XAPP1184 (v2.0) February 19, 2014

PIPE Mode Simulation Using Integrated Endpoint PCI Express Block in Gen2 x8 and Gen3 x8 Configurations

Author: K. Murali Govinda Rao and A. V. Anil Kumar

Summary

The verification of designs involving high speed serial protocols such as PCI Express® can be complex and time consuming. Many verification projects use third-party bus functional models (BFMs) to reduce the complexity of the verification process and to speed up the time spent running the actual simulation.

Because of the complexity of serial transceivers, a significant number of processor cycles are consumed in simulation resulting in long simulation times. In addition, serial transceivers typically have little impact on the behavior of the upper PCI Express layers functionality. With this in mind, many verification projects bypass the serial transceivers for much of their verification and only simulate using transceivers to validate the design(s) at the end of a project.

A specification for interfacing between the PCI Express block and the serial transceivers is maintained by the PCISIG–called the PHY Interface for PCI Express (PIPE). Most PCI Express BFMs allow the device under test to be connected to a PIPE interface rather than a serial interface, effectively bypassing the transceivers, and greatly speeding up simulation times.

This application note describes integrating the PCI-Xactor kit BFM from Avery Design Systems as a root complex with either the Xilinx 7 series FPGAs Integrated PCI Express block operating as a x8 Gen2 Endpoint, or the Virtex®-7 FPGA Gen3 Integrated Block operating as a x8 Gen3 Endpoint. This Avery Design Systems BFM kit v1.0 has limited functionality. For further usage, contact Avery Design Systems [Ref 4].

Introduction

The PHY Interface for the PCI Express Architecture (PIPE) is intended to enable the development of functionally equivalent PCI Express PHYs. The PCI Express PIPE 2.0 and PIPE 3.0 specification [Ref 1] defines the functionality that must be incorporated in a PIPE compliant PHY, and defines a standard interface between the PHY and a Media Access Layer (MAC) contained in a typical PCI Express block.

This application note provides a methodology to connect the PIPE interface of the Avery Design System PCI X-actor BFM (in root complex mode) to the PIPE interface of a Xilinx 7 series FPGAs Integrated PCI Express Endpoint Block. When configured with the proper options, the Xilinx PCI Express Endpoint has PIPE ports at the core top level. These ports can be connected to the X-actor RC BFM to bypass transceivers during simulation.

While this application note demonstrates specific connections to the X-actor BFM from Avery Design Systems, it can also serve as a model on how other third-party BFMs can be connected to the Integrated Endpoint PCI Express Block through the PIPE interface.

PIPE Use Model

The PCI Express PIPE 2.0 and PIPE 3.0 specifications [Ref 1] provides information about combining multiple PIPEs for multi-lane designs. The PIPE mode simulation uses a model from the Avery Design Systems BFM as a Root Complex (RC) and Xilinx Integrated PCI Express Endpoint block (EP) for an 8-lane design operating at the Gen2 rate (Figure 1). For more details about shared signals and per-lane signals, see the PCI Express PIPE 2.0 and PIPE 3.0 specification [Ref 1].

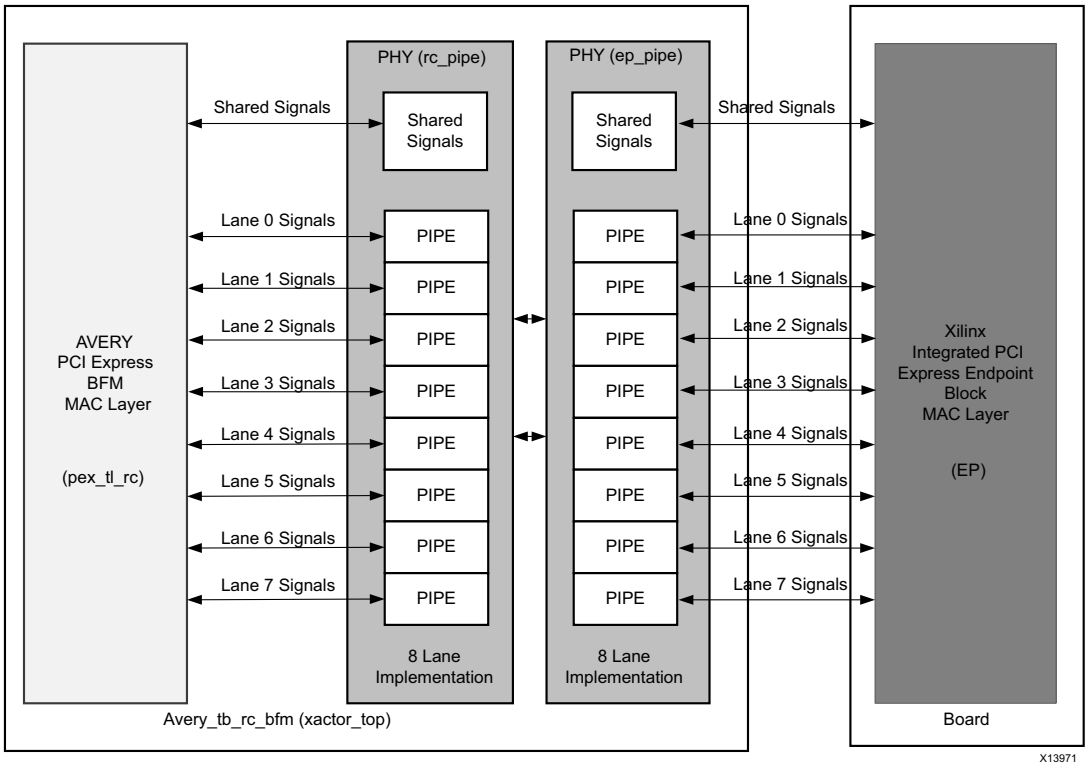

Figure 1: **Block Diagram**

Gen2 x8 Configuration

Xilinx Endpoint PIPE Port Descriptions

The PIPE signals on the Xilinx EP instantiation are encapsulated in buses that are available at the top level of the core. Each lane has one input bus (`pipe_rx_0_sigs[24:0]`, `pipe_rx_1_sigs[24:0]`,...) and one output bus (`pipe_tx_0_sigs[22:0]`, `pipe_tx_1_sigs[22:0]`,...). There are two common bus signals for providing commands, clocks, and status signaling (`common_commands_in[3:0]`, `common_commands_out[11:0]`). Table 1 and Table 2 describe the PIPE bus signals available at the top level of the core and their corresponding mapping inside the EP core (`pcie_top`) PIPE signals.

Table 1: **Common In/Out Commands and Endpoint PIPE Signals Mappings**

In Commands	Endpoint PIPE Signals Mapping	Out Commands	Endpoint PIPE Signals Mapping
common_commands_in[0]	pipe_clk(1)	common_commands_out[5:0]	pl_ltssm_state
common_commands_in[1]	user_clk2(2)	common_commands_out[6]	pipe_tx_rcvr_det_gt
common_commands_in[2]	user_clk(3)	common_commands_out[7]	pipe_tx_rate_gt
common_commands_in[3]	phy_rdy_n(4)	common_commands_out[8]	pipe_tx_deemph_gt
		common_commands_out[11:9]	pipe_tx_margin_gt

Notes:

1. `pipe_clk` is a regenerated clock based on the phase of the AveryDesign Systems BFM clock signal aclk250M. When the link speed is Gen1, `pipe_clk` is 125 MHz. In Gen2, `pipe_clk` is 250 MHz.
2. `user_clk2` is a Xilinx PCI Express Endpoint clock. In Gen2 x8 configuration, `user_clk2` = 250 MHz.
3. `user_clk` is a Xilinx PCI Express Endpoint clock. In Gen2 x8 configuration, `user_clk` = 500 MHz.
4. `phy_rdy_n` should be asserted for at least 20 ns. The required logic is added in the `board.v` file of the reference example design.

Table 2: **Input/Output Bus with Endpoint PIPE Signals Mapping**

Input Bus	Endpoint PIPE Signals Mapping	Output Bus	Endpoint PIPE Signals Mapping
pipe_rx_0_sigs[15: 0] pipe_rx_0_sigs[17:16] pipe_rx_0_sigs[18] pipe_rx_0_sigs[19] pipe_rx_0_sigs[22:20] pipe_rx_0_sigs[23] pipe_rx_0_sigs[24]	pipe_rx0_data_gt pipe_rx0_char_is_k_gt pipe_rx0_valid_gt pipe_rx0_chanisaligned_gt pipe_rx0_status_gt pipe_rx0_phy_status_gt pipe_rx0_elec_idle_gt	pipe_tx_0_sigs[15: 0] pipe_tx_0_sigs[17:16] pipe_tx_0_sigs[18] pipe_tx_0_sigs[19] pipe_tx_0_sigs[20] pipe_tx_0_sigs[22:21]	pipe_tx0_data_gt pipe_tx0_char_is_k_gt pipe_rx0_polarity_gt pipe_tx0_compliance_gt pipe_tx0_elec_idle_gt pipe_tx0_powerdown_gt

Avery Design Systems BFM PIPE Port Descriptions

Xilinx PCI Express Endpoint PIPE ports must be connected with the Avery Design Systems BFM PIPE ports. Connections can be made hierarchically in the simulation top file `board.v` of the Xilinx EP, or they can be connected directly by instantiating Xilinx EP in the `avery_tb_rc_bfm` top wrapper. Table 3 maps corresponding signals between Avery Design Systems BFM and the Xilinx Endpoint core.

Table 3: **Avery Design Systems BFM Port Mapping**

Avery Port Name	Endpoint Core PIPE Signal Name	Bus PIPE Port Name
aTxDetectRx	pipe_tx_rcvr_det_gt	common_commands_out[6]
aRate	pipe_tx_rate_gt	common_commands_out[7]
aTxDeemph	pipe_tx_deemph_gt	common_commands_out[8]
aTxMargin	pipe_tx_margin_gt	common_commands_out[11:9]
aclk250M(1)	pipe_clk	common_commands_in[0]
apipe_txd[15:0]	pipe_tx0_data_gt	pipe_tx_0_sigs[15:0]
apipe_txk[1:0]	pipe_tx0_char_is_k_gt	pipe_tx_0_sigs[17:16]
aRxPolarity[0]	pipe_rx0_polarity_gt	pipe_tx_0_sigs[18]
aTxCompliance[0]	pipe_tx0_compliance_gt	pipe_tx_0_sigs[19]
aTxElecIdle[0]	pipe_tx0_elec_idle_gt	pipe_tx_0_sigs[20]
aPowerDown	pipe_tx0_powerdown_gt	pipe_tx_0_sigs[22:21]
apipe_rxd[15:0]	pipe_rx0_data_gt	pipe_rx_0_sigs[15:0]
apipe_rxk[1:0]	pipe_rx0_char_is_k_gt	pipe_rx_0_sigs[17:16]
aRxValid[0]	pipe_rx0_valid_gt	pipe_rx_0_sigs[18]
aRxStatus[2:0]	pipe_rx0_status_gt	pipe_tx_0_sigs[22:20]
aPhyStatus	pipe_rx0_phy_status_gt	pipe_tx_0_sigs[23]
aRxElecIdle[0]	pipe_rx0_elec_idle_gt	pipe_tx_0_sigs[24]
	pipe_rx0_chanisaligned_gt(2) pl_ltssm_state(2)	pipe_rx_0_sigs[19] common_commands_out[5:0]

Notes:

1. `aclk250M` is expected to change from 125 MHz to 250 MHz when the link has trained to Gen2 speed.
2. These signals can be ignored.

Test Bench Integration

These steps are required to integrate the Avery Design Systems BFM with the Xilinx Integrated PCI Express Endpoint block:

1. Download the Avery Design System BFM kit and Xilinx reference example design of this application note. For details, see Libraries, and Reference Design Files. Extract the files using `tar xfz <tarfile>`. For the BFM kit, follow the Avery Design Systems BFM integration procedure for setting environment variables, paths and simulation instructions located in the README file under `avery_bfm_kit`.
2. In the Xilinx Vivado® integrated design environment, create a new design targeting a device that contains a Gen2 PCI Express block. Target a **KC705** board, and ensure that the target language is set to **Verilog** in the Vivado project settings.
3. In the Flow Navigator, select **IP Catalog**.
4. Select the **7 Series Integrated Block for PCI Express** to customize the PCI Express core.
5. Customize these settings:
   - Lane width = **X8**.
   - Link speed = **5.0 GT/s**.
   - Ensure that **Enable External PIPE Interface** is selected.

   Note: See the *Xilinx Logicore IP 7 Series FPGAs Integrated Block for PCI Express* (PG054) [Ref 2] for detailed information about customizing the core for Gen2 mode of operation, and see the *Xilinx Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 3] for detailed information about using logic simulation tools in the Vivado Design Suite.

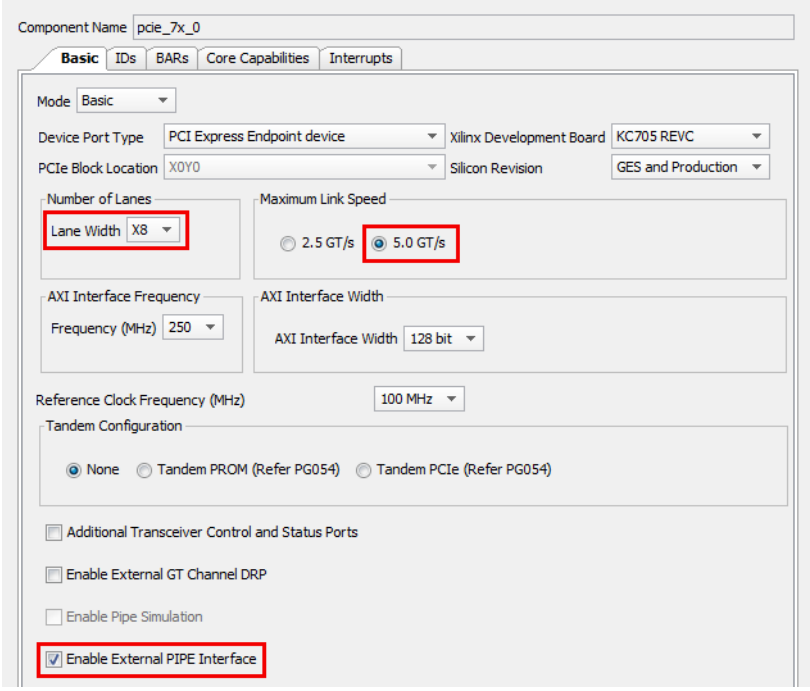

Figure 2: **Customization Options**

6. After the PCIe IP is generated, right-click the IP and select **Open IP Example Design**.
7. Select **QuestaSim** as Target simulator in Simulation Project Settings.
8. In the Flow Navigator, select **Run Simulation**.
9. Type the `run -all` command at the VSIM console to run the Xilinx RP and Xilinx EP in PIPE mode simulation.

10. Identify the board.do file from the simulation directory of the Xilinx example design (. . . /sim_1/behav/), and add the Avery Design Systems compiled simulation libraries in the vsim command as instructed in the README file of Avery Design Systems BFM kit.
11. Add the copy command in board.do file to copy the Avery compiled library to the simulation directory /sim_1/behav/. See the Avery README file for more information.
12. Open the board.v file and connect the PIPE signals to and from the Xilinx EP and Avery RC BFM, as described in Table 1, Table 2, and Table 3. Comment out the Xilinx RP and phy_sig_gen modules instantiation.
13. In the board.v file, connect the apipe_reset signal to the sys_rst_n signal, and generate the phy_rdy_n signal which should be asserted for at least 20 ns.
14. Generate phase aligned (posedge) pipe_clk, clk_250_mhz (user_clk2), clk_500_mhz (user_clk) clocks from Avery Design Systems BFM clock aclk250M in the sys_clk_gen_ds.v file.
15. Add these three clocks as output ports and update the sys_clk_gen_ds.v instantiation in the board.v file.

Gen3 x8 Configuration

Xilinx Endpoint PIPE Port Descriptions

The PIPE signals on the Xilinx EP instantiation are encapsulated in buses that are available at the top level of the core. Each lane has one input bus (pipe_rx_0_sigs[83:0], pipe_rx_1_sigs[83:0],...) and one output bus (pipe_tx_0_sigs[69:0], pipe_tx_1_sigs[69:0],...). There are two common bus signals for providing commands, clocks, and status signaling (common_commands_in[25:0], common_commands_out[16:0]). Table 4 and Table 5 describe the PIPE bus signals available at the top level of the core and their corresponding mapping inside the EP core (pcie_top) PIPE signals.

Table 4: **Common In/Out Commands and Endpoint PIPE Signals Mappings**

In Commands	Endpoint PIPE Signals Mapping	Out Commands	Endpoint PIPE Signals Mapping
common_commands_in[0]	pipe_clk(1)	common_commands_out[0]	pipe_tx_rcvr_det_gt
common_commands_in[1]	core_clk(2)	common_commands_out[2:1]	pipe_tx_rate_gt
common_commands_in[2]	user_clk(3)	common_commands_out[3]	pipe_tx_deemph_gt
common_commands_in[3]	rec_clk(4)	common_commands_out[6:4]	pipe_tx_margin_gt
common_commands_in[4]	phy_rdy(5)	common_commands_out[7]	pipe_tx_swing_gt
common_commands_in[5]	mmcm_lock(6)	common_commands_out[8]	pipe_tx_reset_gt
common_commands_in[11:6]	pipe_tx_eqfs(7)	common_commands_out[16:9]	pipe_tx_slide_gt
common_commands_in[17:12]	pipe_rx_eqlf(8)		
common_commands_in[25:18]	pipe_rx_syncdone(9)		

Notes:

1. pipe_clk is a regenerated clock based on the phase of the AveryDesign Systems BFM clock signal aclk250M. When the link speed is Gen1, pipe_clk is 125 MHz. In Gen3, pipe_clk is 250 MHz.
2. core_clk is a Xilinx PCI Express Endpoint clock. In Gen3 x8 configuration, core_clk = 500 MHz.
3. user_clk is a Xilinx PCI Express Endpoint clock. In Gen3 x8 configuration, user_clk = 250 MHz.
4. rec_clk is a Xilinx PCI Express Endpoint clock. Tie it to the pipe_clk signal.
5. phy_rdy should be asserted after 10 µs. The required logic is added in the board.v file of the reference example design.
6. mmcm_lock can be asserted after 10 ns. The required logic is added in the board.v file of the reference example design.
7. Assign 6'd40 to pipe_tx_eqfs.
8. Assign 6'd15 to pipe_tx_eqlf.
9. Assign 8'd0 to pipe_rx_syncdone.

Table 5: **Input/Output Bus with Endpoint PIPE Signals Mapping**

Input Bus	Endpoint PIPE Signals Mapping	Output Bus	Endpoint PIPE Signals Mapping
pipe_rx_0_sigs[31:0]	pipe_rx0_data_gt	pipe_tx_0_sigs[31: 0]	pipe_tx0_data_gt
pipe_rx_0_sigs[33:32]	pipe_rx0_char_is_k_gt	pipe_tx_0_sigs[33:32]	pipe_tx0_char_is_k_gt
pipe_rx_0_sigs[34]	pipe_rx0_data_valid_gt	pipe_tx_0_sigs[34]	pipe_tx0_elec_idle_gt
pipe_rx_0_sigs[35]	pipe_rx0_elec_idle_gt	pipe_tx_0_sigs[35]	pipe_tx0_data_valid_gt
pipe_rx_0_sigs[36]	pipe_rx0_start_block_gt	pipe_tx_0_sigs[36]	pipe_tx0_start_block_gt
pipe_rx_0_sigs[38:37]	pipe_rx0_syncheader_gt	pipe_tx_0_sigs[38:37]	pipe_tx0_syncheader_gt
pipe_rx_0_sigs[41:39]	pipe_rx0_status_gt	pipe_tx_0_sigs[39]	pipe_tx0_polarity_gt
pipe_rx_0_sigs[42]	pipe_rx0_valid_gt	pipe_tx_0_sigs[41:40]	pipe_tx0_powerdown_gt
pipe_rx_0_sigs[43]	pipe_rx0_phy_status_gt	pipe_tx_0_sigs[43:42]	pipe_tx0_eqcontrol_gt
pipe_rx_0_sigs[44](1)	pipe_rx0_eqdone_gt	pipe_tx_0_sigs[47:44](7)	pipe_tx0_eqpreset_gt
pipe_rx_0_sigs[62:45](2)	pipe_rx0_eqcoeff_gt	pipe_tx_0_sigs[53:48](7)	pipe_tx0_eqdeemph_gt
pipe_rx_0_sigs[80:63](3)	pipe_rx0_eqlp_new_txcoef_forpreset_gt	pipe_tx_0_sigs[55:54]	pipe_rx0_eqcontrol_gt
pipe_rx_0_sigs[81](4)	pipe_rx0_eqlp_lffs_sel_gt	pipe_tx_0_sigs[58:56](7)	pipe_rx0_eqpreset_gt
pipe_rx_0_sigs[82](5)	pipe_rx0_eqlp_adaptdone_gt	pipe_tx_0_sigs[64:59](7)	pipe_rx0_eqlp_lffs_gt
pipe_rx_0_sigs[83](6)	pipe_rx0_eqdone_gt	pipe_tx_0_sigs[68:65](7)	pipe_rx0_eqlp_txpreset_gt
		pipe_tx_0_sigs[69]	pipe_tx0_compliance_gt

Notes:

1. Asserted whenever `pipe_tx0_eqcontrol_gt` (`pipe_tx_0_sigs[43:42]`) is toggled.
2. Assign `18'd2`.
3. Assign `18'd0`.
4. Assign `1'b1`.
5. Assign `1'b0`.
6. Asserted whenever `pipe_rx0_eqcontrol_gt` (`pipe_tx_0_sigs[55:54]`) is toggled. The required logic is added in the `board.v` file of the reference example design.
7. Ignore these signals.

Avery Design Systems BFM PIPE Port Descriptions

Xilinx PCI Express Endpoint PIPE ports must be connected with the Avery Design Systems BFM PIPE ports. Connections can be made hierarchically in the simulation top file `board.v` of the Xilinx EP, or they can be connected directly by instantiating the Xilinx EP in the `avery_tb_rc_bfm` top wrapper. Table 6 maps corresponding signals between Avery Design Systems BFM and the Xilinx Endpoint core.

Table 6: **Avery Design Systems BFM Port Mapping**

Avery Port Name	Endpoint Core PIPE Signal Name	Bus PIPE Port Name
aTxDetectRx	pipe_tx_rcvr_det_gt	common_commands_out[0]
aRate[1:0]	pipe_tx_rate_gt	common_commands_out[2:1]
aTxDeemph[143:0]	pipe_tx_deemph_gt	common_commands_out[3]
aTxMargin[2:0]	pipe_tx_margin_gt	common_commands_out[6:4]
aTxSwing	pipe_tx_swing_gt	common_commands_out[7]
apipe_txd[31:0]	pipe_tx0_data_gt	pipe_tx_0_sigs[31:0]
apipe_txk[1:0]	pipe_tx0_char_is_k_gt	pipe_tx_0_sigs[33:32]
aTxElecIdle[0]	pipe_tx0_elec_idle_gt	pipe_tx_0_sigs[34]
aTxDataValid[0]	pipe_tx0_data_valid_gt	pipe_tx_0_sigs[35]
aTxStartBlock[0]	pipe_tx0_start_block_gt	pipe_tx_0_sigs[36]
aTxSyncHeader[1:0]	pipe_tx0_syncheader_gt	pipe_tx_0_sigs[38:37]

Table 6: **Avery Design Systems BFM Port Mapping *(Cont'd)***

Avery Port Name	Endpoint Core PIPE Signal Name	Bus PIPE Port Name
aRxPolarity[0]	pipe_rx0_polarity_gt	pipe_tx_0_sigs[39]
aPowerDown[1:0]	pipe_tx0_powerdown_gt	pipe_tx_0_sigs[41:40]
aTxCompliance[0]	pipe_tx0_compliance_gt	pipe_tx_0_sigs[69]
apipe_rxd[31:0]	pipe_rx0_data_gt	pipe_rx_0_sigs[31:0]
apipe_rxk[1:0]	pipe_rx0_char_is_k_gt	pipe_rx_0_sigs[33:32]
aRxDataValid[0]	pipe_rx0_data_valid_gt	pipe_rx_0_sigs[34]
aRxElecIdle[0]	pipe_rx0_elec_idle_gt	pipe_rx_0_sigs[35]
aRxStartBlock[0]	pipe_rx0_start_block_gt	pipe_rx_0_sigs[36]
aRxSyncHeader[1:0]	pipe_rx0_syncheader_gt	pipe_rx_0_sigs[38:37]
aRxStatus[2:0]	pipe_rx0_status_gt	pipe_rx_0_sigs[41:39]
aRxValid[0]	pipe_rx0_valid_gt	pipe_rx_0_sigs[42]
aPhyStatus[0]	pipe_rx0_phy_status_gt	pipe_rx_0_sigs[43]
aRxPresetHint[23:0]1		
aFS[47:0]2		
aLF[47:0]2		
aRxEqEval[7:0]2		
aRxStandby[7:0]2		
aInvalidRequest[7:0]2		
aBlockAlignControl(1)		
aEncodeDecodeBypass[7:0](2)		
aTxUpconfig[7:0](2)		
aRxUpconfig[7:0](2)		
aLocalPresetIndex[31:0](2)		
aGetLocalPresetCoefficients[7:0](2)		
aWidth[1:0](3)		
aPCLK_Rate[2:0](4)		

Notes:

1. Assign `24'hFFFFFF`.
2. Assign 0s.
3. If `aRate` is `2'b10` (Gen3 speed), assign 2'b10 (32-bit datapath/quad pipe), otherwise assign `2'b01` (16-bit datapath/dual pipe).
4. If `aRate` is `2'b00` (Gen1 speed), assign `3'b001` (pipe_clk is 125 MHz and dual pipe data), otherwise assign `3'b010` (pipe_clk is 250 MHz and quad pipe data). The required logic is added in the `board.v` file of the reference example design.

Test Bench Integration

These steps are required to integrate the Avery Design Systems BFM with the Xilinx Integrated PCI Express Endpoint block:

1. Download the Avery Design System BFM kit and Xilinx reference example design of this application note. For details, see Libraries, and Reference Design Files. Extract the files using `tar xfz <tarfile>`. For the BFM kit, follow the Avery Design Systems BFM integration procedure for setting environment variables, paths and simulation instructions located in the README file under `avery_bfm_kit`.
2. In the Vivado integrated design environment, create a new design targeting a device that contains a Gen3 PCI Express block. Target a **VC709** board, and ensure that the target language is set to **Verilog** in the Vivado Project Settings.
3. In the Flow Navigator, select **IP Catalog**.
4. Select the **Virtex-7 FPGA Gen3 Integrated Block for PCI Express** to customize the PCI Express core.
5. Customize these settings:
   - Lane width = **X8**.
   - Link speed = **8.0 GT/s**.
   - Ensure that **Enable External PIPE Interface** is selected.

   Note: See the *Virtex-7 FPGA Gen3 Integrated Block for PCI Express* (PG023) [Ref 5] for detailed information about customizing the core for Gen3 mode of operation, and see the *Xilinx Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 3] for detailed information about using logic simulation tools in the Vivado Design Suite.

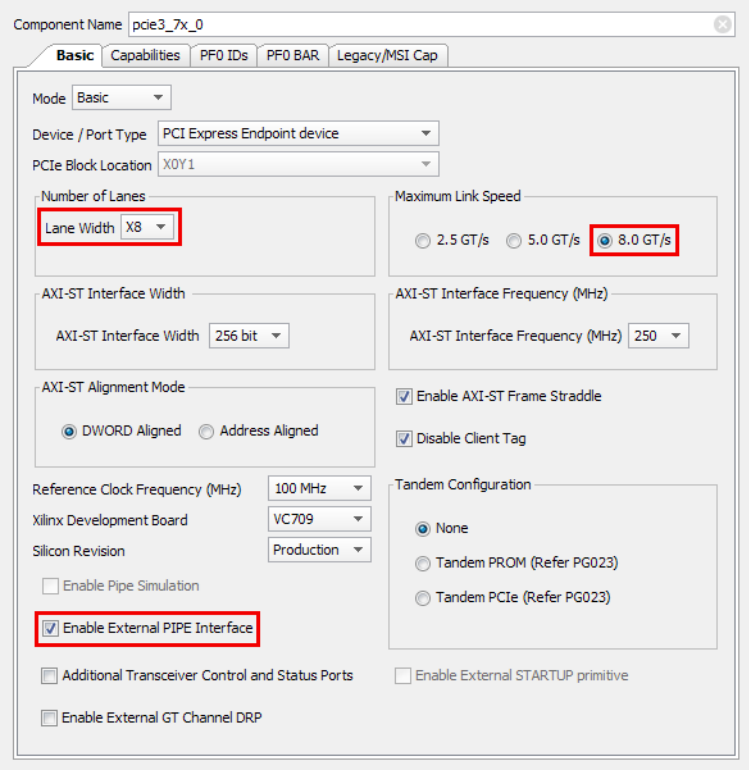

Figure 3: **Customization Options**

6. After the PCIe IP is generated, right-click the IP and select **Open IP Example Design**.
7. Select **QuestaSim** as Target simulator in Simulation Project Settings.
8. In the Flow Navigator, select **Run Simulation**.

9. Type the `run -all` command at the VSIM console to run the Xilinx RP and Xilinx EP in PIPE mode simulation.
10. Identify the `board.do` file from the simulation directory of the Xilinx example design (`../sim_1/behav/`), and add the Avery Design Systems compiled simulation libraries in the vsim command as instructed in the README file of Avery Design Systems BFM kit.
11. Add the `copy` command in the `board.do` file to copy the Avery compiled library to the simulation directory `/sim_1/behav/`. See the Avery README file for more information.
12. Open the `board.v` file and connect the PIPE signals to and from the Xilinx EP and Avery RC BFM, as described in Table 4, Table 5, and Table 6. Comment out the Xilinx RP and `phy_sig_gen` modules instantiation.
13. In the `board.v` file, connect the `apipe_reset` signal to the `sys_rst_n` signal. Generate the `phy_rdy` signal asserted after 10 µs and generate the `mmcm_lock` signal asserted after 10 ns.
14. Generate phase aligned (posedge) `pipe_clk`, `clk_250_mhz` (`user_clk`), `clk_500_mhz` (`core_clk`) clocks from the Avery Design Systems BFM clock `aclk250M` in the `sys_clk_gen_ds.v` file.
15. Add these three clocks as output ports and update the `sys_clk_gen_ds.v` instantiation in the `board.v` file.

Running the Simulation

To run the simulation:

1. Go to the Xilinx EP core generated simulation directories `sim_1/behav/`, and copy the `avery_bfm_kit` directory to the `sim_1/` directory.
2. Set the environment variable $AVERY_BFM_KIT to the `avery_bfm_kit` directory path. Based on your requirement, use either the 32-bit or 64-bit compiled libraries of `avery_bfm_kit` from Avery Design Systems.
3. Run `board.do` using the vsim command `vsim -do board.do` and check the Link up in Gen2 x8/Gen3 x8, LTSSM status.

The following log files created during simulation in `sim_1/behav/`provide valuable information for analyzing and debugging issues.

- **simulation.log**: Provides details about simulation summary, such as Link up, BIOS Enumeration, Memory Write/Read Test PASSED or FAILED.
- **transcript**: Provides details about compiled list of design files that are included in `board.do`.
- **rc1_symbol_tracker.txt**: Provides details about exchange of PLP, DLLP, TLPs in the form of symbols across TX and RX lanes of Avery Design Systems RC BFM, such as TS1, TS2, SKP, SDP, STP, ACK, NAK and the corresponding LTSSM changes.
- **root_complex1_new.track** and **root_complex1.track**: Provide details about DLLP, TLP packets at Avery Design Systems RC BFM, including Header details, Address, Data along with LTSSM changes.

Serial versus PIPE Modes

Table 7 provides guidelines about the simulation time differences between the Serial and PIPE modes of simulation in the Gen2 x8/Gen3 x8 configuration. The findings assume that the design contains Xilinx transceivers.

Note: Xilinx transceivers are found in serial mode simulation only. In PIPE mode simulation, they are bypassed.

Table 7: **Simulation Time Differences in Serial Mode and PIPE Simulation(1)**

Configuration	Simulation Mode	Simulation Time	Wall Clock Time
Gen2 x8	Serial mode simulation (Xilinx EP and Xilinx RP) Includes Link up (L0), configurations, single MEM Wr and Rd transactions	114 µs	36 minutes
	PIPE mode simulation (Xilinx EP and Avery RP) Up to Link up (L0), configurations, Multiple MEM Wr and Rd transactions	60 µs	3 minutes
Gen3 x8	Serial mode simulation (Xilinx EP and Xilinx RP) Includes Link up (L0), configurations, single MEM Wr and Rd transactions	128 µs	60 minutes
	PIPE mode simulation (Xilinx EP and Avery RP) Up to Link up (L0), configurations, Multiple MEM Wr and Rd transactions	100 µs	4 minutes

Notes:

1. The numerical values in this table are captured on a typical Xilinx server with the Mentor Graphics Questa® SIM simulator. Minor deviations can be expected based on the server configuration and simulator used.

Limitations

Xilinx has tested PIPE mode simulation with Avery Design Systems RC BFM kit but it should work with other third-party RC BFMs. This application note targets Gen2 x8 configuration, Gen3 x8 configuration, Verilog HDL, Questa SIM 10.2a simulator. This application note is supported by Xilinx Vivado Design Suite 2013.3 and later releases for Gen2 x8 configuration, 2013.4 and later releases for Gen3 x8 configuration.

For access to the Avery Design Systems BFM kit, contact Avery Design Systems [Ref 4].

Conclusion

PIPE mode simulation is very useful for reducing the simulation time during verification of complex PCI Express applications. From this approach, this application note highlighted the reduction in simulation time and the differences between serial mode and PIPE mode. This document also provided simple integration steps between Avery Design Systems RC BFM and Xilinx EP core. Instructions are also provided for incorporating these changes in user designs for speeding up the PCIE application verification cycle.

References

The documents referenced in this application note are listed in this section.

1. PHY Interface for the PCI Express (PIPE) Specification at www.intel.com
2. *7 Series FPGAs Integrated Block for PCI Express Product Guide* (PG054)
3. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
4. Avery Design Systems
5. *Virtex-7 FPGA Gen3 Integrated Block for PCI Express v3.0 Product Guide* (PG023)

Libraries, and Reference Design Files

Avery Design Systems Libraries

The Avery Design Systems BFM kit v1.0 used in this application note is available at http://demokit.avery-design.com/

Reference Design Files

The Gen2 x8 configuration reference design files for this application note are available at:

https://secure.xilinx.com/webreg/clickthrough.do?cid=351708, registration required.

The Gen3 x8 configuration reference design files for this application note are available at:

https://secure.xilinx.com/webreg/clickthrough.do?cid=355784, registration required.

Table 8: **Reference Design Details**

Parameter	Description
General	
Developer Name	Xilinx
Target Devices	7 series
Source code provided	Yes
Source code format	Verilog
Design uses code and IP from existing Xilinx application note, reference designs, Vivado tools or third party	Yes
Simulation	
Functional Simulation Performed	Yes
Timing simulation performed	No
Test bench used for functional and timing simulations	Yes
Test bench format	Verilog
Simulator software/version used	Questa SIM 10.2a
SPICE/IBIS simulations	-NA-
Implementation	
Synthesis software tools/version used	-NA-
Implementation software tools/versions used	-NA-
Static timing analysis performed	-NA-
Hardware Verification	
Hardware verified	-NA-
Hardware platform used for verification	-NA-

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
11/01/2013	1.0	Initial Xilinx release.
02/19/2014	2.0	Add Gen3 x8 configuration.

Notice of Disclaimer