



# XPHY I/O Source Synchronous Interfaces

XAPP1350 (v1.0) February 4, 2021

## Summary

This application note and the associated reference designs describe how to construct source synchronous high-speed I/O interfaces using the Advanced I/O Wizard (AIOW) on Versal™ devices. The wizard instantiates and configures I/O and clocking logic, such as XPHY nibbles and XPLL blocks, that are included in the physical-side interface (PHY) architecture. The designs in this application note are not hardware tested, but are verified through behavioral simulation.

Download the [reference design files](#) from the Xilinx® website. For detailed information about the design files, see the single-bank source synchronous design [Reference Design](#) section and the multi-bank source synchronous design [Reference Design](#) section.

## Introduction

This application note covers two designs for a source synchronous application using the AIOW:

- Single-bank source synchronous design
- Multi-bank source synchronous design

The AIOW provides the option to choose the number of banks, but not to exceed three banks. The wizard creates one bank instance for each bank. Both designs use low-voltage differential signaling (LVDS) for data transmission speeds at 1800 Mb/s. See the *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics* ([DS957](#)) for the speeds supported that can transmit and receive the LVDS standard. The underlying I/O and XPLL clocking architecture for these designs can be found in the *Versal ACAP SelectIO Resources Architecture Manual* ([AM010](#)) and *Versal ACAP Clocking Resources Architecture Manual* ([AM003](#)), respectively.

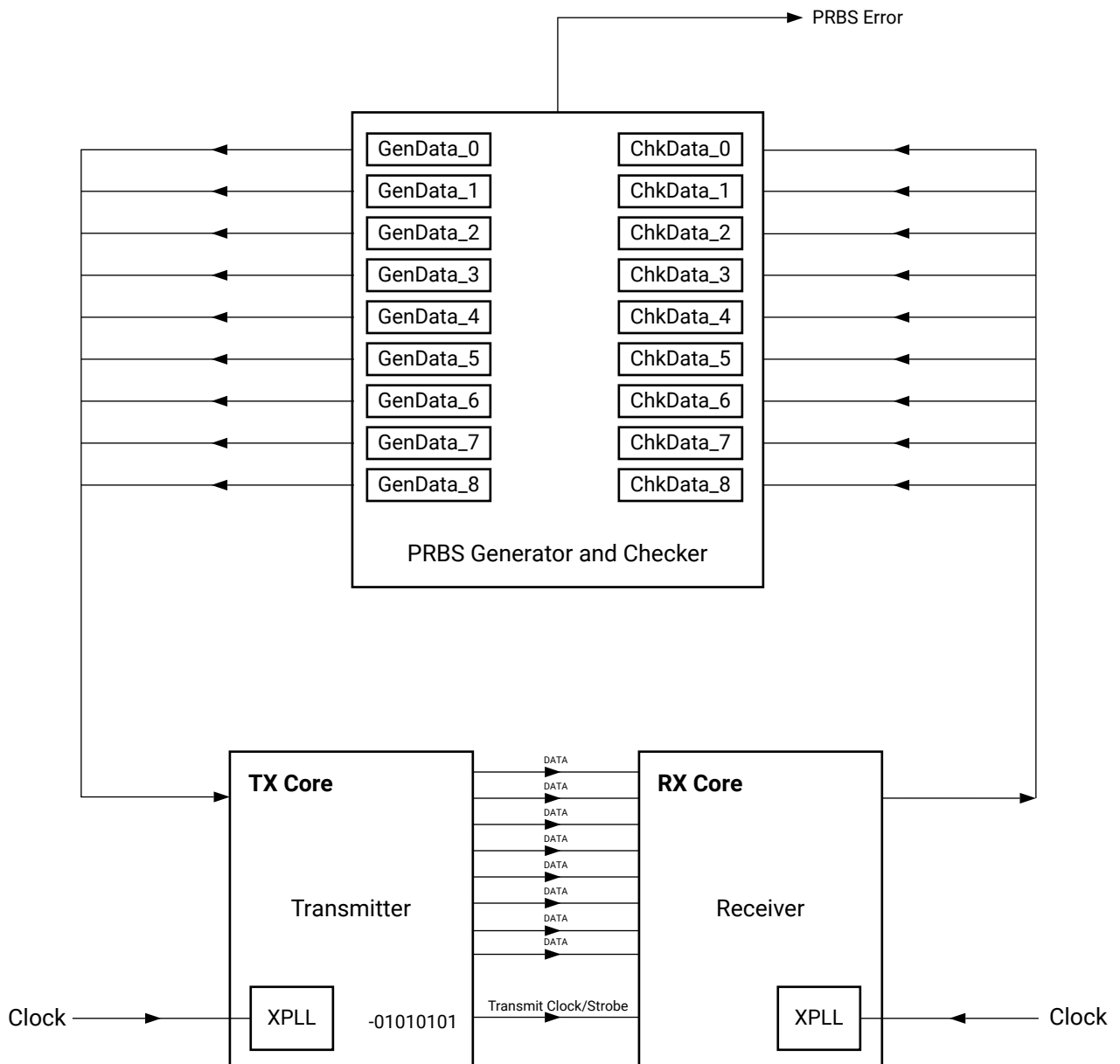
## Single-Bank Source Synchronous Design

In the single-bank source synchronous reference design, all nine XPHY nibbles of the XPIO bank are used. Each XPHY nibble contains six XPHY NIBBLESICES that can transmit and receive data from six individual I/O pins, for a total of 54 pins/bank. The transmit clock can be forwarded from the transmit core either by the clock forward pins or by the transmit data pins from the bank. In this design, the transmit clock is forwarded by the clock forward pins. Because the design uses the LVDS standard for the I/Os, data and clock are available in pairs of I/O pins.

The wizard configures clocking circuitry using an XPLL that is needed to support these configurations. In this design, the XPLL is instantiated in the core and the wizard is used to configure the XPLL clock frequency. The XPLL input clock is fed through a global clock (GC) input pin. The core uses one differential pair of I/O as the PLL input clock for RX/TX, one differential pair of I/O for the transmit/capture clock, and the rest for the data. Consequently, the Transmit and Receive core has 25 pairs for data transmission and reception. See the *Advanced I/O Wizard LogiCORE IP Product Guide* ([PG320](#)) to understand how to use the wizard beyond the scope of this application note.

The reference design uses the PRBS generator and checker to exercise the I/Os. The design files for the PRBS generator and checker are provided in the design suite. The generator and checker are instantiated in the top-level source files. The PRBS generator generates the data and feeds the TX core, which serializes it and transmits it to the RX core via an external loopback in the test bench. The RX core feeds the data to the PRBS checker after deserializing it. Because the design is a simulation-only design, the external loopback is achieved via wires in the test bench. The checker flags an error if it detects any mismatch. The block diagram of the reference design is shown in the following figure. The transmit clock is generated by feeding a 01010101 pattern to the corresponding clock forwarding pins.

Figure 1: Single-Bank Design



**Note:** The design has three PRBS generators/checkers per XPHY nibble, one for each pair of XPHY NIBBLESLICES transmitting and receiving data.

X24590-122320

## Pre-Core Generation Setup

Before following the procedure in this section, download the reference design files from the Xilinx website. For detailed information about the design files, see [Reference Design](#).

The following steps describe how to configure and set up the project before building the TX and RX cores using the AIOW.

1. Browse to the folder where the zipped file is downloaded.

2. Unzip the file and open the top-level `xapp1350` folder. Check the `Single_Bank_Ssync_Loopback_Design` folder, which has all the necessary design files under the `Sources`, `Constraints`, and `Testbench` folders.
3. Create a separate directory named `Versal_Ssync_RxTx_Intrfce_SB` to build the new project.
4. Launch the Vivado® tools 2020.1 or later from the newly created directory.
5. Under Quick Start, select **Create Project**.
6. Click **Next** for the prompt to **Create a New Vivado Project** and use `Versal_Ssync_RxTx_Intrfce_SB` for the name of the project. Deselect **Create a project subdirectory**.
7. Click **Next**. For Project Type, select **RTL project**. Deselect **Do not specify sources at this time**.
8. The next step is to add the sources. Add the source files from the `Sources` folder under the `Single_Bank_Ssync_Loopback_Design` folder.
9. Add the files `toplevel_sb.sv`, `Prbs_Any.vhd`, `Prbs_RxTx.vhd`, and `sync_cell.sv`. Make sure the Library is set to `xil_defaultlib` and the files are used for synthesis and simulation by setting it under HDL Source For.
10. Similarly, add the file `toplevel_testbench_sb.sv` from the `Testbench` folder under the `Single_Bank_Ssync_Loopback_Design` folder. Make sure the Library is set to `xil_defaultlib` and the file is used just for simulation by setting it under HDL Source For.
11. Select **Scan and add RTL include files into project** and **Copy sources into project**. Set the Target Language to **Verilog** and the Simulator Language to **Mixed**.
12. Click **Next** to proceed to adding the constraint files.
13. Add the file `toplevel_sb.xdc` from the `Constraints` folder under the `Single_Bank_Ssync_Loopback_Design` folder. Select **Copy constraints files into project**.
14. Click **Next** to select the part for the project. Select part `xcvc1902-vsva2197-2MP-e-S-eS1` for the reference designs and then click **Next**.
15. On the summary page for the project, make sure all the details match the settings and then click **Finish**.
16. The Vivado tools should create a project and display the hierarchy of the files under the `Sources` folder.

## Introduction to Deserialization and Data Reception

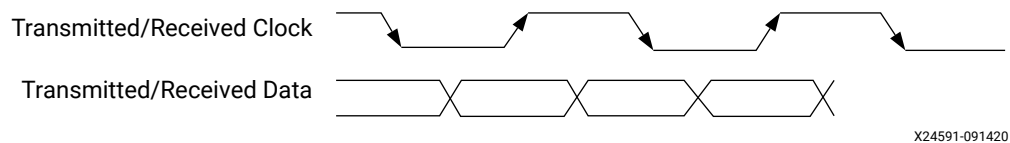
### Data Reception

In this source synchronous design, the capture clock is the same as the transmit clock, which is looped back from the TX to the RX core. The transmit clock is forwarded with the data by the TX core. The clock-to-data relation in this design is center alignment as shown in the following figure. The capture clock should be received on the package pin pair assigned in the constraints file. It should be received on `NIBBLESlice[0]` because it is the only `NIBBLESlice` that has the clock forwarding abilities.

The XPLL in the RX core needs a PLL input clock that is used by the XPHY for calibration (BISC). The PLL input clock (CLKIN port of the XPLL) should only be received on a GC pin. External global user clocks must be brought into the Versal device on (GC) inputs. There are four GC pin pairs in each bank. GC inputs provide dedicated, high-speed access to the internal global and regional clock resources. GC inputs use dedicated routing and must be used for clock inputs where the timing of various clocking features is imperative.

The data received at the RX core interface is forwarded through the bank instance to the programmable logic where it gets checked by the PRBS checker.

Figure 2: Center DDR



X24591-091420

## Configure and Generate a RX Advanced I/O Wizard Core

### Generating a RX Core

After following the previous steps to generate the top-level design, the next step is to generate a TX and RX core for operation. Follow these steps to generate the RX core using the AIOW. See the figures in this section for reference.

1. To start generating a core for RX, open the IP catalog and search for Advanced I/O Wizard. Double-click **Advanced I/O Wizard** from the catalog to open the Customize IP window.
2. For Component Name, enter `RX_1bank_Ssync_Intrfce`, which is used in the reference design.  
**Note:** The component name should match the module name used in the top-level design.
3. Under the Basic Tab, set Application to **SOURCE SYNCHRONOUS** from the drop-down list. Set Bus Direction to **RX ONLY**.
4. On the same tab, set the following:
  - a. Interface Speed: **1800 Mb/s**.
  - b. Clock Data Relation: **Center DDR**.
  - c. PLL Clock Source: **Clock Capable Pin**.
  - d. PLL Input Clock Frequency: **225**.
  - e. RX Serialization Factor: **8**.
  - f. The remaining options can be set to the default.

Figure 3: Single-Bank RX AIOW Basic Configuration

Component Name
RX\_1bank\_Sync\_Intfce

Basic
Advanced
Pin Configuration
IO Timing Estimation
Summary

General

Application
SOURCE SYNCHRONOUS
Bus Direction
RX ONLY

**RX**

The diagram illustrates the RX AIOW interface architecture. It shows a data bus (Input) entering the FPGA, passing through input delays and serializers into FIFOs. The data then moves through a Bitslip block and into the Fabric. A PLL block is shown, which is clocked by a Reference Clock and outputs a High Speed Clock. The PLL is also connected to the Fabric via a BUFG. The Fabric provides a CTRL\_CLK signal to the PLL. A note indicates that CTRL\_CLK can come from a local bank CC pin, fabric (BUFG), or PLL.

Clocking

Interface Speed(Mbps)
1800
[200 - 1800]
Clock Data Relation
Center DDR
PLL Clock Source
Clock Capable Pin
PLL Driven by Data Capture Clock
No
PLL Input Clock Frequency
225.000
Forwarded Clock Phase
0

Data and Control

Tx Serialization Factor
8
Rx Serialization Factor
8
3-State
Combinatorial
☐ Enable Bit-Slip
☐ Enable Data Bitslip
Bitslip Training Pattern
0x2C
☐ Enable RIU Interface

5. In the Advanced tab, set the following:
  - a. Select **REDUCE CONTROL SIGNALS**, **Enable BLI logic**, and **Enable DESKEW Logic**. When BLI logic is enabled, the BLI registers between fabric and XPHY can be used to help with timing closure. When deskew logic is enabled, the instantiated XPLLs are deskew enabled.
  - b. Differential I/O Std: **LVDS15**.
  - c. Number of Banks: **1** (because this is a single-bank design).

Figure 4: Single-Bank RX AIOW Advanced Configuration

Component Name: RX\_1bank\_Ssync\_Intfrc

**Basic** **Advanced** **Pin Configuration** **IO Timing Estimation** **Summary**

**Optional Ports**

☐ PLL CLKOUT1 200.000

☐ FIFO WRCLK OUT

☒ REDUCE CONTROL SIGNALS

☐ ENABLE DELAY CONTROL SIGNALS

☐ ENABLE CDR DEBUG SIGNALS

☒ Enable BLI logic

☒ Enable DESKEW Logic

**Advanced Clocking** **Debug** **IO Standard Selection** **Number of Banks**

☐ Generate CTRL CLK from PLL

Differential IO Std: LVDS15

Single IO Std: NONE

Number of Banks: 1

**Power Saving**

☐ IOB Power Saving

IOB Power Control: User Controlled

6. In the Pin Configuration tab, make two entries in the table. One entry is for the data with the capture clock and the other is for the PLL input clock. Twenty five pairs for data and 1 pair for the PLL input clock are configured under this tab. When selecting 25 pairs for data, the wizard adds 1 pair for the capture clock.
  - Data and the Strobe Setting
    - Pin Direction = RX
    - I/O Type = Differential
    - Signal Type = Data
    - Strobe I/O Type = Differential
    - Strobe Name = Strobe
    - Signal Name = Rx\_data\_pins
    - Number of Data Channels = 25
  - PLL Input Clock Setting
    - Pin Direction = RX

- I/O Type = Differential
- Signal Type = Input Clock
- Signal Name = clk
- Number of Data Channels = 1

Figure 5: Single-Bank RX AIOW Pin Configuration

Component Name: RX_1bank_Ssync_Interface									
Basic	Advanced	Pin Configuration	IO Timing Estimation	Summary					
Pin Direction	IO Type	Signal Type	Enable Strobe	Strobe IO Type	Strobe Name	Signal Name	Number of Data Channels		
Rx	Differential	Data	<input checked="" type="checkbox"/>	Differential	Strobe	Rx_data_pins	25		
Rx	Differential	Input Clock	<input type="checkbox"/>	Single-ended	Strobe_1	clk	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_2	Data_pins_2	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_3	Data_pins_3	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_4	Data_pins_4	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_5	Data_pins_5	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_6	Data_pins_6	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_7	Data_pins_7	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_8	Data_pins_8	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_9	Data_pins_9	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_10	Data_pins_10	1		
None	Configured	Data	<input type="checkbox"/>	Single-ended	Strobe_11	Data_pins_11	1		

7. Check the Summary tab. It should show 52 RX pins enabled (25 pairs of data + 1 pair of strobes). One pair is used by the PLL input clock that feeds the CLKIN port of the XPLL.

**Note:** In the following figure, the number of strobe pins enabled is incorrect. The correct value is 2.

Figure 6: Single-Bank RX AIOW Summary

Component Name				
RX_1bank_Ssync_Intfrce				
Basic	Advanced	Pin Configuration	IO Timing Estimation	Summary
Interface Speed : 1800 Mbps PLL Input CLock Freq : 225.000 MHz  Number of Tx Pins Enabled : 0 Number of Rx Pins Enabled : 52 Number of BiDir Pins Enabled : 0 Number of Strobe Pins Enabled : 3				
Default Nibble groups for the data pins in the design are as follows: ->RX_GROUP_0 Rx_data_pins_10 Rx_data_pins_11 Rx_data_pins_12 Rx_data_pins_13 Rx_data_pins_14 Rx_data_pins_15 ->RX_GROUP_1 Rx_data_pins_16 Rx_data_pins_17 Rx_data_pins_18 Rx_data_pins_19 Rx_data_pins_20 Rx_data_pins_21 ->RX_GROUP_2 strobe_0 strobe_1 Rx_data_pins_0 Rx_data_pins_1 Rx_data_pins_2 Rx_data_pins_3 ->RX_GROUP_3 Rx_data_pins_4 Rx_data_pins_5 Rx_data_pins_6 Rx_data_pins_7 Rx_data_pins_8 Rx_data_pins_9 ->RX_GROUP_4 Rx_data_pins_22 Rx_data_pins_23 Rx_data_pins_24 Rx_data_pins_25 Rx_data_pins_26 Rx_data_pins_27 ->RX_GROUP_5 Rx_data_pins_28 Rx_data_pins_29 Rx_data_pins_30 Rx_data_pins_31 Rx_data_pins_32 Rx_data_pins_33 ->RX_GROUP_6 Rx_data_pins_34 Rx_data_pins_35 Rx_data_pins_36 Rx_data_pins_37 Rx_data_pins_38 Rx_data_pins_39 ->RX_GROUP_7 Rx_data_pins_40 Rx_data_pins_41 Rx_data_pins_42 Rx_data_pins_43 Rx_data_pins_44 Rx_data_pins_45 ->RX_GROUP_8 clk_0 clk_1 Rx_data_pins_46 Rx_data_pins_47 Rx_data_pins_48 Rx_data_pins_49				
Next Steps: 1. Synthesize the generated design 2. IO planning of the design using IO planner or Nibble planner				

8. Click **OK** after reviewing the settings. The IP is now customized and the Generate the Output Products prompt appears. Set the Synthesis Option to **Out of context per IP** and then click **Generate** to launch the design run for the newly generated RX core. See the following table.

**Table 1: Receiver Requirements**

	Requirement
Component name	RX_1bank_Ssync_Intrfce
Bus direction	RX_ONLY
RX serialization factor	8
Interface speed (Mb/s)	1800 Mb/s
Clock data relation	Center DDR
PLL clock source	Clock capable pin
PLL input clock frequency	225 MHz
Include PLL in core	Yes
PLL CLKOUT1	No
FIFO WRCLK OUT	No
Reduce control signals	Yes
Enable delay control signals	No
Enable BLI logic	Yes
Enable deskew logic	Yes
Differential I/O Std	LVDS15
Number of banks	1
Pin configuration	The number of data channels is set to 25 pairs. One pair is added for the capture clock and one pair is assigned to the input clock for the XPLL.

## Receiver Design Considerations

This RX core is set up to work for a data rate of 1800 Mb/s. Also, the core is configured for LVDS15 in the reference design. This single-bank design uses a clock capable pin as a PLL clock source and is configured for a center aligned DDR system. An XPIO bank has 54 pins and the design uses all 54 pins in the form of 25 pairs of data pins, 1 pair of capture clock pins, and 1 pair of PLL input clock pins to the RX XPLL. The wizard provides the options to set the frequency for the CLKOUT1 port of the XPLL, which must be provided. The design constrains the ports for the receive interfaces and the wizard takes care of the placement.

# Introduction to Serialization and Data Transmission

## Data Transmission

In the reference design, the TX core sends out the transmit clock along with the data. The data in this design is generated using the PRBS generator and the transmit clock is generated by feeding the pattern 01010101 to the clock forwarding pins. The data generated by the PRBS generator is fed into the TX core from the programmable logic, which follows the TX datapath through a serializer and output delay within the PHY. The serializer supports 8:1, 4:1, and 2:1 serialization. In the design, an 8:1 serialization factor is used. The data is transmitted through the TX data pins of the core. To understand the data flow operation inside the TX core, see *Versal ACAP SelectIO Resources Architecture Manual* ([AM010](#)).

The PLL input clock to the TX core is fed to the XPLL through a clock capable GC/XCC pin within the same bank. In the design, the PLL input clock is received on the bank0\_pll\_clk\_in port of the TX core instantiated by the wizard. The transmit clock can be forwarded either through the data pins or the clock forwarding pins on the TX core. The design uses the clock forwarding pins to support the center DDR alignment. The wizard provides an option to set the phase on the forwarded clock to 90, which aligns the data-to-clock relation to support the center DDR on the RX core. The transmit clock is transmitted on NIBBLES\_LICE[0] and NIBBLES\_LICE[1] to the RX core.

## Configure and Generate a TX Advanced I/O Wizard Core

### Generating a TX Core

Follow these steps to generate the TX core using the AIOW. See the figures in this section for reference.

1. To start generating a core for TX, open the IP catalog and search for Advanced I/O Wizard. Double-click **Advanced I/O Wizard** to open the Customize IP window for the wizard.
2. For Component Name, enter `TX_1bank_Ssync_Interface`, which is used in the reference design.  
**Note:** The component name should match the module name used in the top-level design.
3. Under the Basic Tab, set Application to **SOURCE SYNCHRONOUS** from the drop-down list, and set Bus Direction to **TX ONLY**.
4. On the same tab, set the following:
  - a. Interface Speed: **1800 Mb/s**.
  - b. Clock Data Relation: **Edge DDR**.
  - c. PLL Clock Source: **Fabric (Driven by BUFG)**.
  - d. PLL Input Clock Frequency: **225**.
  - e. Forwarded Clock Phase to **90**.
  - f. TX Serialization Factor: **8**

- g. Select **Include PLL in Core**.
- h. The remaining options can be set to the default.

Figure 7: Single-Bank TX AIOW Basic Configuration

Component Name

TX\_1bank\_Ssync\_Intrfce

Basic

Advanced

Pin Configuration

IO Timing Estimation

Summary

General

Application

SOURCE SYNCHRONOUS

Bus Direction

TX ONLY

TX

TX Clock Source

FPGA

Fabric

CTRL\_CLK can come from local bank, CC pin, fabric (BUFG), or PLL

BUFG

PLL

High Speed Clock Generated from Intern Source clock can come from BUFG or CC

CTRL\_CLK

CLKOUT\_PHY

Nibble

FIFO

Serializer

Output Delay

Data Bus (Output)

Clocking

Interface Speed(Mbps)

1800

[200 - 1800]

Clock Data Relation

Edge DDR

PLL Clock Source

Fabric(Driven by BUFG)

PLL Driven by Data Capture Clock

No

PLL Input Clock Frequency

225.000

Forwarded Clock Phase

90

Data and Control

Tx Serialization Factor

8

Rx Serialization Factor

8

3-State

Combinatorial

Enable Bit-Slip

Enable Data Bit-slip

Bit-slip Training Pattern

0x2C

Enable RIU Interface

Include PLL in Core

5. In the Advanced tab, set the following:
  - a. Select **REDUCE CONTROL SIGNALS**, **Enable BLI logic**, and **Enable DESKEW Logic**. When BLI logic is enabled, the BLI registers between fabric and XPHY can be used to help with timing closure. When deskew logic is enabled, the instantiated XPLLs are deskew enabled.
  - b. Differential I/O Std: **LVDS15**
  - c. Number of Banks: **1** (because this is a single-bank design).

**Figure 8: Single-Bank TX AIOW Advanced Configuration**

Component Name: TX\_1bank\_Ssync\_Intrfce

**Basic** **Advanced** Pin Configuration IO Timing Estimation Summary

**Optional Ports**

☐ PLL CLKOUT1 200.000

☐ FIFO WRCLK OUT

☒ REDUCE CONTROL SIGNALS

☐ ENABLE DELAY CONTROL SIGNALS

☐ ENABLE CDR DEBUG SIGNALS

☒ Enable BLI logic

☒ Enable DESKEW Logic

**Advanced Clocking** **Debug** **IO Standard Selection** **Number of Banks**

☐ Generate CTRL CLK from PLL

Differential IO Std: LVDS15

Single IO Std: NONE

Number of Banks: 1

**Power Saving**

☐ IOB Power Saving

IOB Power Control: User Controlled

6. In the Pin Configuration tab, add two entries in the table. One entry is for the data with I/O type as differential, signal type as data, and signal name as Tx\_data\_pins. The other entry is for the clock forwarding pins with I/O type as differential, signal type as Clk Fwd, and signal name as Clk\_fwd.

- Data Setting
  - Pin Direction = TX
  - I/O Type = Differential
  - Signal Type = Data
  - Signal Name = Tx\_data
  - Number of Data Channels = 25
- Data Setting
  - Pin Direction = TX
  - I/O Type = Differential
  - Signal Type = Clk Fwd

- Signal Name = Clk\_fwd
- Number of Data Channels = 1

Figure 9: Single-Bank TX AIOW Pin Configuration

Component Name: TX_1bank_Ssync_Intfrce							
Basic	Advanced	Pin Configuration	IO Timing Estimation	Summary			
Pin Direction	IO Type	Signal Type	Enable Strobe	Strobe IO Type	Strobe Name	Signal Name	Number of Data Channels
TX	Differential	Data	<input checked="" type="checkbox"/>	Single-ended	Strobe_0	Tx_data	25
TX	Differential	Clk Fwd	<input checked="" type="checkbox"/>	Single-ended	Strobe_1	Clk_fwd	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_2	Data_pins_2	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_3	Data_pins_3	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_4	Data_pins_4	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_5	Data_pins_5	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_6	Data_pins_6	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_7	Data_pins_7	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_8	Data_pins_8	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_9	Data_pins_9	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_10	Data_pins_10	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_11	Data_pins_11	1
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_12	Data_pins_12	1

- Check the Summary tab. It should show 52 TX pins enabled (25 pairs of data + 1 pair of forwarded clock). The PLL input clock for the TX is driven from the programmable logic as selected under the Basic tab while configuring the TX core.

Figure 10: Single-Bank TX AIOW Summary

Component Name
TX\_1bank\_Ssync\_Intfrce

Basic
Advanced
Pin Configuration
IO Timing Estimation
Summary

Interface Speed : 1800 Mbps  
PLL Input Clock Freq : 225.000 MHz  
  
Number of Tx Pins Enabled : 52  
Number of Rx Pins Enabled : 0  
Number of BiDir Pins Enabled : 0  
Number of Strobe Pins Enabled : 0

**Default Nibble groups for the data pins in the design are as follows:**

->TX\_GROUP\_0 Tx\_data\_0 Tx\_data\_1 Tx\_data\_2 Tx\_data\_3 Tx\_data\_4 Tx\_data\_5

->TX\_GROUP\_1 Tx\_data\_6 Tx\_data\_7 Tx\_data\_8 Tx\_data\_9 Tx\_data\_10 Tx\_data\_11

->TX\_GROUP\_2 Tx\_data\_12 Tx\_data\_13 Tx\_data\_14 Tx\_data\_15 Tx\_data\_16 Tx\_data\_17

->TX\_GROUP\_3 Tx\_data\_18 Tx\_data\_19 Tx\_data\_20 Tx\_data\_21 Tx\_data\_22 Tx\_data\_23

->TX\_GROUP\_4 Tx\_data\_24 Tx\_data\_25 Tx\_data\_26 Tx\_data\_27 Tx\_data\_28 Tx\_data\_29

->TX\_GROUP\_5 Tx\_data\_30 Tx\_data\_31 Tx\_data\_32 Tx\_data\_33 Tx\_data\_34 Tx\_data\_35

->TX\_GROUP\_6 Tx\_data\_36 Tx\_data\_37 Tx\_data\_38 Tx\_data\_39 Tx\_data\_40 Tx\_data\_41

->TX\_GROUP\_7 Tx\_data\_42 Tx\_data\_43 Tx\_data\_44 Tx\_data\_45 Tx\_data\_46 Tx\_data\_47

->TX\_GROUP\_8 Tx\_data\_48 Tx\_data\_49 Clk\_fwd\_0 Clk\_fwd\_1

**Next Steps:**

- Synthesize the generated design
- IO planning of the design using IO planner or Nibble planner

- Click **OK** after reviewing the settings. The IP is now customized and the Generate the Output Products prompt appears. Set the Synthesis Option to **Out of context per IP** and then click **Generate** to launch the design run for the newly generated TX core. See the following table.

**Table 2: Transmitter Requirements**

	Requirement
Component name	TX_1bank_Ssync_Intrfce
Bus direction	TX_ONLY
TX serialization factor	8
Interface speed (Mb/s)	1800 Mb/s
Clock data relation	Edge DDR
PLL clock source	Fabric (driven by BUFG)
Forwarded clock phase	90
PLL input clock frequency	225 MHz
Include PLL in core	Yes
PLL CLKOUT1	No
FIFO WRCLK OUT	No
Reduce control signals	Yes
Enable delay control signals	No
Enable BLI logic	Yes
Enable deskew logic	Yes
Differential I/O Std	LVDS15
Number of banks	1
Pin configuration	The number of data channels is set to 25 pairs. One pair is reserved for the transmit clock via the clock forwarding pins.

## Transmitter Design Considerations

This TX core is set up for a data rate of 1800 Mb/s. Also, the core is configured and tested for LVDS15 in the reference design. This single bank design uses the same clock to feed the XPLL of both the RX and TX cores. The PLL input clock that feeds the TX core is driven through the IBUFDS and then passed into the core. An XPIO bank has 54 pins and the design uses 52 pins in the form of 25 pairs of data pins and 1 pair of transmit clock/strobe via the clock forwarding pins. One pair is reserved for the PLL input clock to the RX core, and consequently, is not used for TX core. The design constrains the ports for the transmit interfaces and the placement is taken care of by the Vivado tools.

## Top-Level using TX and RX Advanced I/O Wizard Cores

The top-level design file (`toplevel_sb sv`) includes the `toplevel_sb` module. This module helps connect the interfaces such as clocks, debug ports, I/O ports, etc., with the appropriate sources. The top-level design houses the instantiation of both the RX and the TX cores. To test the design, the PRBS patterns from the custom PRBS generator provided in the design suite can be used to generate and check the received data.

The PRBS generator generates 8-bit data for each pair of pins and feeds it to the TX core, which in turn transmits it through the TX pins. The PRBS generator also houses an error injecting mechanism. The data is received on the I/O ports of the RX core through external loopback. The RX core forwards the deserialized data through the PHY to the programmable logic. This deserialized data is then fed into the PRBS checker to check for any failures.

The top-level design file includes a constraints file (`toplevel_sb.xdc`). This file is used to create clocks for the design, assign locations or pins to all the I/O ports in the design, and set attributes if needed. The reference design constrains the design to optimally support high data rates. The user must constrain the TX and RX ports. The Vivado tools can assign the XPHY nibbles to the XPHY sites. The Advanced I/O Planner can be used to help with pin assignments (see *Advanced I/O Wizard LogiCORE IP Product Guide* ([PG320](#))) when using the AIOW.

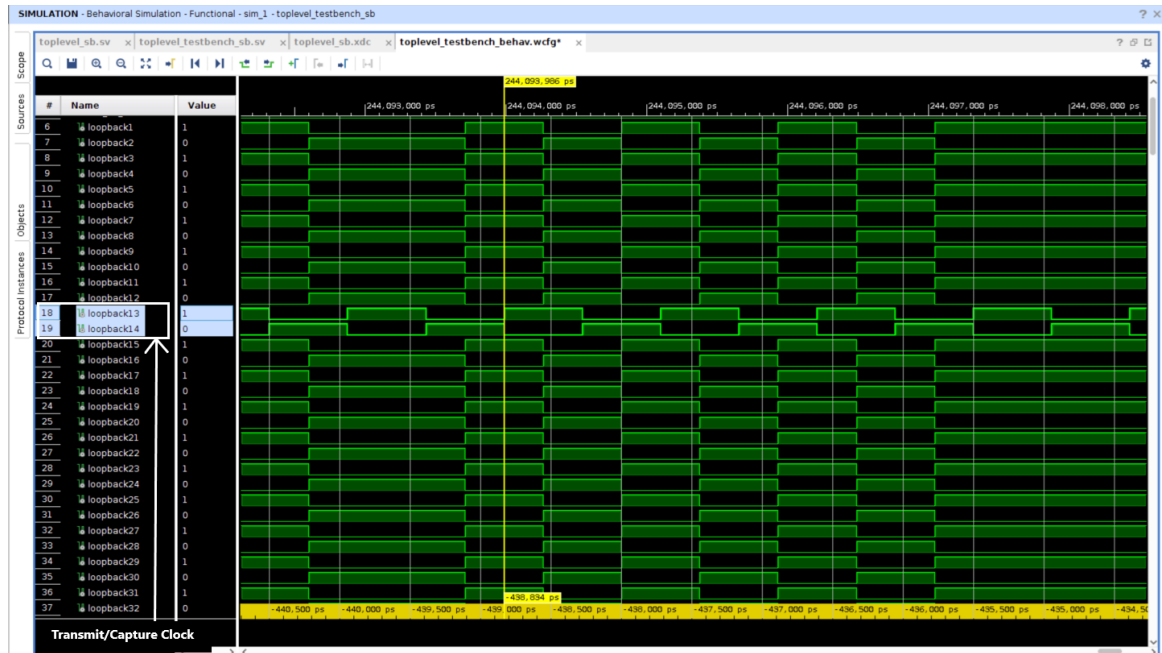
In the reference design, bank 707 for part `xvc1902-vsva2197-2MP-e-S-eS1` is used for the TX core and bank 704 is used for the RX core. The RX core has one pair reserved for the PLL input clock. The design assigns `IO_L9P_GC_XCC_N3P0_M1P72_704_BE31` and `IO_L9N_GC_XCC_N3P1_M1P73_704_BD32` in the constraints file. Similarly, `IO_L6P_GC_XCC_N2P0_M1P66_704_BC31` and `IO_L6N_GC_XCC_N2P1_M1P67_704_BC30` are assigned to the strobe. The corresponding pins in bank 707 are assigned to the PLL input clock and forwarded clock for the TX core. Check the board schematics and the `toplevel_sb.xdc` constraints file to understand the pin assignments.

At this point, all the files are added to the project and both the RX and TX cores are generated. The design is ready to be synthesized and implemented.

## Simulation

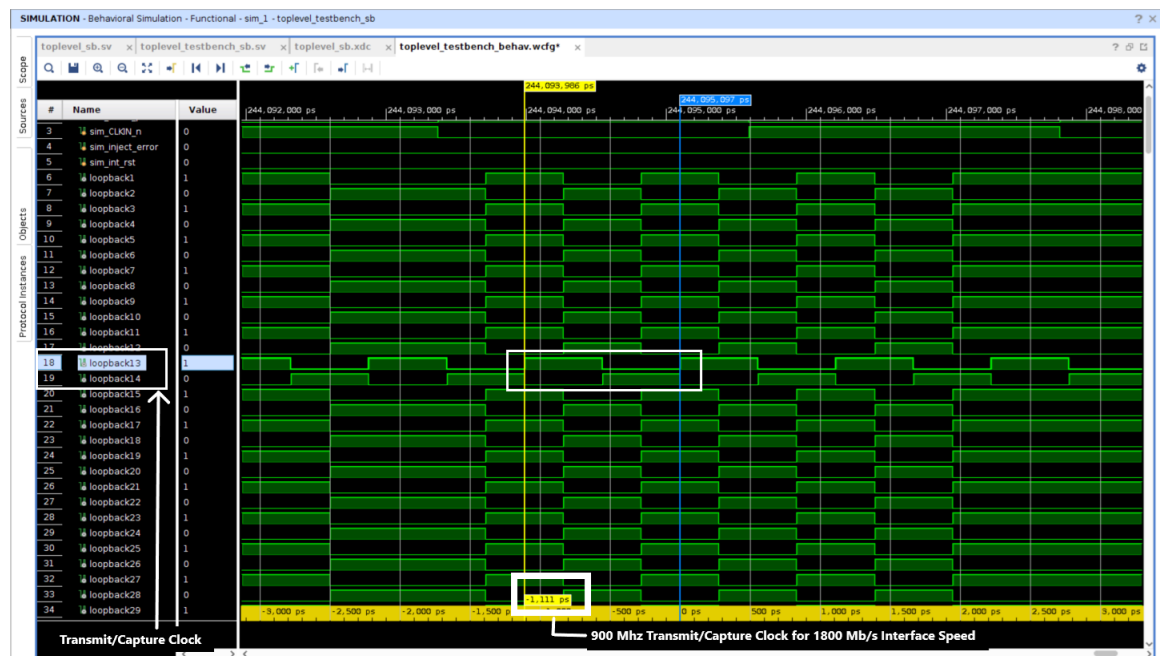
The reference design uses the `toplevel_testbench_sb.sv` file to create a simple test bench. This test bench connects the TX core to the RX core via loopback<num> connections (wires). The transmit clock is transmitted on loopback13 and loopback14 loopback wires. All the other loopback connections are used to transmit and receive the data. The clock-to-data relationship is center aligned as shown in the following figure captured from the simulation.

Figure 11: Clock-to-Data Relation - Center Alignment



The test bench provides the necessary clock and resets to the design and triggers its operation. The PLL input clock is provided at 4.444 ns (225 MHz) to the RX and TX cores. The transmit/capture clock toggles at 900 MHz (1.111 ns period) as shown in the following figure. Because the system is double data rate, the interface operates at 1800 Mb/s.

Figure 12: Single Bank Design - Interface Speed 1800 Mb/s



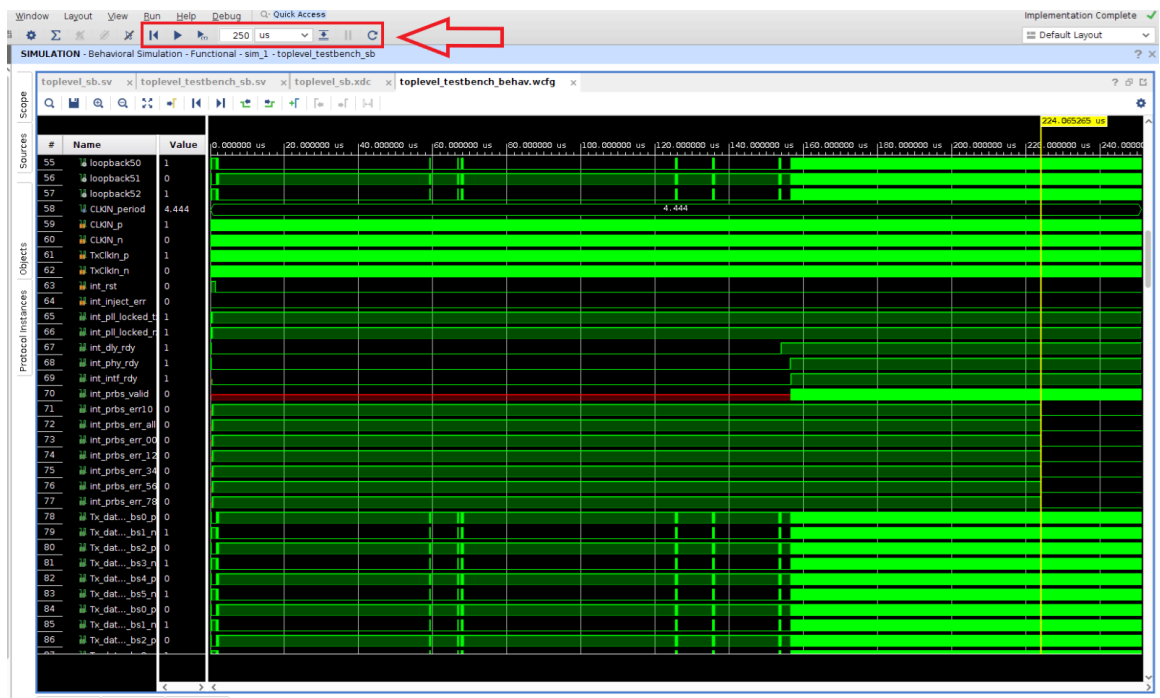
**Note:** The Vivado 2020.1 tools allow only behavioral simulation when using the AIOW.

## Simulating the Design

The design is tested with the Vivado Simulator 2020.1. This section describes how to launch the simulation. Assuming the Vivado project is already created for the design, follow these steps to simulate the design.

1. Click **Run Simulation** under Simulation from the Flow Navigator.
2. From the listed options, select **Run Behavioral Simulation**, which elaborates the design and launches the simulation.
3. The run time by default is 1 ns for the simulation. It takes about 230  $\mu$ s for the `intf_rdy` to be asserted for the RX core. Only after its assertion and the next `Prbs_Valid`, should the data be compared. Consequently, launch the simulation for a duration of more than 230  $\mu$ s using the box highlighted in the following figure. A waveform config file `toplevel_testbench_behav.wcfg` is provided under the `Testbench` directory to add a list of signals for the simulation.

Figure 13: Single-Bank Design - Changing the Run Time for Simulation



4. Once the behavioral simulation is finished, check for any errors by observing error flags for the PRBS generator and checker. For example, `int_prbs_err<num>` reports the errors for each instantiation of the PRBS generator and checker. `int_prbs_err24` denotes the error flag for the PRBS module instantiated for NIBBLE[2], NIBBLESlice[4]. `int_prbs_err_00` denotes the error for any NIBBLESlices on NIBBLE[2]. `int_prbs_err_12` denotes the error for any NIBBLESlices on NIBBLE[1] and NIBBLE[2]. `int_prbs_err_all` denotes an error on any NIBBLESlices across all the nibbles.

## Reference Design

Download the [reference design files](#) from the Xilinx website. Unzip this file and browse into the top-level folder `xapp1350`. Check the `Single_Bank_Ssync_Loopback_Design` folder, which has all the source files under `Sources`, `Constraints`, and `Testbench`.

**Table 3: Source Files for Single-Bank Design**

Folder	File Name	Description
Sources	<code>toplevel_sb.sv</code>	The top-level design file that instantiates the TX and RX cores and connects the design
	<code>Prbs_RxTx.vhd</code>	Outer wrapper for the custom PRBS generator and checker
	<code>Prbs_Any.vhd</code>	Contains the modules for the PRBS generator and checker
Constraints	<code>toplevel_sb.xdc</code>	The constraints file to assign pin locations to ports of the design and set the necessary attributes
Test Bench	<code>toplevel_testbench_sb.sv</code>	The top-level test bench file to test the design
	<code>toplevel_testbench_behav.wcfg</code>	Waveform configuration file

## Reference Design Matrix

The following checklist indicates the procedures used for the provided reference design.

**Table 4: Reference Design Matrix**

Parameter	Description
General	
Developer name	Xilinx
Target devices	Versal ACAP
Source code provided?	Yes
Source code format (if provided)	Verilog and VHDL
Design uses code or IP from existing reference design, application note, third party or Vivado software? If yes, list.	Yes. Uses the Advanced I/O Wizard from the IP catalog.
Simulation	
Functional simulation performed	Yes
Timing simulation performed?	No
Test bench provided for functional and timing simulation?	Yes
Test bench format	Verilog
Simulator software and version	Vivado simulator 2020.1
SPICE/IBIS simulations	No
Implementation	
Synthesis software tools/versions used	Vivado synthesis 2020.1
Implementation software tool(s) and version	Vivado 2020.1 implementation
Static timing analysis performed?	Yes
Hardware Verification	
Hardware verified?	No
Platform used for verification	N/A

## Multi-Bank Source Synchronous Design

In the multi-bank source synchronous reference design, all three available banks are used, which is the maximum width of a single I/O interface supported by the AIOW. This design uses all nine XPHY nibbles from one XPIO bank, with each XPHY nibble containing six XPHY NIBBLESLICEs that transmit and receive data from six individual I/O pins, for a total of 54 pins/bank. This design has three banks, and consequently, uses a total of 27 XPHY nibbles. The wizard requires the banks to be adjacent to each other. The transmit clock can be forwarded from the transmit core either via the clock forward pins or the transmit data pins from the bank. In the reference design, the transmit clock is forwarded via the transmit data pins. The design uses the LVDS standard for the I/Os, and consequently, the data, transmit/capture clock, and PLL input clock use differential I/O pin pairs.

The wizard configures clocking circuitry using the XPLL that is needed to support these configurations. In this design, the XPLL is instantiated in the core and the wizard configures the XPLL clock frequencies. The XPLL input clock is fed through a global clock (GC) input pin. As a result, the core reserves a pair of I/Os for the PLL input clock for RX, one pair of I/Os for the capture clock, and the remainder for the data in each bank. Consequently, the transmit and receive cores have 25 pairs for data transmission and reception per bank. See *Advanced I/O Wizard LogiCORE IP Product Guide* ([PG320](#)) for more information on the AIOW. Each bank has a transmit clock of its own that the transmitter transmits. Similarly, the receiver receives a capture clock on each bank.

Clocking in the multi-bank source synchronous reference design is similar to the single-bank design. Instead of one instantiation of the XPLL as in the single-bank design, this design has three instantiations of the XPLLs, one for each bank. A common PLL input clock for all the XPLLs is fed into the RX core via the clock capable pins. This clock drives the CLKIN of the XPLLs for each bank. The TX core has three separate inputs for the PLL input clock, which is fed to the bank<0/1/2>\_pll\_clkln ports of the TX core. It uses the same clock as the RX core except that it passes through an IBUFDS to create a single-ended version that is passed to the respective CLKIN ports of each bank on the TX core.

In this design, the TX core uses the "Fabric (driven by BUFG)" option as the PLL clock source in the AIOW. These clocks are received on the GC pins and, consequently, do not need any BUFGs added with the IBUFDS. The clocks need BUFGs added with the IBUFDS if they are not received on the GC pins. The following figures illustrate how the clocks are connected to the TX and RX cores, respectively.

Figure 14: Clock Connection for TX Core

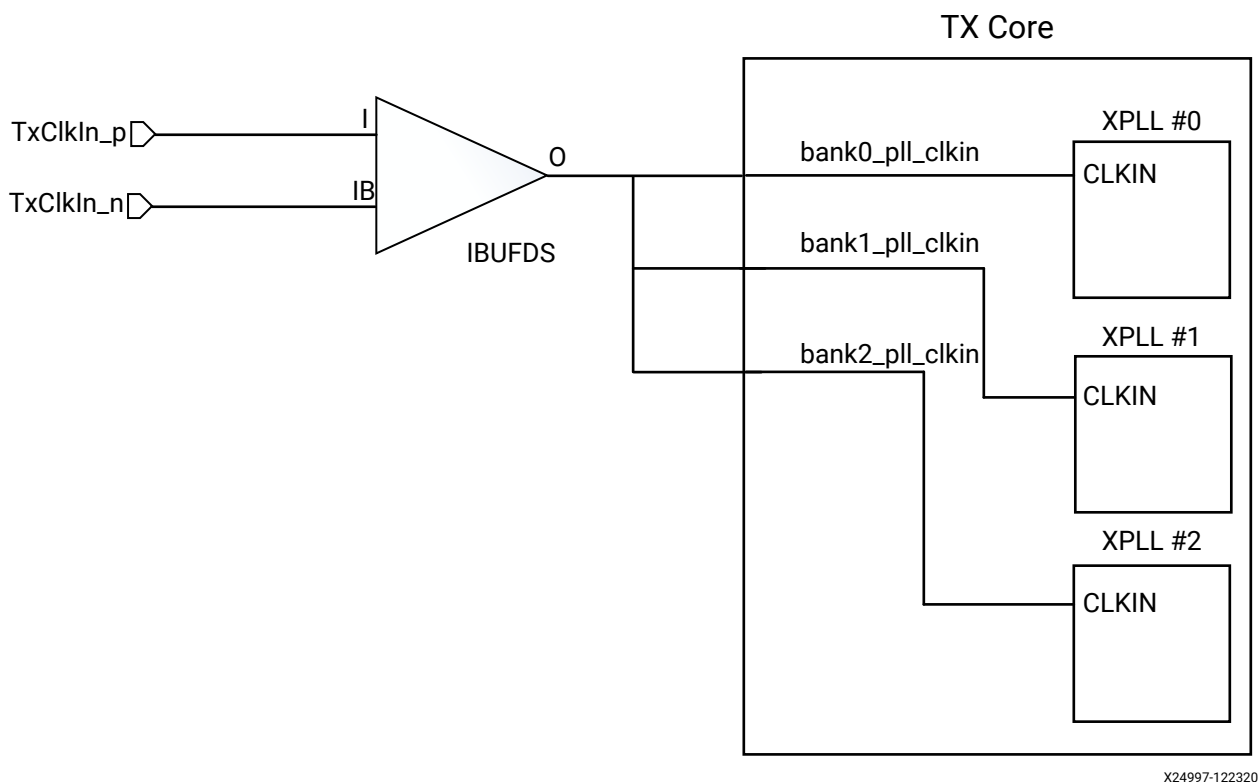
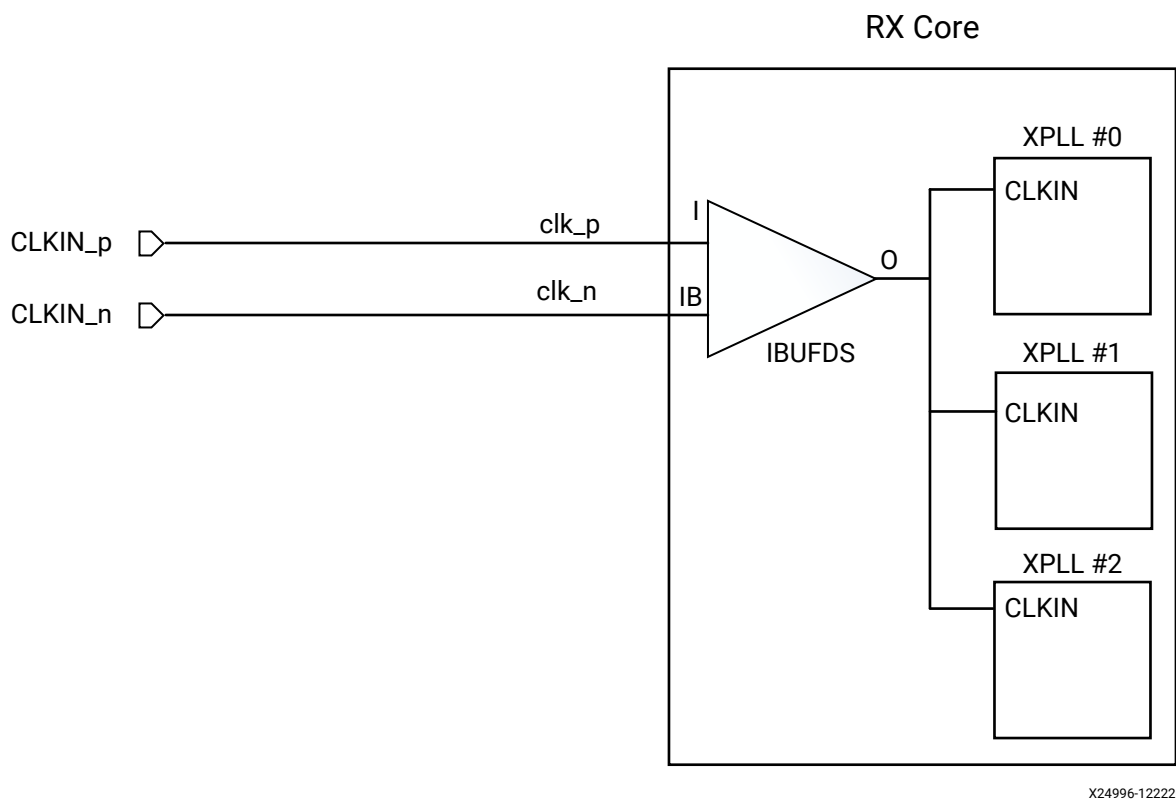


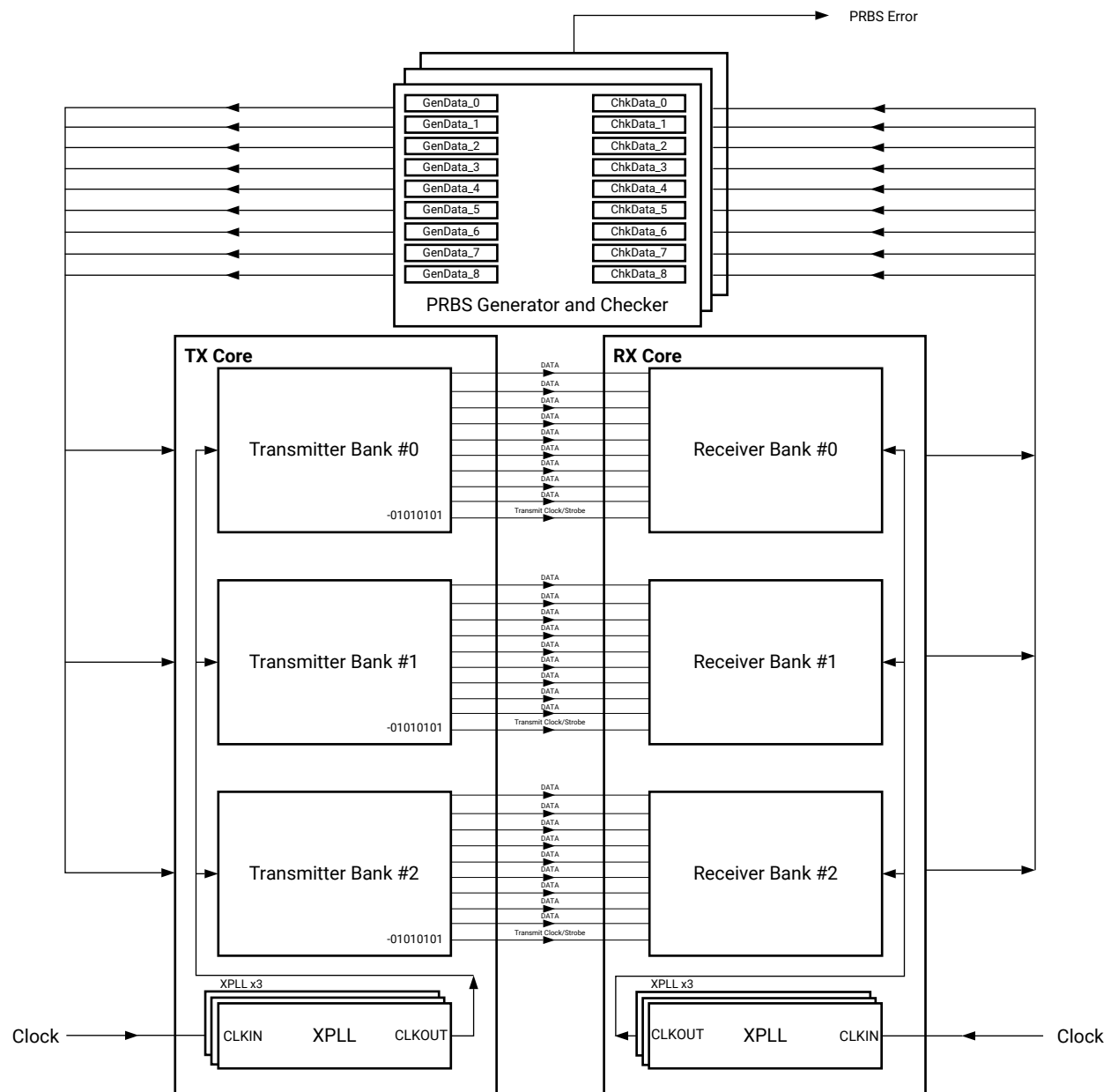
Figure 15: Clock Connection for RX Core



The reference design uses the PRBS generator and checker to exercise the I/Os. The design files for the PRBS generator and checker are provided in the design suite. It is instantiated in the top-level source file. The PRBS generator generates the data and feeds the TX core, which serializes it and transmits to the RX core via external loopback. The RX core feeds the data to the PRBS checker after deserializing it. It flags if it detects any mismatch. The block diagram of the reference design is shown in the following figure. The transmit clock or strobe is generated by feeding the pattern 01010101 to the corresponding TX NIBBLESlices for each bank.

**Note:** The design has three PRBS generators/checkers per XPHY nibble, one for each pair of XPHY NIBBLESlices transmitting and receiving data.

Figure 16: Multi-Bank Design



X24592-020221

## Pre-Core Generation Setup

Before following the procedure in this section, download the reference design files from the Xilinx website. For detailed information about the design files, see [Reference Design](#).

The following steps describe how to configure and set up the project before building the TX and RX cores using the AIOW.

1. Browse to the folder where the zipped file is downloaded.
2. Unzip the file and open the top-level `xapp1350` folder. Check the `Multi_Bank_Ssync_Loopback_Design` folder, which has all the necessary design files under the `Sources`, `Constraints`, and `Testbench` folders.
3. Create a separate directory named `Versal_Ssync_RxTx_Intrfce_MB` to build the new project.
4. Launch the Vivado tools 2020.1 or later from the newly created directory.
5. Under Quick Start, select **Create Project**.
6. Click **Next** for the prompt to **Create a New Vivado Project** and use `Versal_Ssync_RxTx_Intrfce_MB` for the name of the project. Deselect **Create a project subdirectory**.
7. Click **Next**. For Project Type, select **RTL project**. Deselect **Do not specify sources at this time**.
8. The next step is to add the sources. Add the source files from the `Sources` folder under the `Multi_Bank_Ssync_Loopback_Design` folder.
9. Add the files `toplevel_mb.sv`, `Prbs_Any.vhd`, `Prbs_RxTx.vhd`, and `sync_cell.sv`. Make sure the Library is set to `xil_defaultlib` and the files are used for synthesis and simulation by setting it under HDL Source For.
10. Similarly, add the file `toplevel_testbench_mb.sv` from the `Testbench` folder under the `Multi_Bank_Ssync_Loopback_Design` folder. Make sure the Library is set to `xil_defaultlib` and the file is used just for simulation by setting it under HDL Source For.
11. Select **Scan and add RTL include files into project** and **Copy sources into project**. Set the Target Language to **Verilog** and the Simulator Language to **Mixed**.
12. Click **Next** to proceed to adding the constraint files.
13. Add the files `toplevel_mb.xdc` and `attributes_mb.xdc` from the `Constraints` folder under the `Multi_Bank_Ssync_Loopback_Design` folder. Select **Copy constraints files into project**.
14. Click **Next** to proceed to select the part for the project. Select part **xcvc1902-vsua2197-2MP-e-S-eS1** for the reference designs and then click **Next**.
15. On the summary page for the project, make sure all the details match the settings and then click **Finish**.
16. The Vivado tools should create a project and display the hierarchy of the files under the `Sources` folder.

17. Under the `Sources` folder, right-click on `attributes_mb.xdc` and click **Source File Properties**. Under Source File Properties, open the tab for Properties and under `USED_IN`, select `opt_design_post`.

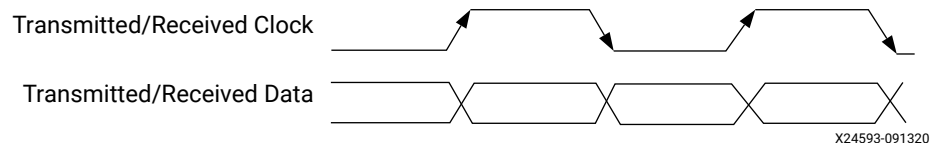
## Introduction to Deserialization and Data Reception

### Data Reception

In this source synchronous design, the capture clock is the same as the transmit clock, which is looped back from the TX to the RX core. The transmit clock is forwarded with the data by the TX core. The clock-to-data relation in this design is edge aligned and is shown in the following figure. The XPLL in the RX core needs a PLL input clock apart from the capture clock received with the data. This PLL input clock acts as the input to the XPLL (CLKIN port of the XPLL) and should only be received on a GC/XCC pin. Because this is a multi-bank design (three banks), the wizard instantiates three instances of the XPLLs for each bank. Consequently, the PLL input clock received on the GC/XCC pin is fed to the CLKIN ports of all the XPLL instantiations in the design. The capture clock is received on `NIBBLESlice[0]` and `NIBBLESlice[1]` of `NIBBLE[2]`, which forwards it to other nibbles through inter-byte and inter-nibble clocking. `NIBBLESlice[0]` and `NIBBLESlice[1]` have clock forwarding abilities. See the "Inter-nibble and Inter-byte Clocking Within an XPIO Bank" figure in the *Versal ACAP SelectIO Resources Architecture Manual* (AM010).

The data received at the RX core interface is transmitted to the programmable logic via the PHY and the bank instances, where it gets checked by the PRBS checker.

Figure 17: Edge DDR



## Configure and Generate a RX Advanced I/O Wizard Core

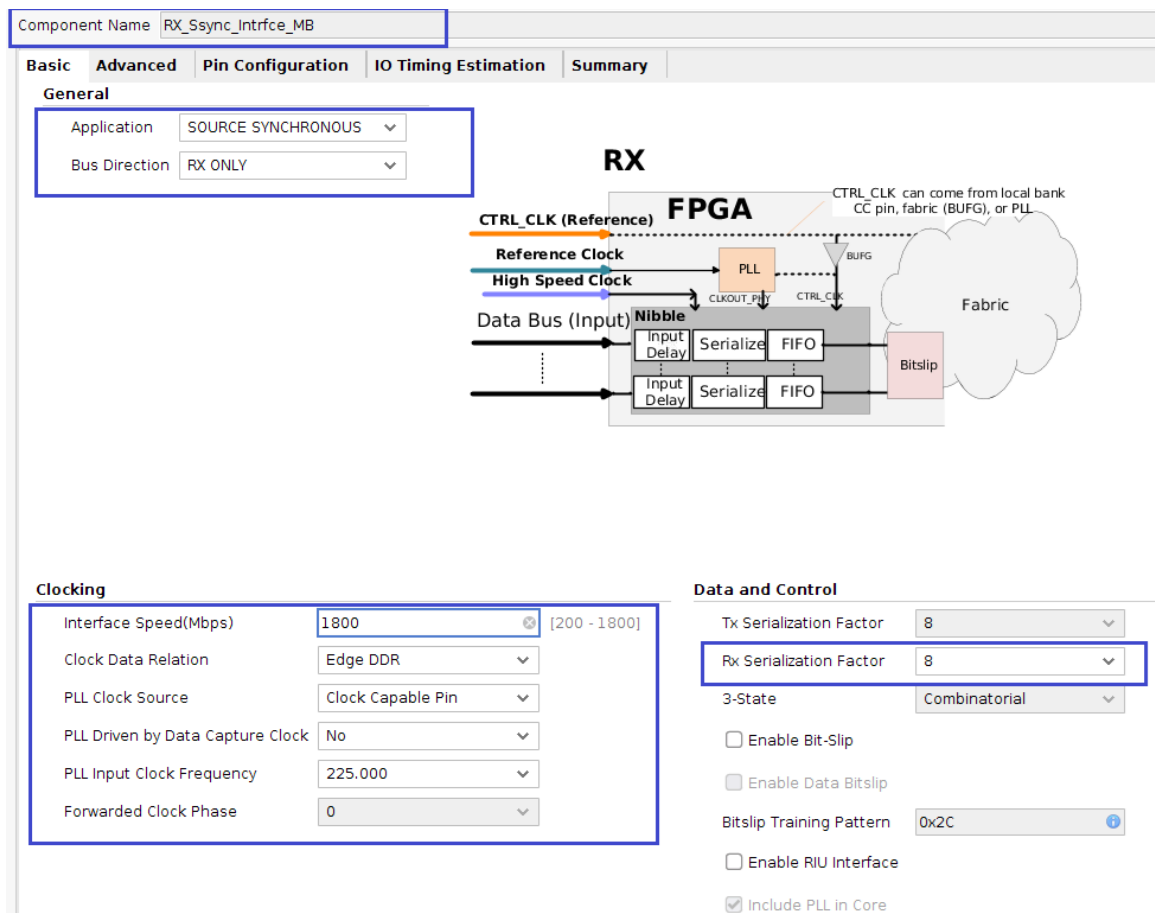
### Generating a RX Core

After following the previous steps to generate the top-level design, the next step is to generate a TX and RX core for operation. Follow these steps to generate the RX core using the AIOW. See the figures in this section for reference.

1. To start generating a core for RX, open the IP catalog and search for Advanced I/O Wizard. Double-click **Advanced I/O Wizard** from the catalog to open the Customize IP window.
2. For Component Name, enter `RX_Ssync_Intrfce_MB`, which is used in the reference design.  
**Note:** The component name should match the module name used in the top-level design.
3. Under the Basic Tab, set Application to **SOURCE SYNCHRONOUS** from the drop-down list. Set Bus Direction to **RX ONLY**.

4. On the same tab, set the following:
  - a. Interface Speed: **1800 Mb/s**.
  - b. Clock Data Relation: **Edge DDR**.
  - c. PLL Clock Source: **Clock Capable Pin**.
  - d. PLL Input Clock Frequency: **225**.
  - e. RX Serialization Factor: **8**.
  - f. The remaining options can be set to the default.

Figure 18: Multi-Bank RX AIOW Basic Configuration



5. In the Advanced tab, set the following:
  - a. Select **REDUCE CONTROL SIGNALS**, **Enable BLI logic**, and **Enable DESKEW Logic**.  
When the BLI logic is enabled, the BLI registers between the fabric and the XPHY can be used to help with timing closure. When deskew logic is enabled, the instantiated XPLLs are deskew enabled.  
  
When Deskew Logic is enabled, XPLLs instantiated will be deskew enabled.
  - b. Differential I/O Std: **LVDS15**.
  - c. Number of Banks: **3** (because this is a multi-bank design).

Figure 19: Multi-Bank RX AIOW Advanced Configuration

Component Name: RX\_Ssync\_Intfrc\_MB

Basic Advanced Pin Configuration IO Timing Estimation Summary

**Optional Ports**

☐ PLL CLKOUT1 200.000

☐ FIFO WRCLK OUT

☒ REDUCE CONTROL SIGNALS

☐ ENABLE DELAY CONTROL SIGNALS

☐ ENABLE CDR DEBUG SIGNALS

☒ Enable BLI logic

☒ Enable DESKEW Logic

**Advanced Clocking**

☐ Generate CTRL CLK from PLL

**Debug IO Standard Selection**

Differential IO Std: LVDS15

Single IO Std: NONE

**Number of Banks**

Number of Banks: 3

**Power Saving**

☐ IOB Power Saving

IOB Power Control: User Controlled

6. In the Pin Configuration tab, make four entries in the table. One entry is for the data and the strobe per bank and the other is for the common PLL input clock for all the banks.
  - Data and the Strobe Setting #1
    - Pin Direction = RX
    - I/O Type = Differential
    - Signal Type = Data
    - Strobe I/O Type = Differential
    - Strobe Name = strobe\_b0
    - Signal Name = Rx\_data\_b0
    - Number of Data Channels = 25
  - Data and the Strobe Setting #2
    - Pin Direction = RX
    - I/O Type = Differential
    - Signal Type = Data
    - Strobe I/O Type = Differential
    - Strobe Name = strobe\_b1
    - Signal Name = Rx\_data\_b1

- Number of Data Channels = 25
- Data and the Strobe Setting #3
  - Pin Direction = RX
  - I/O Type = Differential
  - Signal Type = Data
  - Strobe I/O Type = Differential
  - Strobe Name = strobe\_b2
  - Signal Name = Rx\_data\_b2
  - Number of Data Channels = 25
- PLL Input Clock Setting
  - Pin Direction = RX
  - I/O Type = Differential
  - Signal Type = Input Clock
  - Signal Name = clk
  - Number of Data Channels = 1

Figure 20: Multi-Bank RX AIOW Pin Configuration

Component Name: RX_Sync_Interface_MB									
Basic	Advanced	Pin Configuration	IO Timing Estimation	Summary					
Pin Direction	I/O Type	Signal Type	Enable Strobe	Strobe I/O Type	Strobe Name	Signal Name	Number of Data Channels		
RX	Differential	Data	<input checked="" type="checkbox"/>	Differential	strobe_b0	Rx_data_b0	25		
RX	Differential	Data	<input checked="" type="checkbox"/>	Differential	strobe_b1	Rx_data_b1	25		
RX	Differential	Data	<input checked="" type="checkbox"/>	Differential	strobe_b2	Rx_data_b2	25		
RX	Differential	Input Clock	<input type="checkbox"/>	Single-ended	Strobe_3	clk	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_4	Data_pins_4	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_5	Data_pins_5	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_6	Data_pins_6	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_7	Data_pins_7	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_8	Data_pins_8	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_9	Data_pins_9	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_10	Data_pins_10	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_11	Data_pins_11	1		

7. Check the Summary tab. It should show 158 RX pins enabled (25 pairs of data x 3 banks + 1 pair of capture clocks x 3 banks + a pair for the PLL input clock).

**Note:** In the following figure, the number of strobe pins enabled is incorrect. The correct value is 6.

Figure 21: Multi-Bank RX AIOW Summary

Component Name RX_Ssync_Intrfce_MB				
Basic	Advanced	Pin Configuration	IO Timing Estimation	Summary
<p>Interface Speed : 1800 Mbps</p> <p>PLL Input Clock Freq : 225.000 MHz</p> <p>Number of Tx Pins Enabled : 0</p> <p>Number of Rx Pins Enabled : 152</p> <p>Number of BiDir Pins Enabled : 0</p> <p>Number of Strobe Pins Enabled : 7</p> <p><b>Default Nibble groups for the data pins in the design are as follows:</b></p> <p>-&gt;RX_GROUP_0 Rx_data_b0_10 Rx_data_b0_11 Rx_data_b0_12 Rx_data_b0_13 Rx_data_b0_14 Rx_data_b0_15</p> <p>-&gt;RX_GROUP_1 Rx_data_b0_16 Rx_data_b0_17 Rx_data_b0_18 Rx_data_b0_19 Rx_data_b0_20 Rx_data_b0_21</p> <p>-&gt;RX_GROUP_10 Rx_data_b1_16 Rx_data_b1_17 Rx_data_b1_18 Rx_data_b1_19 Rx_data_b1_20 Rx_data_b1_21</p> <p>-&gt;RX_GROUP_11 strobe_b1_0 strobe_b1_1 Rx_data_b1_0 Rx_data_b1_1 Rx_data_b1_2 Rx_data_b1_3</p> <p>-&gt;RX_GROUP_12 Rx_data_b1_4 Rx_data_b1_5 Rx_data_b1_6 Rx_data_b1_7 Rx_data_b1_8 Rx_data_b1_9</p> <p>-&gt;RX_GROUP_13 Rx_data_b1_22 Rx_data_b1_23 Rx_data_b1_24 Rx_data_b1_25 Rx_data_b1_26 Rx_data_b1_27</p> <p>-&gt;RX_GROUP_14 Rx_data_b1_28 Rx_data_b1_29 Rx_data_b1_30 Rx_data_b1_31 Rx_data_b1_32 Rx_data_b1_33</p> <p>-&gt;RX_GROUP_15 Rx_data_b1_34 Rx_data_b1_35 Rx_data_b1_36 Rx_data_b1_37 Rx_data_b1_38 Rx_data_b1_39</p> <p>-&gt;RX_GROUP_16 Rx_data_b1_40 Rx_data_b1_41 Rx_data_b1_42 Rx_data_b1_43 Rx_data_b1_44 Rx_data_b1_45</p> <p>-&gt;RX_GROUP_17 clk_0 clk_1 Rx_data_b1_46 Rx_data_b1_47 Rx_data_b1_48 Rx_data_b1_49</p> <p>-&gt;RX_GROUP_18 Rx_data_b2_10 Rx_data_b2_11 Rx_data_b2_12 Rx_data_b2_13 Rx_data_b2_14 Rx_data_b2_15</p> <p>-&gt;RX_GROUP_19 Rx_data_b2_16 Rx_data_b2_17 Rx_data_b2_18 Rx_data_b2_19 Rx_data_b2_20 Rx_data_b2_21</p> <p>-&gt;RX_GROUP_2 strobe_b0_0 strobe_b0_1 Rx_data_b0_0 Rx_data_b0_1 Rx_data_b0_2 Rx_data_b0_3</p> <p>-&gt;RX_GROUP_20 strobe_b2_0 strobe_b2_1 Rx_data_b2_0 Rx_data_b2_1 Rx_data_b2_2 Rx_data_b2_3</p> <p>-&gt;RX_GROUP_21 Rx_data_b2_4 Rx_data_b2_5 Rx_data_b2_6 Rx_data_b2_7 Rx_data_b2_8 Rx_data_b2_9</p> <p>-&gt;RX_GROUP_22 Rx_data_b2_22 Rx_data_b2_23 Rx_data_b2_24 Rx_data_b2_25 Rx_data_b2_26 Rx_data_b2_27</p> <p>-&gt;RX_GROUP_23 Rx_data_b2_28 Rx_data_b2_29 Rx_data_b2_30 Rx_data_b2_31 Rx_data_b2_32 Rx_data_b2_33</p> <p>-&gt;RX_GROUP_24 Rx_data_b2_34 Rx_data_b2_35 Rx_data_b2_36 Rx_data_b2_37 Rx_data_b2_38 Rx_data_b2_39</p> <p>-&gt;RX_GROUP_25 Rx_data_b2_40 Rx_data_b2_41 Rx_data_b2_42 Rx_data_b2_43 Rx_data_b2_44 Rx_data_b2_45</p> <p>-&gt;RX_GROUP_26 Rx_data_b2_46 Rx_data_b2_47 Rx_data_b2_48 Rx_data_b2_49 -&gt;RX_GROUP_3 Rx_data_b0_4</p> <p>Rx_data_b0_5 Rx_data_b0_6 Rx_data_b0_7 Rx_data_b0_8 Rx_data_b0_9 -&gt;RX_GROUP_4 Rx_data_b0_22</p> <p>Rx_data_b0_23 Rx_data_b0_24 Rx_data_b0_25 Rx_data_b0_26 Rx_data_b0_27 -&gt;RX_GROUP_5 Rx_data_b0_28</p> <p>Rx_data_b0_29 Rx_data_b0_30 Rx_data_b0_31 Rx_data_b0_32 Rx_data_b0_33 -&gt;RX_GROUP_6 Rx_data_b0_34</p> <p>Rx_data_b0_35 Rx_data_b0_36 Rx_data_b0_37 Rx_data_b0_38 Rx_data_b0_39 -&gt;RX_GROUP_7 Rx_data_b0_40</p> <p>Rx_data_b0_41 Rx_data_b0_42 Rx_data_b0_43 Rx_data_b0_44 Rx_data_b0_45 -&gt;RX_GROUP_8 Rx_data_b0_46</p> <p>Rx_data_b0_47 Rx_data_b0_48 Rx_data_b0_49 -&gt;RX_GROUP_9 Rx_data_b1_10 Rx_data_b1_11 Rx_data_b1_12</p> <p>Rx_data_b1_13 Rx_data_b1_14 Rx_data_b1_15</p> <p><b>Next Steps:</b></p> <ol style="list-style-type: none"> <li>1. Synthesize the generated design</li> <li>2. IO planning of the design using IO planner or Nibble planner</li> </ol>				

8. Click **OK** after reviewing the settings. The IP is now customized and the Generate the Output Products prompt appears. Set the Synthesis Option to **Out of context per IP** and then click **Generate** to launch the design run for the newly generated RX core. See the following table.

Table 5: Receiver Requirements Multi-Bank

	Requirement
Component name	RX_Ssync_Intrfce_MB
Bus direction	RX_ONLY
Serialization factor	8
Interface speed (Mb/s)	1800 Mb/s
Clock data relation	Edge DDR
PLL clock source	Clock capable pin
PLL input clock frequency	225 MHz
Include PLL in core	Yes
PLL CLKOUT1	No

**Table 5: Receiver Requirements Multi-Bank (cont'd)**

	Requirement
FIFO WRCLK OUT	No
Reduce control signals	Yes
Enable delay control signals	No
Enable BLI logic	Yes
Enable deskew logic	Yes
Differential I/O Std	LVDS15
Number of banks	3
Pin configuration	As mentioned above. The configuration occupies 158 pins across all three banks.

## Receiver Design Considerations

This RX core is set up to work for a data rate of 1800 Mb/s. Also, the core is configured for LVDS15 in the reference design. This multi-bank design uses a clock capable pin as a PLL clock source and is configured for an edge aligned DDR system. An XPIO bank has 54 pins and the design uses all 54 pins in the form of 25 pairs of data pins, one pair for capture clock, and one pair of PLL input clock, in the bank receiving the input clock. The other two banks use 52 pins in the form of 25 pairs of data pins and one pair for capture clock. The PLL input clock is received by the core and then forwarded to all of the three instantiations of the XPLLs. The wizard allocates one XPLL per bank. The design constrains the ports for the receive interfaces and the wizard takes care of the placement.

## Introduction to Serialization and Data Transmission

### Data Transmission

In the reference design, the TX core sends out the strobe or the transmit clock along with the data. The data in this design is generated using the PRBS generator and the strobe is generated by feeding the pattern 01010101 to the NIBBLESlices forwarding the strobe. The data generated by the PRBS generator is fed into the TX core from the programmable logic, which follows the TX datapath through a serializer and output delay. The serializer supports 8:1, 4:1, and 2:1 serialization. This design uses 8:1 serialization. The data is transmitted through the TX data pins of the core. To understand the data flow operation inside the TX core, see the *Versal ACAP SelectIO Resources Architecture Manual* ([AM010](#)).

The TX core needs a PLL input clock that acts as the input to the XPLL (CLKIN port of the XPLL) and should only be received on the GC/XCC pin as in the RX core. This option is available if the clock capable (XCC) pin is selected for the XPLL clock source. The PLL input clock is received on the bank0\_pll\_clkln, bank1\_pll\_clkln, and bank2\_pll\_clkln ports of the TX core, which is passed to the XPLLs instantiated in the core for all three banks. The strobe should be transmitted on NIBBLESlice[0] and NIBBLESlice[1] to forward it to the RX core.

The PLL input clock to the TX core is fed to the XPLL through a bank<0/1/2>\_pll\_clkln port within each bank. In this design, the PLL input clock is provided similarly to the RX core, except that it is passed through an IBUFDS, and the single-ended clock is provided to the respective bank<0/1/2>\_pll\_clkln ports. The transmit clock can be forwarded either through the data pins or clock forwarding pins on the TX core. In both cases, a clock pattern of 01010101 needs to be fed to either the data pins or the clock forwarding pins. This results in the same output clock. The design uses the data pins to forward the transmit clock. The transmit clock is transmitted on NIBBLESlice[0] and NIBBLESlice[1] to the RX core.

## Configure and Generate a TX Advanced I/O Wizard Core

### *Generating a TX Core*

Follow these steps to generate the TX core using the AIOW. See the figures in this section for reference.

1. To start generating a core for TX, open the IP catalog and search for Advanced I/O Wizard. Double-click **Advanced I/O Wizard** to open the Customize IP window for the wizard.
2. For Component Name, enter `TX_Ssync_Intrfce_MB`, which is used in the reference design.  
**Note:** The component name should match the module name used in the top-level design.
3. Under the Basic Tab, set Application to **SOURCE SYNCHRONOUS** from the drop-down list, and set Bus Direction to **TX ONLY**.
4. On the same tab, set the following:
  - a. Interface Speed: **1800 Mb/s**.
  - b. Clock Data Relation: **Edge DDR**.
  - c. PLL Clock Source: **Fabric (Driven by BUFG)**.
  - d. PLL Input Clock Frequency: **225**.
  - e. Forwarded Clock Phase to **0**.
  - f. TX Serialization Factor: **8**.
  - g. The remaining options can be set to the default.

Figure 22: Multi-Bank RX AIOW Basic Configuration

Component Name

TX\_Sync\_Intrfce\_MB

Basic

Advanced

Pin Configuration

IO Timing Estimation

Summary

General

Application

SOURCE SYNCHRONOUS

Bus Direction

TX ONLY

CTRL\_CLK can come from local fabric, CC pin, fabric (BUFG), or PLL

BUFG

CTRL\_CLK

PLL

CLKOUT\_PHY

Fabric

Nibble

FIFO

Serializer

Output Delay

FIFO

Serializer

Output Delay

High Speed Clock Generated from Intern. Source clock can come from BUFG or CC

TX Clock Source

Data Bus (Output)

Clocking

Interface Speed(Mbps)

1800

[200 - 1800]

Clock Data Relation

Edge DDR

PLL Clock Source

Fabric(Driven by BUFG)

PLL Driven by Data Capture Clock

No

PLL Input Clock Frequency

225.000

Forwarded Clock Phase

0

Data and Control

Tx Serialization Factor

8

Rx Serialization Factor

8

3-State

Combinatorial

Enable Bit-Slip

☐

Enable Data Bit-Slip

☐

Bitslip Training Pattern

0x2C

Enable RIU Interface

☐

Include PLL in Core

☒

5. In the Advanced tab, set the following:
  - a. PLL CLKOUT1: 225
  - b. Select **REDUCE CONTROL SIGNALS**, **Enable BLI logic**, and **Enable DESKEW Logic**.  
When the BLI logic is enabled, the BLI registers between the fabric and the XPHY can be used to help with timing closure. When the deskew logic is enabled, the instantiated XPLLs are deskew enabled.
  - c. Differential I/O Std: **LVDS15**
  - d. Number of Banks: **3** (because this is a multi-bank design).

XAPP1350 (v1.0) February 4, 2021  
Application Note

Send Feedback

[www.xilinx.com](http://www.xilinx.com)  
30

Figure 23: Multi-Bank RX AIOW Advanced Configuration

Component Name: TX\_Ssync\_Intrfce\_MB

Basic | **Advanced** | Pin Configuration | IO Timing Estimation | Summary

**Optional Ports**

☐ PLL CLKOUT1 200,000

☐ FIFO WRCLK OUT

☒ REDUCE CONTROL SIGNALS

☐ ENABLE DELAY CONTROL SIGNALS

☐ ENABLE CDR DEBUG SIGNALS

☒ Enable BLI logic

☒ Enable DESKEW Logic

**Advanced Clocking** ☐ Generate CTRL CLK from PLL

**IO Standard Selection**

Differential IO Std: LVDS15

Single IO Std: NONE

**Number of Banks**

Number of Banks: 3

**Power Saving**

☐ IOB Power Saving

IOB Power Control: User Controlled

6. In the Pin Configuration tab, add three entries in the table. This entry is for the data with I/O type as differential, signal type as data, and mark the signal name as “Tx\_data\_b#”, where # is the bank number. This also accounts for the transmit clock.

- Data Setting #1
  - Pin Direction = TX
  - I/O Type = Differential
  - Signal Type = Data
  - Signal Name = Tx\_data\_b0
  - Number of Data Channels = 26
- Data Setting #2
  - Pin Direction = TX
  - I/O Type = Differential
  - Signal Type = Data
  - Signal Name = Tx\_data\_b1
  - Number of Data Channels = 26
- Data Setting #3
  - Pin Direction = TX

- I/O Type = Differential
- Signal Type = Data
- Signal Name = Tx\_data\_b2
- Number of Data Channels = 26

Figure 24: Multi-Bank RX AIOW Pin Configuration

Component Name: TX_Sync_intfce_MB									
Basic	Advanced	Pin Configuration	IO Timing Estimation	Summary					
Pin Direction	IO Type	Signal Type	Enable Strobe	Strobe IO Type	Strobe Name	Signal Name	Number of Data Channels		
TX	Differential	Data	<input type="checkbox"/>	Single-ended	Strobe_0	Tx_data_b0	26		
TX	Differential	Data	<input type="checkbox"/>	Single-ended	Strobe_1	Tx_data_b1	26		
TX	Differential	Data	<input type="checkbox"/>	Single-ended	Strobe_2	Tx_data_b2	26		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_3	Data_pms_3	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_4	Data_pms_4	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_5	Data_pms_5	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_6	Data_pms_6	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_7	Data_pms_7	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_8	Data_pms_8	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_9	Data_pms_9	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_10	Data_pms_10	1		
None	Single-ended	Data	<input type="checkbox"/>	Single-ended	Strobe_11	Data_pms_11	1		

7. Check the Summary tab. It should show 156 TX pins enabled (25 pairs of data x 3 banks + 1 pair of transmit clock x 3 banks).

Figure 25: Multi-Bank RX AIOW Summary

Component Name
TX\_Ssync\_Intrfce\_MB

Basic

Advanced

Pin Configuration

IO Timing Estimation

Summary

Interface Speed : 1800 Mbps  
PLL Input CLock Freq : 225.000 MHz  
  
Number of Tx Pins Enabled : 156  
Number of Rx Pins Enabled : 0  
Number of BiDir Pins Enabled : 0  
Number of Strobe Pins Enabled : 0  
  
**Default Nibble groups for the data pins in the design are as follows:**  
->TX\_GROUP\_0 Tx\_data\_b0\_0 Tx\_data\_b0\_1 Tx\_data\_b0\_2 Tx\_data\_b0\_3 Tx\_data\_b0\_4 Tx\_data\_b0\_5  
->TX\_GROUP\_1 Tx\_data\_b0\_6 Tx\_data\_b0\_7 Tx\_data\_b0\_8 Tx\_data\_b0\_9 Tx\_data\_b0\_10 Tx\_data\_b0\_11  
->TX\_GROUP\_10 Tx\_data\_b1\_8 Tx\_data\_b1\_9 Tx\_data\_b1\_10 Tx\_data\_b1\_11 Tx\_data\_b1\_12 Tx\_data\_b1\_13  
->TX\_GROUP\_11 Tx\_data\_b1\_14 Tx\_data\_b1\_15 Tx\_data\_b1\_16 Tx\_data\_b1\_17 Tx\_data\_b1\_18 Tx\_data\_b1\_19  
->TX\_GROUP\_12 Tx\_data\_b1\_20 Tx\_data\_b1\_21 Tx\_data\_b1\_22 Tx\_data\_b1\_23 Tx\_data\_b1\_24 Tx\_data\_b1\_25  
->TX\_GROUP\_13 Tx\_data\_b1\_26 Tx\_data\_b1\_27 Tx\_data\_b1\_28 Tx\_data\_b1\_29 Tx\_data\_b1\_30 Tx\_data\_b1\_31  
->TX\_GROUP\_14 Tx\_data\_b1\_32 Tx\_data\_b1\_33 Tx\_data\_b1\_34 Tx\_data\_b1\_35 Tx\_data\_b1\_36 Tx\_data\_b1\_37  
->TX\_GROUP\_15 Tx\_data\_b1\_38 Tx\_data\_b1\_39 Tx\_data\_b1\_40 Tx\_data\_b1\_41 Tx\_data\_b1\_42 Tx\_data\_b1\_43  
->TX\_GROUP\_16 Tx\_data\_b1\_44 Tx\_data\_b1\_45 Tx\_data\_b1\_46 Tx\_data\_b1\_47 Tx\_data\_b1\_48 Tx\_data\_b1\_49  
->TX\_GROUP\_17 Tx\_data\_b1\_50 Tx\_data\_b1\_51 Tx\_data\_b2\_0 Tx\_data\_b2\_1 Tx\_data\_b2\_2 Tx\_data\_b2\_3  
->TX\_GROUP\_18 Tx\_data\_b2\_4 Tx\_data\_b2\_5 Tx\_data\_b2\_6 Tx\_data\_b2\_7 Tx\_data\_b2\_8 Tx\_data\_b2\_9  
->TX\_GROUP\_19 Tx\_data\_b2\_10 Tx\_data\_b2\_11 Tx\_data\_b2\_12 Tx\_data\_b2\_13 Tx\_data\_b2\_14 Tx\_data\_b2\_15  
->TX\_GROUP\_2 Tx\_data\_b0\_12 Tx\_data\_b0\_13 Tx\_data\_b0\_14 Tx\_data\_b0\_15 Tx\_data\_b0\_16 Tx\_data\_b0\_17  
->TX\_GROUP\_20 Tx\_data\_b2\_16 Tx\_data\_b2\_17 Tx\_data\_b2\_18 Tx\_data\_b2\_19 Tx\_data\_b2\_20 Tx\_data\_b2\_21  
->TX\_GROUP\_21 Tx\_data\_b2\_22 Tx\_data\_b2\_23 Tx\_data\_b2\_24 Tx\_data\_b2\_25 Tx\_data\_b2\_26 Tx\_data\_b2\_27  
->TX\_GROUP\_22 Tx\_data\_b2\_28 Tx\_data\_b2\_29 Tx\_data\_b2\_30 Tx\_data\_b2\_31 Tx\_data\_b2\_32 Tx\_data\_b2\_33  
->TX\_GROUP\_23 Tx\_data\_b2\_34 Tx\_data\_b2\_35 Tx\_data\_b2\_36 Tx\_data\_b2\_37 Tx\_data\_b2\_38 Tx\_data\_b2\_39  
->TX\_GROUP\_24 Tx\_data\_b2\_40 Tx\_data\_b2\_41 Tx\_data\_b2\_42 Tx\_data\_b2\_43 Tx\_data\_b2\_44 Tx\_data\_b2\_45  
->TX\_GROUP\_25 Tx\_data\_b2\_46 Tx\_data\_b2\_47 Tx\_data\_b2\_48 Tx\_data\_b2\_49 Tx\_data\_b2\_50 Tx\_data\_b2\_51  
->TX\_GROUP\_3 Tx\_data\_b0\_18 Tx\_data\_b0\_19 Tx\_data\_b0\_20 Tx\_data\_b0\_21 Tx\_data\_b0\_22 Tx\_data\_b0\_23  
->TX\_GROUP\_4 Tx\_data\_b0\_24 Tx\_data\_b0\_25 Tx\_data\_b0\_26 Tx\_data\_b0\_27 Tx\_data\_b0\_28 Tx\_data\_b0\_29  
->TX\_GROUP\_5 Tx\_data\_b0\_30 Tx\_data\_b0\_31 Tx\_data\_b0\_32 Tx\_data\_b0\_33 Tx\_data\_b0\_34 Tx\_data\_b0\_35  
->TX\_GROUP\_6 Tx\_data\_b0\_36 Tx\_data\_b0\_37 Tx\_data\_b0\_38 Tx\_data\_b0\_39 Tx\_data\_b0\_40 Tx\_data\_b0\_41  
->TX\_GROUP\_7 Tx\_data\_b0\_42 Tx\_data\_b0\_43 Tx\_data\_b0\_44 Tx\_data\_b0\_45 Tx\_data\_b0\_46 Tx\_data\_b0\_47  
->TX\_GROUP\_8 Tx\_data\_b0\_48 Tx\_data\_b0\_49 Tx\_data\_b0\_50 Tx\_data\_b0\_51 Tx\_data\_b1\_0 Tx\_data\_b1\_1  
->TX\_GROUP\_9 Tx\_data\_b1\_2 Tx\_data\_b1\_3 Tx\_data\_b1\_4 Tx\_data\_b1\_5 Tx\_data\_b1\_6 Tx\_data\_b1\_7  
  
**Next Steps:**  
1. Synthesize the generated design  
2. IO planning of the design using IO planner or Nibble planner

- Click **OK** after reviewing the settings. The IP is now customized and the Generate the Output Products prompt appears. Set the Synthesis Option to **Out of context per IP** and then click **Generate** to launch the design run for the newly generated TX core.

Table 6: Transmitter Requirements Multi-Bank

	Requirement
Component name	TX_Ssync_Intrfce_MB
Bus direction	TX_ONLY
Serialization factor	8
Interface speed (Mb/s)	1800 Mb/s
Clock data relation	Edge DDR
PLL clock source	Fabric (driven by BUFG)
Forwarded clock phase	0
PLL input clock frequency	225 MHz
Include PLL in core	Yes

**Table 6: Transmitter Requirements Multi-Bank (cont'd)**

	Requirement
PLL CLKOUT1	No
FIFO WRCLK OUT	No
Reduce control signals	Yes
Enable delay control signals	No
Enable BLI logic	Yes
Enable deskew logic	Yes
Differential I/O Std	LVDS15
Number of banks	3
Pin configuration	The number of data channels is set to 26 pairs. One pair is reserved for the transmit clock.

## Transmitter Design Considerations

This TX core is set up for a data rate of 1800 Mb/s. Also, the core is configured and tested for LVDS15 in this reference design. The PLL input clock to the TX core is fed to the XPLL through a bank<0/1/2>\_pll\_clkln port within each bank. In the design, the PLL input clock is provided similarly to the RX core, except that it is passed through an IBUFDS, and the single-ended clock is provided to the respective bank<0/1/2>\_pll\_clkln ports. An XPIO bank has 54 pins and the design uses 52 pins in the form of 25 pairs of data pins and one pair of strobe for each of the three bank instances. The RX core has a pair of pins reserved for the PLL input clock, and consequently, the TX core leaves an equivalent pair unused. The design constrains the ports for the transmit interfaces and the wizard takes care of the placement.

## Top-Level using TX and RX Advanced I/O Wizard Cores

The top-level design file (`toplevel_mb.sv`) includes the `toplevel_mb` module. This module helps connect the interfaces such as clocks, debug ports, I/O ports, etc., with the appropriate sources. The top-level design houses the instantiation of both the RX and the TX cores. To test the design, the PRBS patterns from the custom PRBS generator provided in the design suite can be used to generate and check the received data.

The PRBS generator generates the 8-bit data for each pair and feeds it to the TX core, which in turn transmits it through the TX pins. The PRBS generator also houses an error injecting mechanism. The data is received on the I/O ports of the RX core through external loopback. The RX core forwards the deserialized data through the PHY to the programmable logic. This deserialized data is then fed into the PRBS checker to check for any failures.

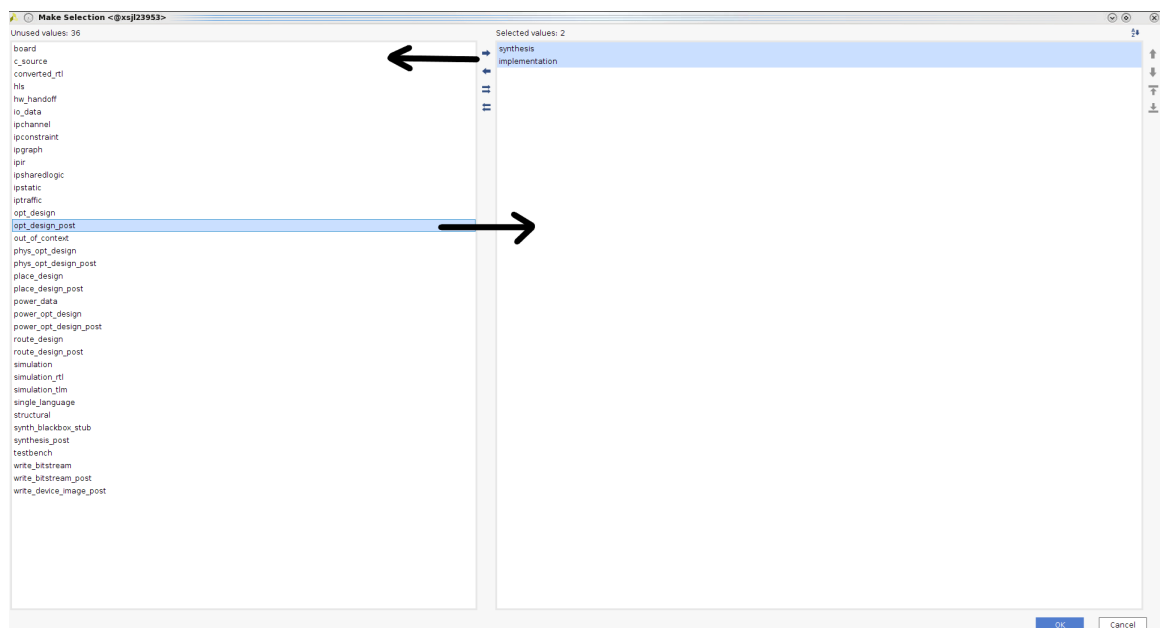
The top-level design files include two constraints files. The `toplevel_mb.xdc` file is used to create clocks for the design and assign a location to all the I/O ports in the design. The `attributes_mb.xdc` file is provided as a placeholder for any attributes to be added for non-default bank instances. The current design is provided as an empty file because it does not have any constraints for non-default bank instances. This file must be marked for post-opt usage. The

current release of the Vivado tools fails to recognize any bank instances before the optimization stage, and consequently, the design uses two separate constraints files. As a result, any constraints to modify the attributes of the non-default bank instance are recognized, but the ones for the non-default bank instances are ignored. Consequently, the second constraint file needs to be added and marked to be used for post-optimization.

To set the Xilinx design constraints (XDC) to be used in post\_opt, open the Vivado tools and follow these steps.

1. Select the XDC file, right-click, and click **Source File Properties**.
2. Select **Properties** from the Source File Properties window.
3. Navigate to **USED IN**. Move `opt_design_post` from left to right and `synthesis` and `implementation` from right to left as shown in the following figure.

Figure 26: Setting the XDC to be Used in Post\_Opt



4. Click **OK**.

The reference design constrains the design to optimally support high data rates. This file is used to create clocks for the design, assign locations or pins to all the I/O ports in the design, and set attributes if needed. The reference design constrains the design to optimally support high data rates. The user must constrain the TX and RX ports. The Vivado tools can assign the XPHY nibbles to the XPHY sites.

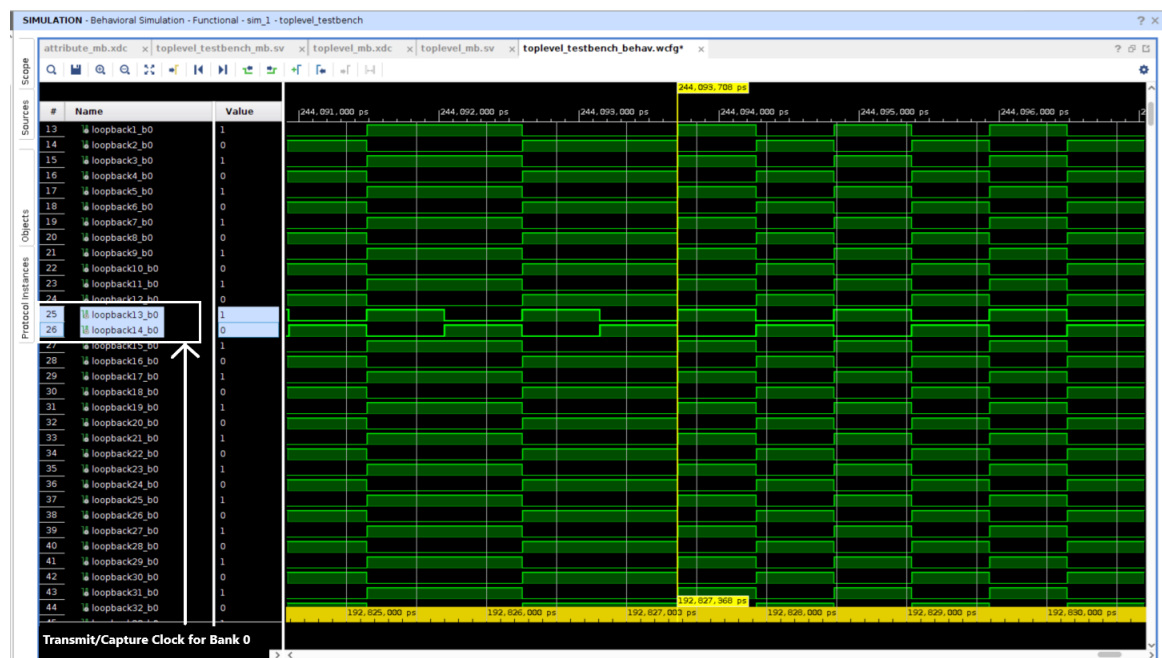
In the reference design, bank 706, bank 707, and bank 708 for part xcvc1902-vsua2197-2MP-e-S-eS1 are used for the TX core and bank 703, bank 704, and bank 705 are used for the RX core. The RX core has one pair reserved for the PLL input clock per bank, the design assigns `IO_L9P_GC_XCC_N3P0_M1P72_704_BE31` and `IO_L9N_GC_XCC_N3P1_M1P73_704_BD32` in the constraint file for bank 704. Similarly, `IO_L6P_GC_XCC_N2P0_M1P66_704_BC31` and `IO_L6N_GC_XCC_N2P1_M1P67_704_BC30` are assigned to the capture clock. The corresponding pins in other banks for both the TX core and RX core are assigned similarly.

At this point all the files are added to the project and both the RX and TX cores are generated. The design is ready to be synthesized and implemented.

## Simulation

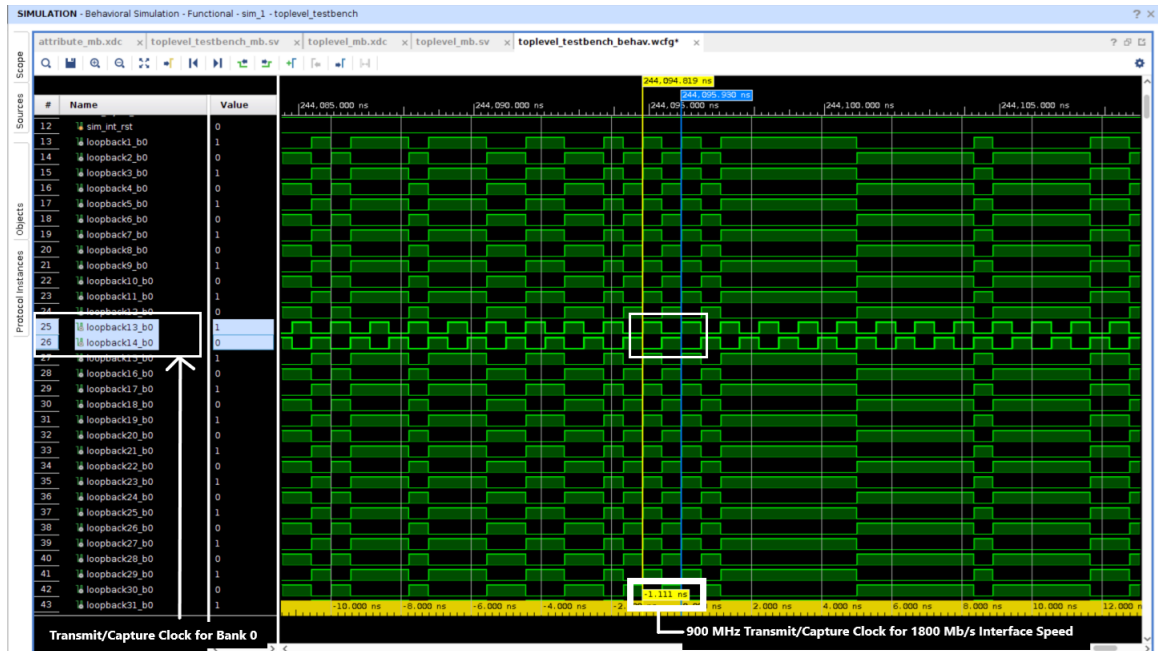
The design uses the `toplevel_testbench_mb.sv` file to create a simple test bench. This test bench connects the TX core to the RX core via `loopback<num>_b<0/1/2>` connections (wires). The transmit clock is transmitted on `loopback13_b<0/1/2>` and `loopback14_b<0/1/2>` loopback wires. All the other loopback connections are used to transmit and receive the data. The clock-to-data relationship is edge aligned as shown in the following figure captured from the simulation.

Figure 27: Clock-to-Data Relation - Edge Alignment



The test bench provides the necessary clock and resets to the design and triggers its operation. The PLL input clock is provided at 4.444 ns (225 MHz) to the RX and TX cores. The transmit/capture clock toggles at 900 Mb/s (1.111 ns period) as shown in the following figure. Because the system is double data rate, the interface operates at 1800 Mb/s.

Figure 28: Multi-Bank Design - Interface Speed 1800 Mb/s



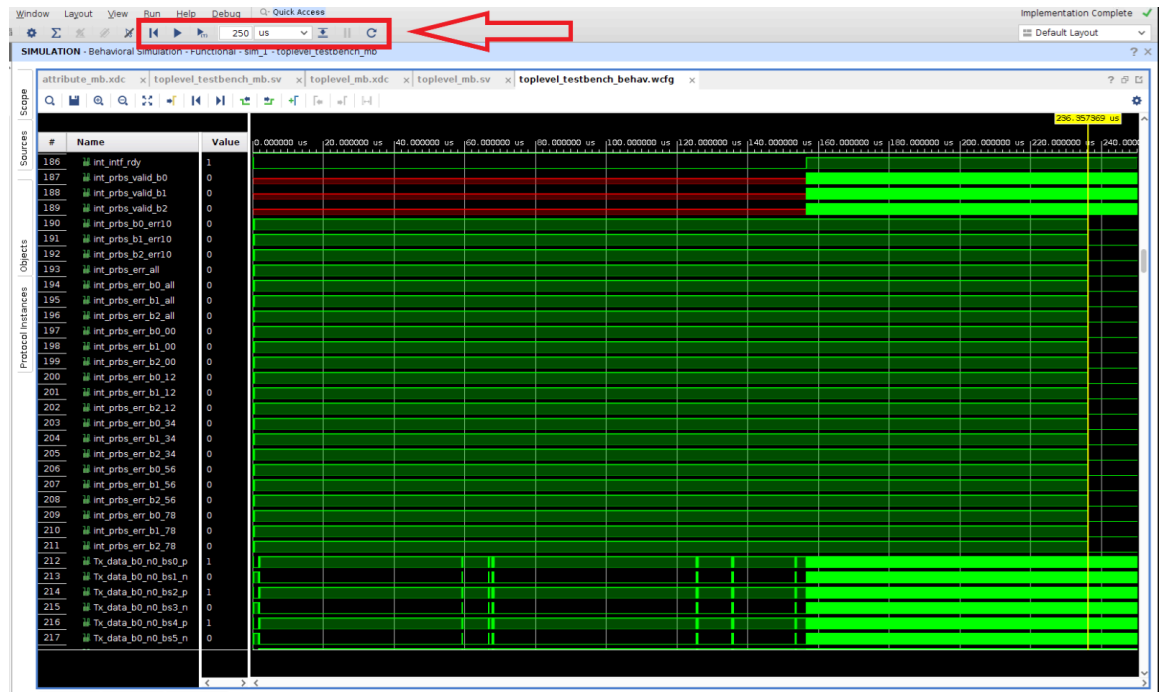
The Vivado 2020.1 tools allow only behavioral simulation when using the AIOW.

## Simulating the Design

The design is tested with the Vivado Simulator 2020.1. This section describes how to launch the simulation. Assuming the Vivado project is already created for the design, follow these steps to simulate the design.

1. Click **Run Simulation** under Simulation from the Flow Navigator.
2. From the listed options, select **Run Behavioral Simulation**, which elaborates the design and launches the simulation.
3. The run time by default is 1 ns for the simulation. It takes about 240  $\mu$ s for the `intf_rdy` to be asserted for the RX core. Only after its assertion and the next `Prbs_Valid`, should the data be compared. Consequently, launch the simulation for a duration of more than 240  $\mu$ s using the box highlighted in the following figure. A waveform config file `toplevel_testbench_behav.wcfg` is provided under the `Testbench` directory to add a list of signals for the simulation.

Figure 29: Multi-Bank Design - Changing the Simulation Time



- Once the behavioral simulation is finished, check for any errors by observing error flags for the PRBS generator and checker. For example, `int_prbs_b<0/1/2>_err<num>` reports the errors for each instantiation of the PRBS generator and checker. `Int_prbs_b0_err24` denotes the error flag for the PRBS module instantiated for NIBBLE[2], NIBBLESlice[4] of bank instance 0. `Int_prbs_err_b<0/1/2>_00` denotes the error for any NIBBLESlices on NIBBLE[0]. `Int_prbs_err_b<0/1/2>_12` denotes the error for any NIBBLESlices on NIBBLE[1] and NIBBLE[2]. `Int_prbs_err_b0_all` denotes an error on any NIBBLESlice across all the nibbles in bank instance 0. `Int_prbs_err_all` denotes an error on any NIBBLESlice across all the nibbles in any of the bank instances.

## Reference Design

Download the [reference design files](#) from the Xilinx website. Unzip this file and browse into the top-level folder `xapp1350`. Check the `Multi_Bank_Ssync_Loopback_Design` folder, which has all the source files under `Sources`, `Constraints`, and `Testbench`.

Table 7: Source Files for Multi-Bank Design

Folder	File Name	Description
Sources	<code>toplevel_mb.sv</code>	The top-level design file that instantiates the TX and RX cores and connects the design
	<code>Prbs_RxTx.vhd</code>	Outer wrapper for the custom PRBS generator and checker
	<code>Prbs_Any.vhd</code>	Contains the modules for the PRBS generator and checker

**Table 7: Source Files for Multi-Bank Design (cont'd)**

Folder	File Name	Description
Constraints	<code>toplevel_mb.xdc</code>	The constraints file to assign pin locations to ports of the design for all bank instances and set the necessary attributes for bank instance 0 (Default bank instance)
	<code>attributes_mb.xdc</code>	The constraints file to set the attributes for bank instances 1 and 2 (Non-default bank instances). This file is marked to be used only for post optimization.
Test Bench	<code>toplevel_testbench_mb.sv</code>	The top-level test bench file to test the design
	<code>toplevel_testbench_behav.wcfg</code>	Waveform configuration file

## Reference Design Matrix

The following checklist indicates the procedures used for the provided reference design.

**Table 8: Reference Design Matrix**

Parameter	Description
General	
Developer name	Xilinx
Target devices	Versal ACAP
Source code provided?	Yes
Source code format (if provided)	Verilog and VHDL
Design uses code or IP from existing reference design, application note, 3rd party or Vivado software? If yes, list.	Yes. Uses the Advanced I/O Wizard from the IP catalog.
Simulation	
Functional simulation performed	Yes
Timing simulation performed?	No
Test bench provided for functional and timing simulation?	Yes
Test bench format	Verilog
Simulator software and version	Vivado simulator 2020.1
SPICE/IBIS simulations	No
Implementation	
Synthesis software tools/versions used	Vivado synthesis 2020.1
Implementation software tool(s) and version	Vivado 2020.1 implementation
Static timing analysis performed?	Yes
Hardware Verification	
Hardware verified?	No
Platform used for verification	N/A

## Conclusion

This application note confirms the use of the Advanced I/O Wizard for the source synchronous application. It uses two reference designs, a single-bank source synchronous design and a multi-bank source synchronous design to showcase how the wizard instantiates the bank instances. It can further be expanded to modify various attributes of the XPHY nibbles and XPLL for further applications.

## References

These documents provide supplemental material useful with this guide:

1. *Versal Architecture and Product Data Sheet: Overview* ([DS950](#))
2. *Versal ACAP Clocking Resources Architecture Manual* ([AM003](#))
3. *Versal ACAP SelectIO Resources Architecture Manual* ([AM010](#))
4. *Advanced I/O Wizard LogiCORE IP Product Guide* ([PG320](#))

## Revision History

The following table shows the revision history for this document.

Section	Revision Summary
02/04/2021 Version 1.0	
Initial release.	N/A

## Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>.

**AUTOMOTIVE APPLICATIONS DISCLAIMER**

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

**Copyright**

© Copyright 2021 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Spartan, Versal, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.