HLS Compiler–Platform Designer Stitching and Testbench Generation Tutorial

1 Introduction

This tutorial demonstrates how to generate multiple components with the Intel HLS Compiler and connect them together in Platform Designer (formerly QSys) to function as a cohesive design. A high-pass filter and a low-pass filter are implemented in C++, and they are cascaded using Platform Designer. Cascading a high-pass filter and a low-pass filter is equivalent to implementing a bandpass filter. You can also write a single component that does bandpass filtering, but there are cases where it is useful to split the components, such as:

- You want explicit control of the datapath and the communication of different parts of the system.
- You want to integrate the components generated by the Intel HLS Compiler into a system with other existing IPs.

A `main()` function is not necessary because this tutorial does not run the design through any functional verification stages (either in emulation or co-simulation). However, this tutorial walks you through the steps to test the full system with a SystemVerilog testbench that was written separately.

2 Running the Tutorial

To run this tutorial:

- Run `make` (on Linux)
- Run `build` (on Windows)

The following things occur when you run this tutorial:

- The components are compiled using the Intel HLS Compiler
- The `ip-make-ipx` command is run to generate the components.ipx file which enables Platform Designer to easily identify the components generated by HLS.

- The `qsys-script` command is run on `top.tcl` to generate the Platform Designer system automatically.

  Typically, you would make these connections manually. For this particular case, the connections have been made and saved in the tcl file and running this command on the tcl file generates a `top.qsys` file. You can also view the connections from the Platform Designer. See Section 3 for more information.

- The `qsys-generate` command is run on `top.qsys` to generate the HDL files for simulation. You can also generate the HDL files from Platform Designer. See Section 4 for more information.

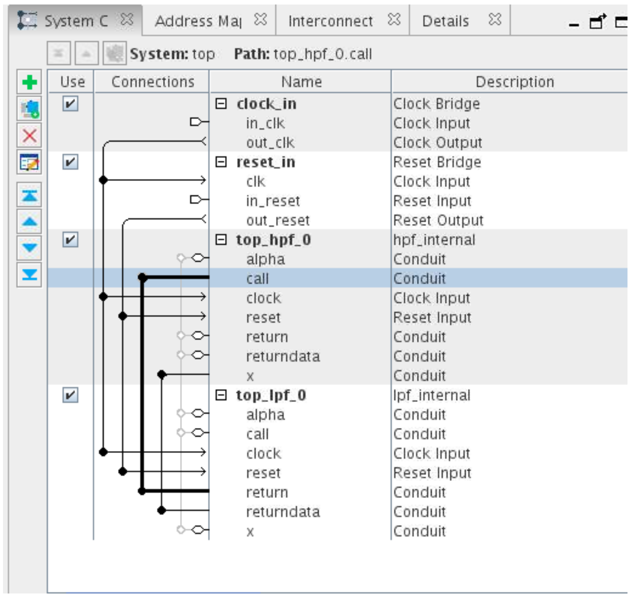
- The `vsim` command is to run the simulation using a tcl script, `test.tcl`, which compile and elaborate a pre-written testbench.

3 Editing Component Connections in Platform Designer

You can use Platform Designer to edit the connections between components manually. To invoke the application, type the following command `qsys-edit top.qsys`.

If you are using Platform Designer Pro version, you must specify the Intel Quartus project path. If you have a Intel Quartus project that you want to associate with, enter the path to the Intel Quartus project file. Otherwise, manually enter `None` to the textbox.

After you open the system in the Platform Designer, you can observe how the two components are chained together. These connections are shown in Figure 1.



System: top **Path:** top_hpf_0.call

Use	Connections	Name	Description
✓		⊟ **clock_in**	Clock Bridge
		in_clk	Clock Input
		out_clk	Clock Output
✓		⊟ **reset_in**	Reset Bridge
		clk	Clock Input
		in_reset	Reset Input
		out_reset	Reset Output
✓		⊟ **top_hpf_0**	hpf_internal
		alpha	Conduit
		call	Conduit
		clock	Clock Input
		reset	Reset Input
		return	Conduit
		returndata	Conduit
		x	Conduit
✓		⊟ **top_lpf_0**	lpf_internal
		alpha	Conduit
		call	Conduit
		clock	Clock Input
		reset	Reset Input
		return	Conduit
		returndata	Conduit
		x	Conduit

Figure 1: Stitched Design in Platform Designer System Contents Window

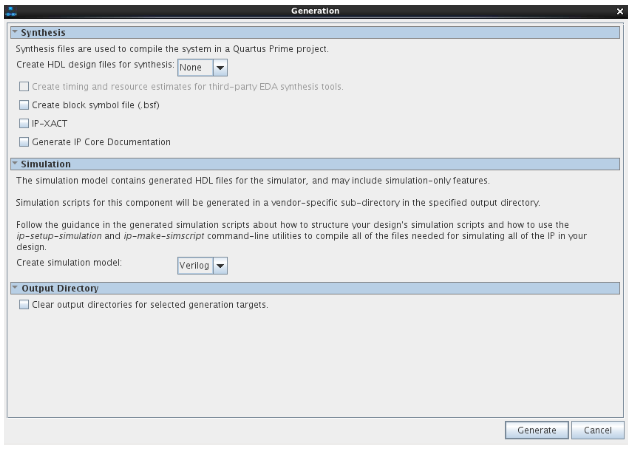
The two components share a single clock and a reset signal. The return interface of the low-pass filter component, called `top_lpf_0`, is connected to the call interface of the high-pass filter component, called `top_hpf_0`. Also, the `returndata` signal of the low-pass filter component is fed to the `x` signal of the high-pass filter component.

The HLS components that the Intel HLS Compiler generated are shown in the IP library. For the high-pass filter, the `alpha`, `return`, and `returndata` ports are exported to expose these conduits. Similarly, for the low-pass filter, the `alpha`, `call`, and `x` ports are exported.

4 Generating RTL for the Platform Designer System for Simulation

To generate the RTL for the stitched Platform Designer system:

1. Click **Generate** > **Generate HDL** from the menu bar. This opens the dialog box shown in Figure 2.



Generation

Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: None

☐ Create timing and resource estimates for third-party EDA synthesis tools.

☐ Create block symbol file (.bsf)

☐ IP-XACT

☐ Generate IP Core Documentation

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: Verilog

Output Directory

☐ Clear output directories for selected generation targets.

Figure 2: Generate HDL Window

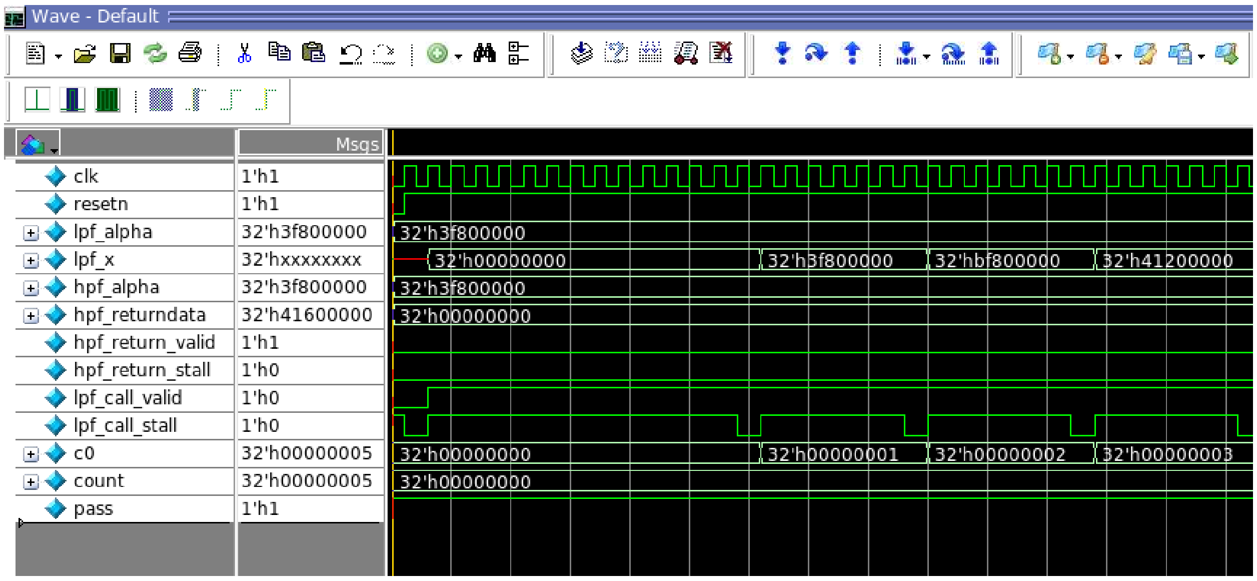
2. Navigate to the **Simulation** section and choose **Verilog** as the simulation model.

3. Click **Generate** to generate the RTL for the system.

5 Running the Simulation with Testbench

The testbench file, `tb.sv`, has been created for test the full system. The script, `test.tcl`, drives the simulation using this testbench. With access to ModelSim, run `vsim -batch -do test.tcl`. This command invokes ModelSim, compiles and runs the simulation, prints the simulation result to the stdout, and saves the resulting waveform to `top/sim/mentor/vsim.wlf`.

If the simulation is successful, the stdout prints a **PASSED** message; otherwise, the stdout prints a **FAILED** message along with a detailed message containing the expected output and the actual output.


Wave - Default

	Msgs
clk	1'h1
resetn	1'h1
lpf_alpha	32'h3f800000
lpf_x	32'hxxxxxxxx
hpf_alpha	32'h3f800000
hpf_returndata	32'h41600000
hpf_return_valid	1'h1
hpf_return_stall	1'h0
lpf_call_valid	1'h0
lpf_call_stall	1'h0
c0	32'h00000005
count	32'h00000005
pass	1'h1

Figure 3: Testbench Simulation Waveform Results

Additionally, you can also see the resulting waveform by typing `vsimtop/sim/mentor/vsim.wlf`.

A capture of the partial waveforms is shown in Figure 3.