0L_N2_66
1	☐	RX	Single-ended	0	Data	bg0_pin1	E8	IO_L1N_T0L_N1_DBC_66
0	☐	RX	Single-ended	0	Data	bg0_pin0	F8	IO_L1P_T0L_N0_DBC_66

X20371-030318

Figure 9: **High Speed SelectIO Wizard TX Setup - Pin Selection Tab**

The settings shown in Figure 10, Figure 11, and Figure 12 should be used for the RX.



Component Name: Rx_Async_Interface

Clocking

Bus Direction	RX ONLY	
RX External Clock and Data	ASYNC/NONE(Beta)	AC Coupling Required
☐ Enable Advanced Strobe/Clock Selection		
PLL Clock Source	Fabric(Driven by BUFG)	For portability, reference design uses Fabric clock source
☐ Access Clock output from IBUF		
Interface Speed	1250	[600.00 - 1400.00] (Mbps)
PLL Input Clk Frequency	625.000	(MHz) Recommended Input
PLL CLKOUT0	312.500	(MHz)
PLL Phase Shift Mode	LATENCY	
✓ Include PLL in core		

Other

Bank	67 (HP)	Select I/O Bank to be used
Bitslice Serialization Factor	4	Serialization Factor for Rx and TX are different
Bitslip Training Pattern	0x2C	
Data 3-State	Combinatorial	
Strobe/Clock 3-State	Combinatorial	

☐ Apper

✓ RIU Interface — RIU used by reference design

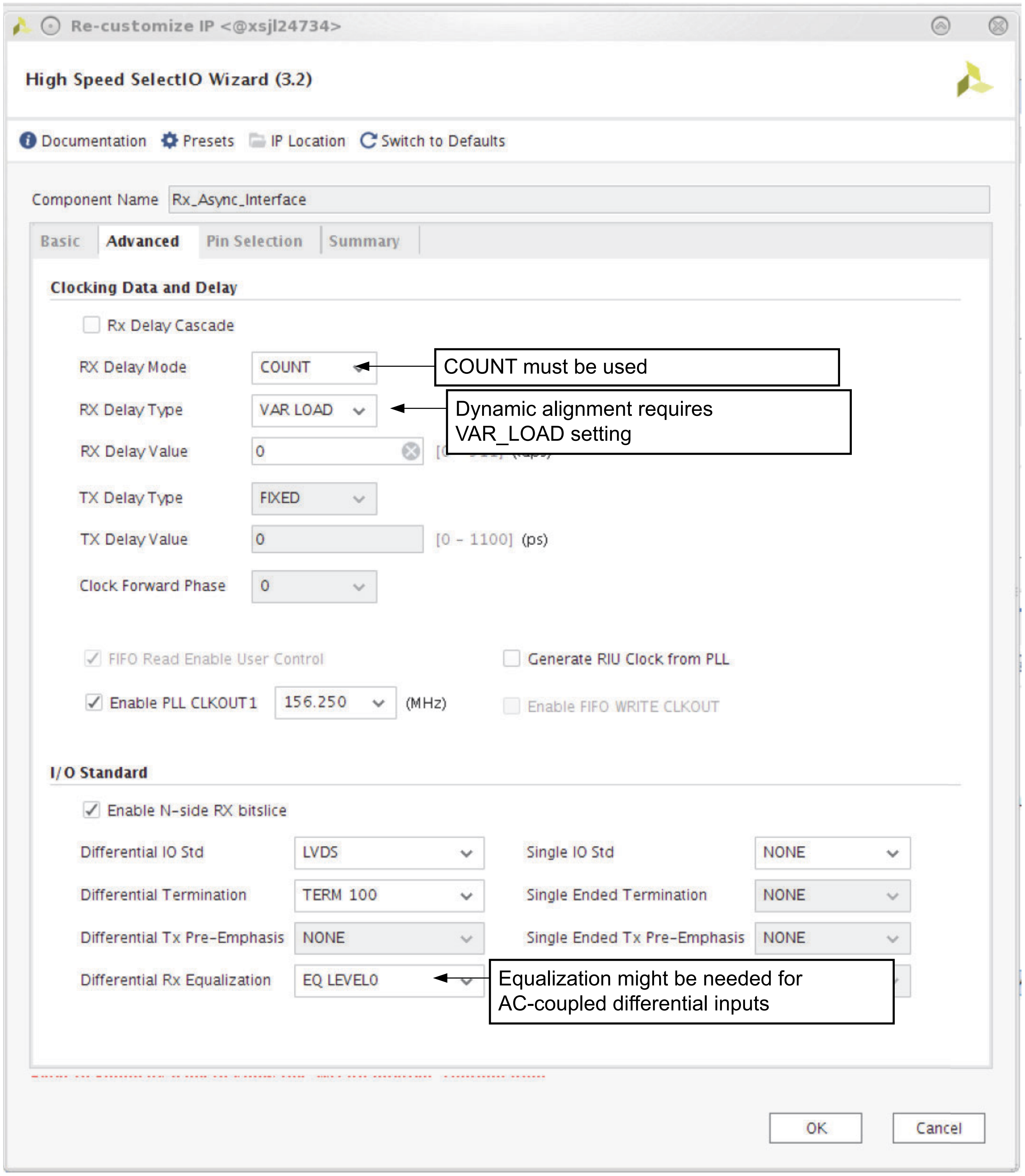
☐ Enab

☐ Enable Data BitSlip

☐ Enable Ports to Connect Multiple Interfaces

X20881-052418

Figure 10: **High Speed SelectIO Wizard RX Setup - Basic Tab**



Re-customize IP <@xsjl24734>

High Speed SelectIO Wizard (3.2)

Documentation | Presets | IP Location | Switch to Defaults

Component Name: Rx_Async_Interface

Basic | **Advanced** | Pin Selection | Summary

Clocking Data and Delay

- [ ] Rx Delay Cascade
- RX Delay Mode: COUNT
- RX Delay Type: VAR LOAD
- RX Delay Value: 0
- TX Delay Type: FIXED
- TX Delay Value: 0 [0 - 1100] (ps)
- Clock Forward Phase: 0
- [✓] FIFO Read Enable User Control
- [ ] Generate RIU Clock from PLL
- [✓] Enable PLL CLKOUT1: 156.250 (MHz)
- [ ] Enable FIFO WRITE CLKOUT

I/O Standard

- [✓] Enable N-side RX bitslice

Differential IO Std	LVDS	Single IO Std	NONE
Differential Termination	TERM 100	Single Ended Termination	NONE
Differential Tx Pre-Emphasis	NONE	Single Ended Tx Pre-Emphasis	NONE
Differential Rx Equalization	EQ LEVEL0		

COUNT must be used

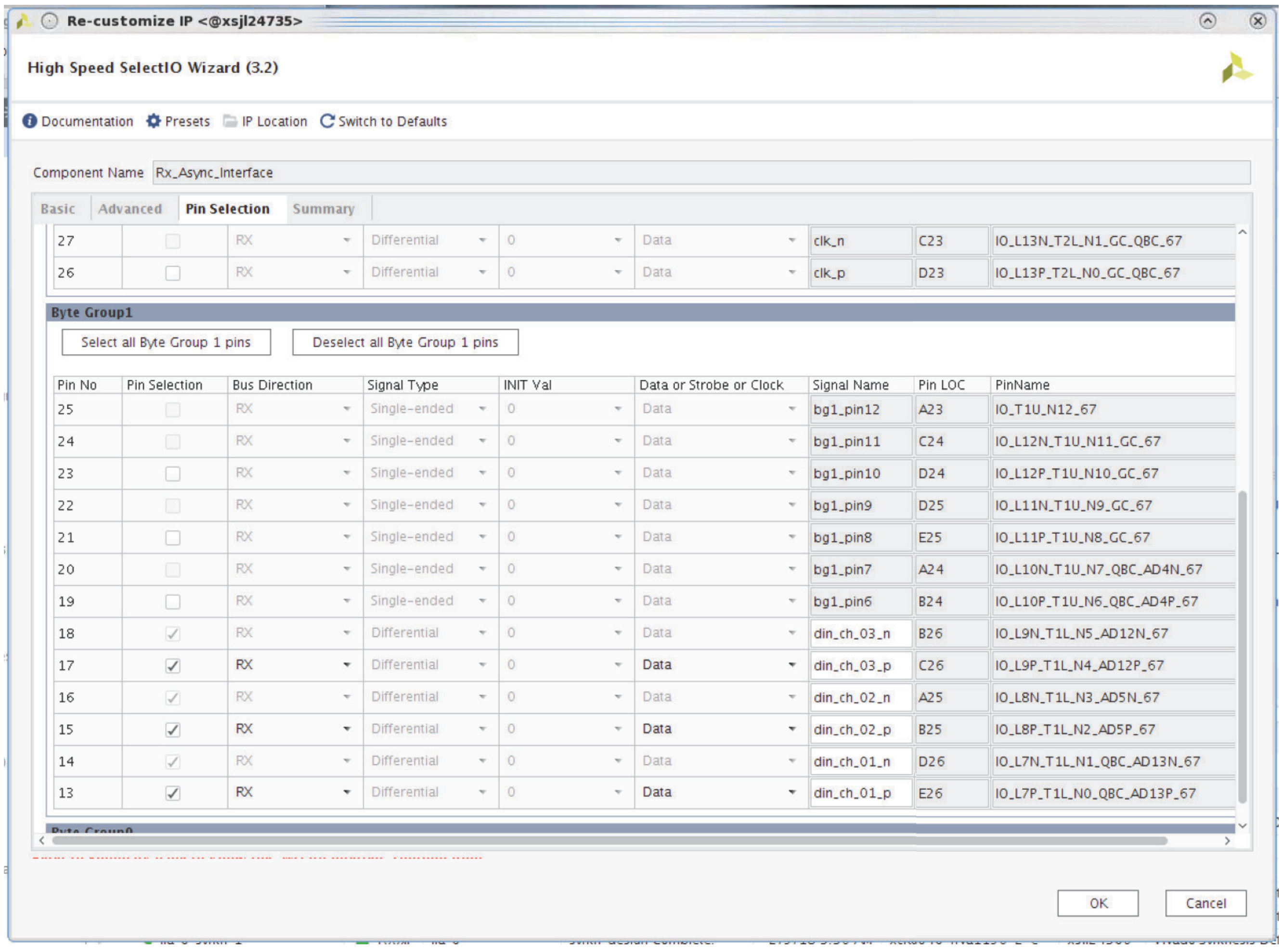
Dynamic alignment requires VAR_LOAD setting

Equalization might be needed for AC-coupled differential inputs

OK | Cancel

X20373-030518

Figure 11: **High Speed SelectIO Wizard RX Setup - Advanced Tab**



Re-customize IP <@xsjl24735>

High Speed SelectIO Wizard (3.2)

Documentation | Presets | IP Location | Switch to Defaults

Component Name: Rx_Async_Interface

Basic | Advanced | **Pin Selection** | Summary

27	☐	RX	Differential	0	Data	clk_n	C23	IO_L13N_T2L_N1_GC_QBC_67
26	☐	RX	Differential	0	Data	clk_p	D23	IO_L13P_T2L_N0_GC_QBC_67

Byte Group1

Select all Byte Group 1 pins | Deselect all Byte Group 1 pins

Pin No	Pin Selection	Bus Direction	Signal Type	INIT Val	Data or Strobe or Clock	Signal Name	Pin LOC	PinName
25	☐	RX	Single-ended	0	Data	bg1_pin12	A23	IO_T1U_N12_67
24	☐	RX	Single-ended	0	Data	bg1_pin11	C24	IO_L12N_T1U_N11_GC_67
23	☐	RX	Single-ended	0	Data	bg1_pin10	D24	IO_L12P_T1U_N10_GC_67
22	☐	RX	Single-ended	0	Data	bg1_pin9	D25	IO_L11N_T1U_N9_GC_67
21	☐	RX	Single-ended	0	Data	bg1_pin8	E25	IO_L11P_T1U_N8_GC_67
20	☐	RX	Single-ended	0	Data	bg1_pin7	A24	IO_L10N_T1U_N7_QBC_AD4N_67
19	☐	RX	Single-ended	0	Data	bg1_pin6	B24	IO_L10P_T1U_N6_QBC_AD4P_67
18	✓	RX	Differential	0	Data	din_ch_03_n	B26	IO_L9N_T1L_N5_AD12N_67
17	✓	RX	Differential	0	Data	din_ch_03_p	C26	IO_L9P_T1L_N4_AD12P_67
16	✓	RX	Differential	0	Data	din_ch_02_n	A25	IO_L8N_T1L_N3_AD5N_67
15	✓	RX	Differential	0	Data	din_ch_02_p	B25	IO_L8P_T1L_N2_AD5P_67
14	✓	RX	Differential	0	Data	din_ch_01_n	D26	IO_L7N_T1L_N1_QBC_AD13N_67
13	✓	RX	Differential	0	Data	din_ch_01_p	E26	IO_L7P_T1L_N0_QBC_AD13P_67

X20374-052418

Figure 12: **High Speed SelectIO Wizard RX Setup - Pin Selection Tab**

To allow the reference clock to be moved to different banks, an extra clock buffer (BUFGCE) has been added as shown in Figure 13. To free up the clock routes, the CLOCK_DEDICATED_ROUTE has been added. In this specific case, the clock is only routed to the PLLs and a divided clock, which is used as a control clock that does not require any phase relationship to the input pins.

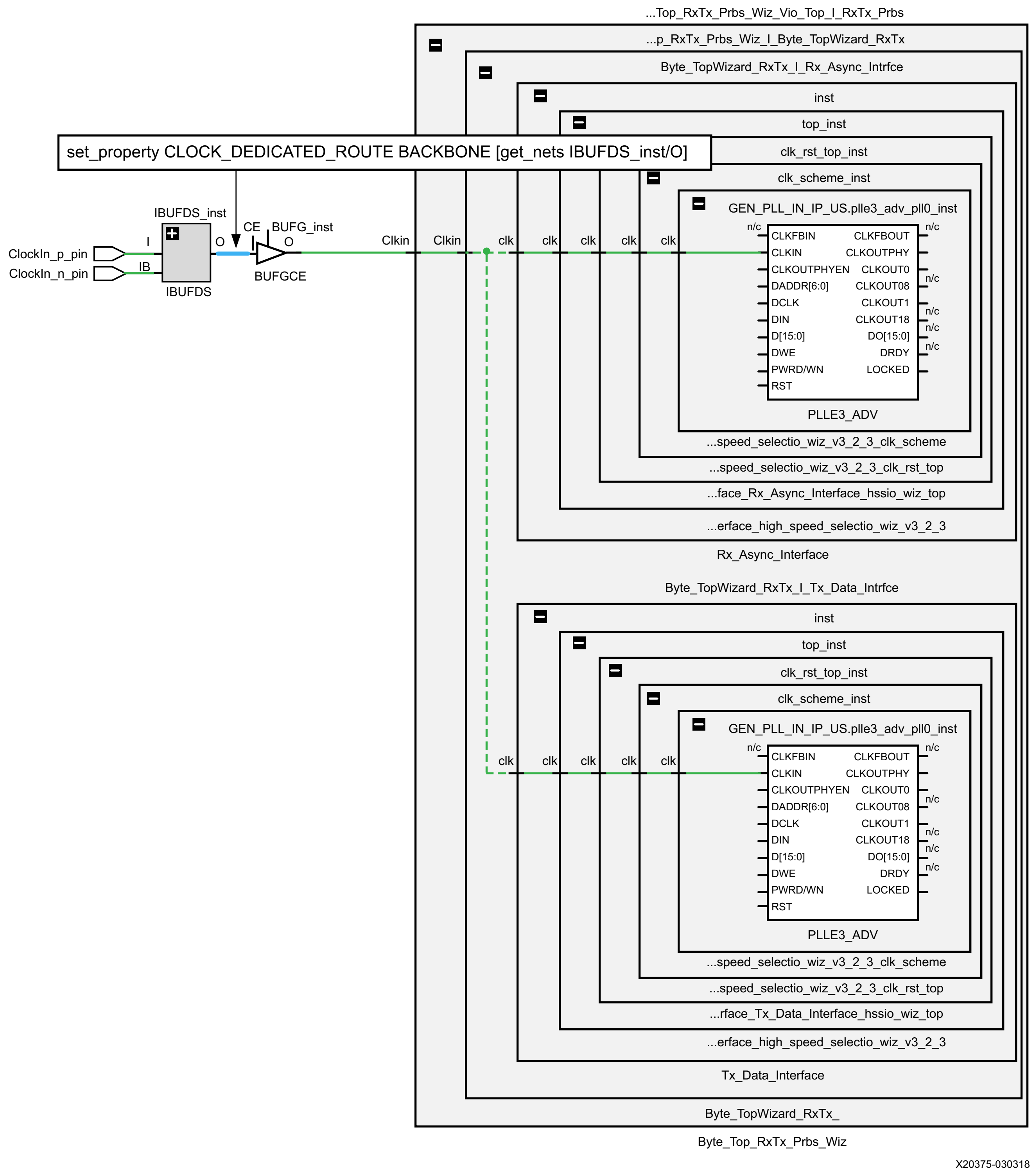

set_property CLOCK_DEDICATED_ROUTE BACKBONE [get_nets IBUFDS_inst/O]
ClockIn_p_pin
ClockIn_n_pin
IBUFDS_inst
IBUFDS
BUFG_inst
BUFGCE
CE
Clkin
clk
...Top_RxTx_Prbs_Wiz_Vio_Top_I_RxTx_Prbs
...p_RxTx_Prbs_Wiz_I_Byte_TopWizard_RxTx
Byte_TopWizard_RxTx_I_Rx_Async_Intrfce
inst
top_inst
clk_rst_top_inst
clk_scheme_inst
GEN_PLL_IN_IP_US.plle3_adv_pll0_inst
CLKFBIN
CLKIN
CLKOUTPHYEN
DADDR[6:0]
DCLK
DIN
D[15:0]
DWE
PWRD/WN
RST
CLKFBOUT
CLKOUTPHY
CLKOUT0
CLKOUT08
CLKOUT1
CLKOUT18
DO[15:0]
DRDY
LOCKED
n/c
PLLE3_ADV
...speed_selectio_wiz_v3_2_3_clk_scheme
...speed_selectio_wiz_v3_2_3_clk_rst_top
...face_Rx_Async_Interface_hssio_wiz_top
...erface_high_speed_selectio_wiz_v3_2_3
Rx_Async_Interface
Byte_TopWizard_RxTx_I_Tx_Data_Intrfce
...rface_Tx_Data_Interface_hssio_wiz_top
Tx_Data_Interface
Byte_TopWizard_RxTx_
Byte_Top_RxTx_Prbs_Wiz
X20375-030318

Figure 13: **Reference Clock Connections to PLLs**

Simulation

A simple testbench has been provided. The reference design uses registers that are not supported by simulation. For proper simulation, set C_InSimulation = "true" in the top-level design file `Byte_Top_RxTx_Prbs_Wiz_Vio_Top.vhd`.

```
Byte_Top_RxTx_Prbs_Wiz_Vio_Top_I_RxTx_Prbs : entity
xil_defaultlib.Byte_Top_RxTx_Prbs_Wiz
    generic map (
        C_InSimulation              => "true",
        C_SimDevice                 => "ULTRASCALE"
    )
```

The reference design accesses a reserved BITSLICE_CONTROL RIU that is not supported in simulation. In `Riu_StateMach.vhd`, when C_InSimulation = "true", Rx_BtVal is set to 160 (`0xA0`). C_InSimulation = "true" fixes the Rx_BtVal in post-synthesis and post-implementation simulations.

After simulation is complete, set C_InSimulation = "false" to ensure the Rx_BtVal is being read from the BITSLICE_CONTROL. The design needs to be re-synthesized afterwards so that the Rx_BtVal is read from the Rx_RdData connected to the BITSLICE_CONTROL.

To simplify the RIU state machine and allow for more portability, the bit time for `Rx_BtVal` is used as the bit time for all of the receivers, because each receiver should be running at the same data rate. In `Byte_TopWizard_RxTx.vhd`, Int_Base_Rx_BtVal is used for each BaseX_TxRxLogic instantiation.

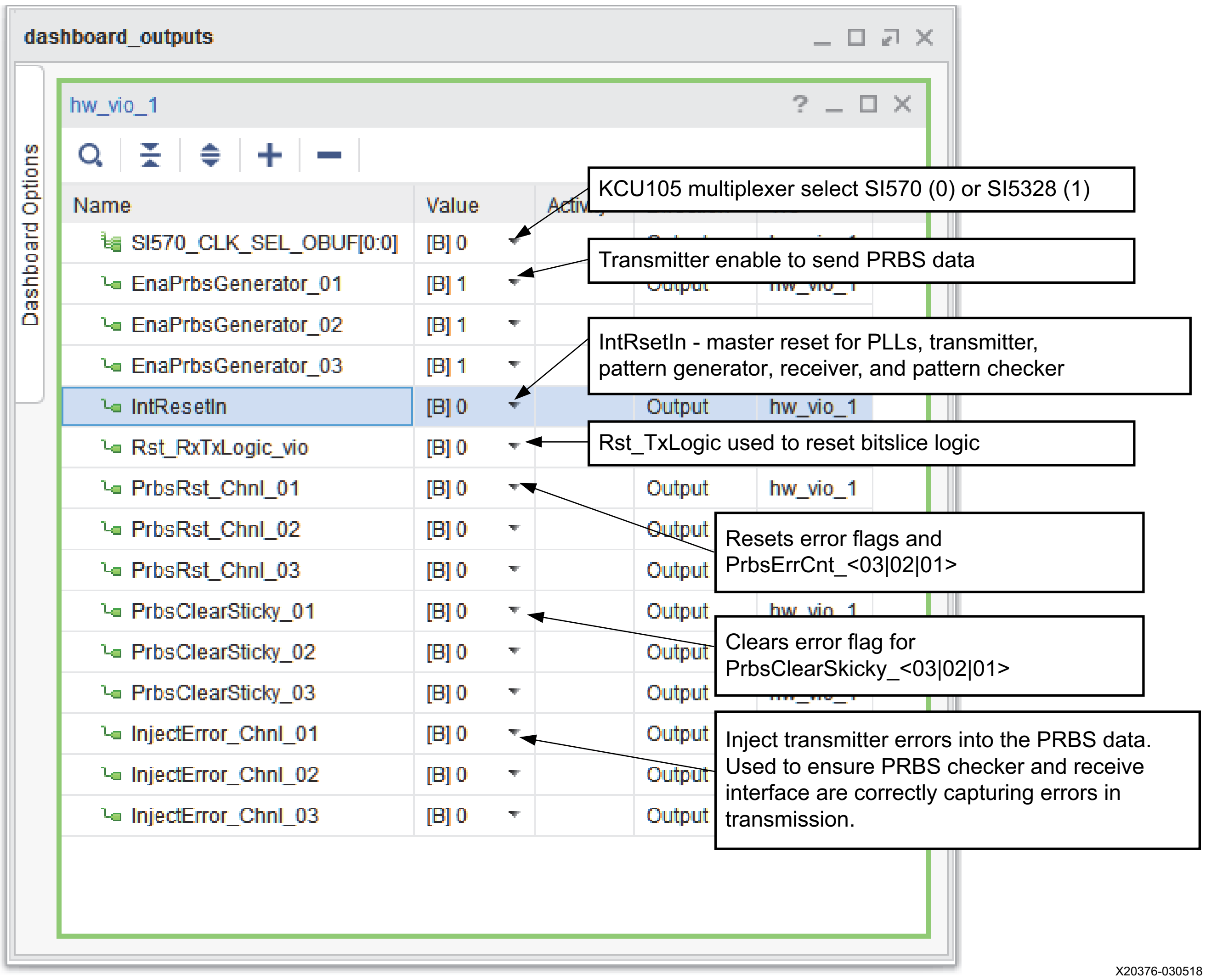
Hardware Operation

The reference design has been set up for the KCU105 board using the FMC HPC connector. Attach the XM107 FMC loopback card to the J22 FMC connector. The XM107 connects LA[16:0] to LA[33:17]. See the *FMC XM107 Loopback Card User Guide* (UG539) [Ref 2].

The design requires two clock sources. The VIO interface is directly connected to the 100 MHz clock to control the VIO interface and generate the input/output signals for controlling the design.

The transmitter and receiver channels require a 625 MHz clock which is connected to the Si570 when SI570_CLK_SEL = 1. To control the clocks, use the serial port as described in the *KCU105 Board User Guide* (UG197) [Ref 1] using the system controller. By using the serial port, the system controller can set the KCU105 Si570 user clock frequency to 625 MHz.

For some systems the FMC might also need to be powered using the serial port. Follow the instructions in the board user guide, for example, *KCU105 Board User Guide* (UG917). Hardware operation should follow the reset sequence using the VIO outputs, as shown in Figure 14.



Name	Value	Activ		
SI570_CLK_SEL_OBUF[0:0]	[B] 0			
EnaPrbsGenerator_01	[B] 1		Output	hw_vio_1
EnaPrbsGenerator_02	[B] 1			
EnaPrbsGenerator_03	[B] 1			
IntResetIn	[B] 0		Output	hw_vio_1
Rst_RxTxLogic_vio	[B] 0			
PrbsRst_Chnl_01	[B] 0		Output	hw_vio_1
PrbsRst_Chnl_02	[B] 0		Output	
PrbsRst_Chnl_03	[B] 0		Output	
PrbsClearSticky_01	[B] 0		Output	hw_vio_1
PrbsClearSticky_02	[B] 0		Output	
PrbsClearSticky_03	[B] 0		Output	hw_vio_1
InjectError_Chnl_01	[B] 0		Output	
InjectError_Chnl_02	[B] 0		Output	
InjectError_Chnl_03	[B] 0		Output	

KCU105 multiplexer select SI570 (0) or SI5328 (1)

Transmitter enable to send PRBS data

IntRsetIn - master reset for PLLs, transmitter, pattern generator, receiver, and pattern checker

Rst_TxLogic used to reset bitslice logic

Resets error flags and PrbsErrCnt_<03|02|01>

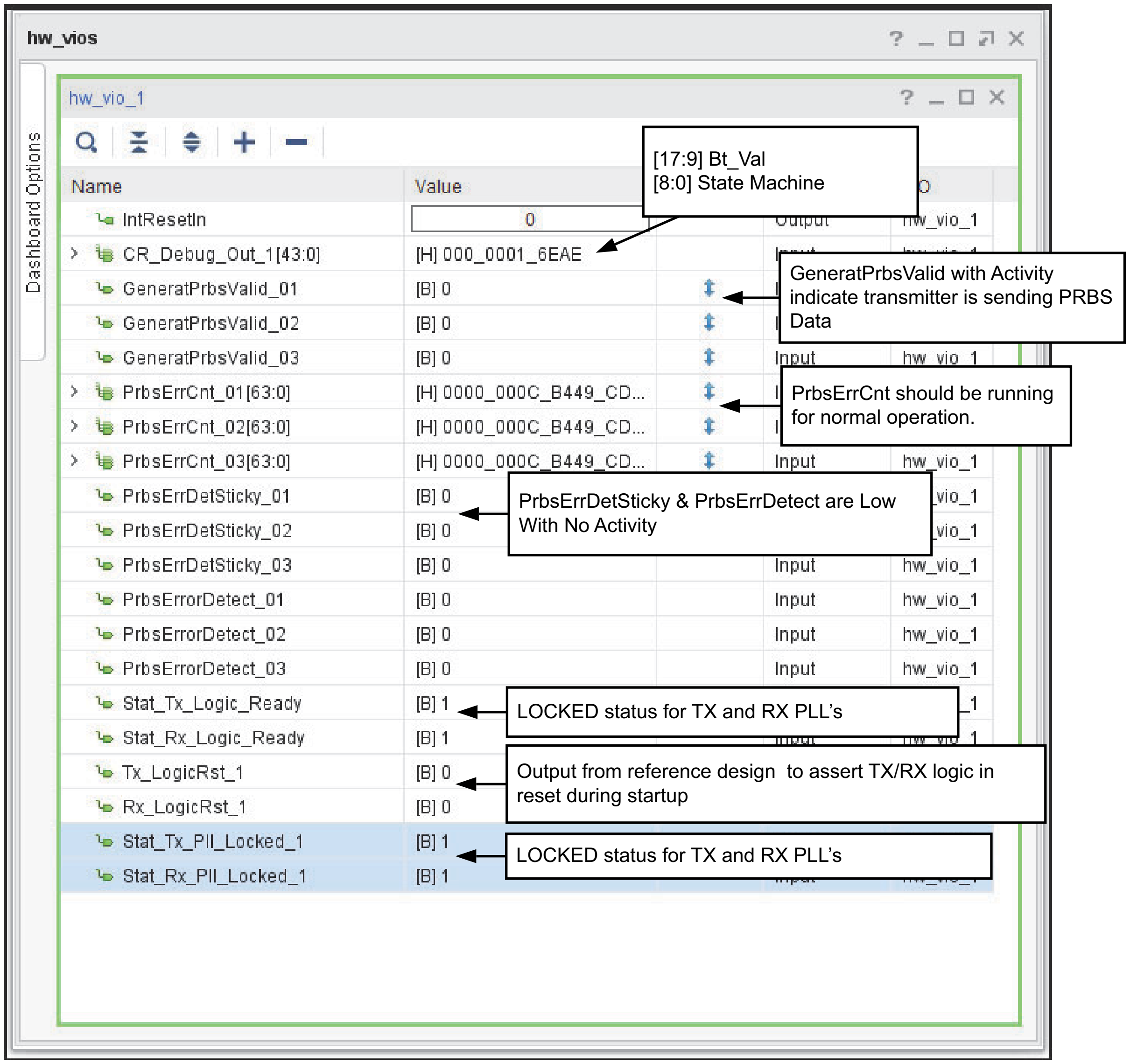
Clears error flag for PrbsClearSkicky_<03|02|01>

Inject transmitter errors into the PRBS data. Used to ensure PRBS checker and receive interface are correctly capturing errors in transmission.

Figure 14: **Virtual I/O (VIO) Outputs**

The reset sequence has been simplified to reset when IntResetIn has been asserted High. The reset and enable controls are still available, as shown in Figure 14.

When operating correctly, GernatePrbsValid_<03|02|01> and PrbsErrorDetec<03|02|01> continue to toggle, indicating the interface is running.



Name	Value			O
IntResetIn	0		Output	hw_vio_1
CR_Debug_Out_1[43:0]	[H] 000_0001_6EAE		Input	hw_vio_1
GeneratPrbsValid_01	[B] 0	↕	I	
GeneratPrbsValid_02	[B] 0	↕	I	
GeneratPrbsValid_03	[B] 0	↕	Input	hw_vio_1
PrbsErrCnt_01[63:0]	[H] 0000_000C_B449_CD...	↕	I	
PrbsErrCnt_02[63:0]	[H] 0000_000C_B449_CD...	↕	I	
PrbsErrCnt_03[63:0]	[H] 0000_000C_B449_CD...	↕	Input	hw_vio_1
PrbsErrDetSticky_01	[B] 0			vio_1
PrbsErrDetSticky_02	[B] 0			vio_1
PrbsErrDetSticky_03	[B] 0		Input	hw_vio_1
PrbsErrorDetect_01	[B] 0		Input	hw_vio_1
PrbsErrorDetect_02	[B] 0		Input	hw_vio_1
PrbsErrorDetect_03	[B] 0		Input	hw_vio_1
Stat_Tx_Logic_Ready	[B] 1			1
Stat_Rx_Logic_Ready	[B] 1		Input	hw_vio_1
Tx_LogicRst_1	[B] 0			
Rx_LogicRst_1	[B] 0			
Stat_Tx_Pll_Locked_1	[B] 1			
Stat_Rx_Pll_Locked_1	[B] 1		Input	hw_vio_1

[17:9] Bt_Val
[8:0] State Machine

GeneratPrbsValid with Activity indicate transmitter is sending PRBS Data

PrbsErrCnt should be running for normal operation.

PrbsErrDetSticky & PrbsErrDetect are Low With No Activity

LOCKED status for TX and RX PLL's

Output from reference design to assert TX/RX logic in reset during startup

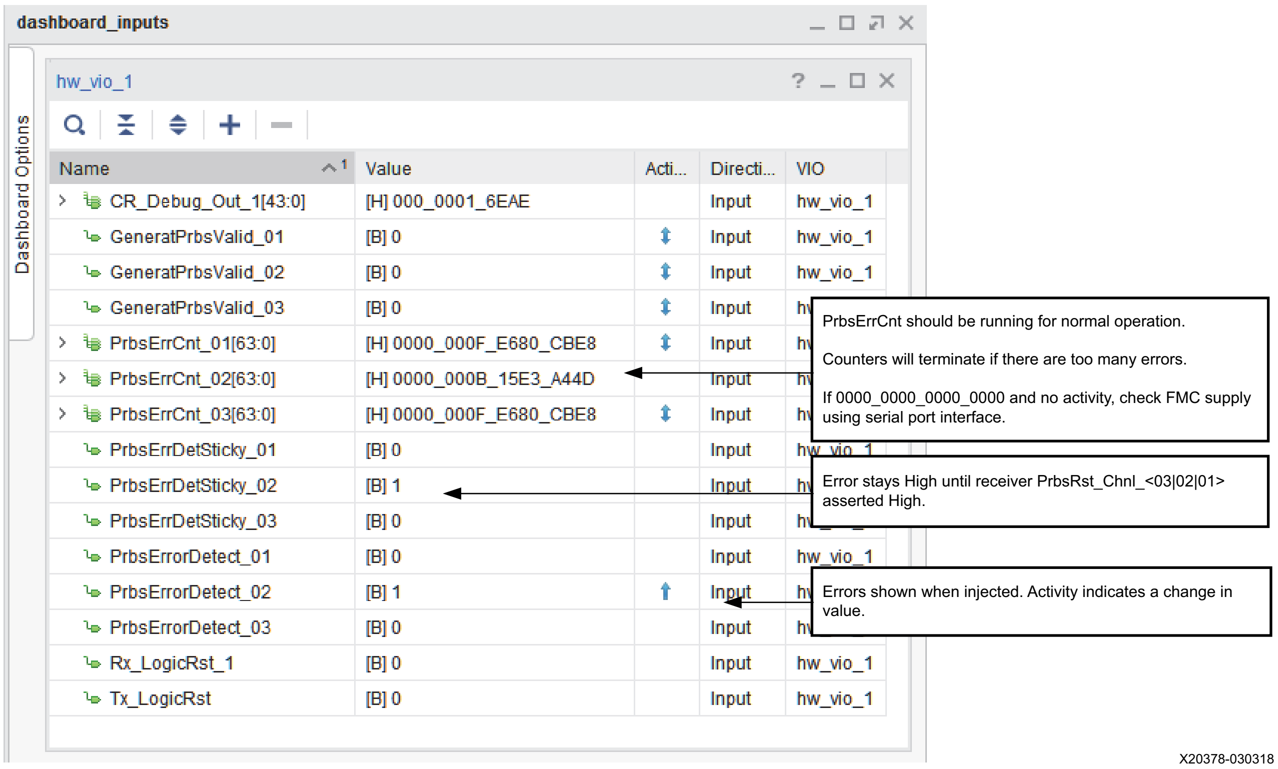
LOCKED status for TX and RX PLL's

X20882-052418

Figure 15: **Virtual I/O (VIO) Inputs**

Check PrbsErrDetSticky_<03|02|01> and PrbsErrCnt_<03|02|01> for errors.

To make sure the interface is working, use the InjectError_Chnl_<03|02|01> to inject an error on the given channel. As shown in Figure 16, PrbsErrDetSticky remains High if any errors occur. In the example, InjectError_Chnl_02 has been asserted High. When the error occurs, PrbsErrorDetect_02 momentarily goes High with the activity also indicating a change in values. When enough errors occur, PrbsErrCnt terminates with the activity stopping.



Name	Value	Acti...	Directi...	VIO
CR_Debug_Out_1[43:0]	[H] 000_0001_6EAE		Input	hw_vio_1
GeneratPrbsValid_01	[B] 0	↕	Input	hw_vio_1
GeneratPrbsValid_02	[B] 0	↕	Input	hw_vio_1
GeneratPrbsValid_03	[B] 0	↕	Input	hv
PrbsErrCnt_01[63:0]	[H] 0000_000F_E680_CBE8	↕	Input	hv
PrbsErrCnt_02[63:0]	[H] 0000_000B_15E3_A44D		Input	hv
PrbsErrCnt_03[63:0]	[H] 0000_000F_E680_CBE8	↕	Input	hv
PrbsErrDetSticky_01	[B] 0		Input	hw_vio_1
PrbsErrDetSticky_02	[B] 1		Input	hv
PrbsErrDetSticky_03	[B] 0		Input	hv
PrbsErrorDetect_01	[B] 0		Input	hw_vio_1
PrbsErrorDetect_02	[B] 1	↑	Input	hv
PrbsErrorDetect_03	[B] 0		Input	hv
Rx_LogicRst_1	[B] 0		Input	hw_vio_1
Tx_LogicRst	[B] 0		Input	hw_vio_1

PrbsErrCnt should be running for normal operation.

Counters will terminate if there are too many errors.

If 0000_0000_0000_0000 and no activity, check FMC supply using serial port interface.

Error stays High until receiver PrbsRst_Chnl_<03|02|01> asserted High.

Errors shown when injected. Activity indicates a change in value.

Figure 16: **Virtual I/O (VIO) Inputs After Injected Errors**

After the error has occurred, follow the reset sequence. PrbsClearSticky_<03|02|01> and PrbsRst_Chnl_<03|02|01> will clear the error flags and error counter.

Note: If PrbsErrCnt_<03|02|01> does not have any activity and is [H]0000_0000_0000_0000, check to make sure the FMC supplies are set. Check the user guide for the given board on how to set the FMC supply.

Conclusion

The asynchronous reference design works with the native primitives to control the RX_BITSLICE and BITSLICE_CONTROL primitives. As a result, when the pins are modified in the High Speed SelectIO Wizard, the port connections need to be updated. By following the guidelines in this application note, the reference design pinouts can be modified. Additional instructions on using the design have also been provided.

Documentation Navigator and Design Hubs

Xilinx Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

1. *KCU105 Board User Guide* (UG917)
2. *FMC XM107 Loopback Card User Guide* (UG539)
3. *LogiCORE IP High Speed SelectIO Wizard Product Guide* (PG188)
4. *UltraScale Architecture SelectIO Resources* (UG571)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/29/2018	1.0	Initial Xilinx release.
08/03/2018	1.1	Removed all references to *Native High-Speed I/O Interfaces* (XAPP1274). Added Asynchronous Data Capture. Updated description of phase alignment algorithm and first bullet of pin selection guidelines in Asynchronous Reference Design. Updated description of clocks before Figure 13. Updated Figure 4, Figure 7, Figure 8, Figure 10, Figure 12, and Figure 15.

Please Read: Important Legal Notices

AUTOMOTIVE APPLICATIONS DISCLAIMER