AXI4-Stream Interfaces Demo Script

Introduction

This demonstration script provides high-level instructions on using the INTERFACE directive.

Preparation:

- Required files: Necessary files are located at *C:\training\hls\demos\axi_streaming*
- Required hardware: None
- Supporting materials: None

AXI4-Stream Interfaces

Action with Description	Point of Emphasis and Key Takeaway
• Launch the Vivado® HLS tool. • Open the provided **axi_interfaces_prj** Vivado HLS Project located at: *C:\training\hls\demos\axi_streaming*	You can open existing Vivado HLS tool projects from the Vivado HLS tool Welcome page.
• Access and review the source files (*axi_interface.c, axi_interface.h*, and *axi_interface_test.c*) from the Explorer pane.	This design has an input array and an output array. The comments in the C source files explain how the data in the input array is ordered as channels and how the channels are accumulated. To understand the design, you can also review the testbench and the input and output data in *result.golden.dat* file.

Action with Description	Point of Emphasis and Key Takeaway

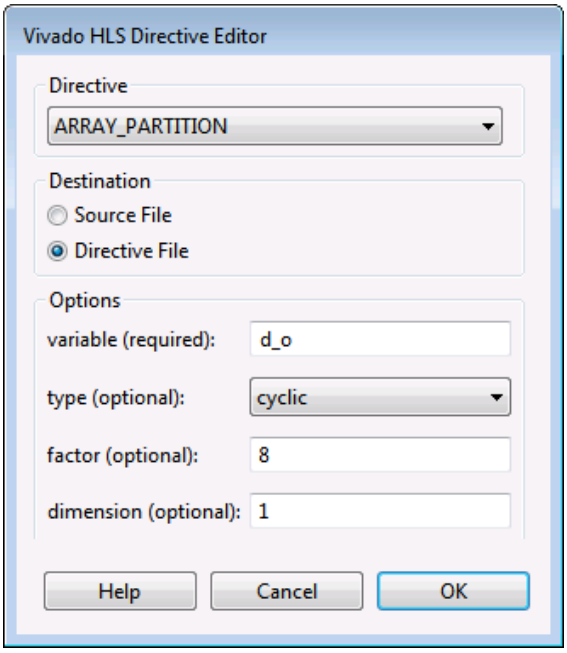
axi_interfaces.c

```
46 #include "axi_interfaces.h"
47
48 // The data comes in organized in a single array.
49 // - The first sample for the first channel (CHAN)
50 // - Then the first sample for the 2nd channel etc.
51 // The channels are accumulated independently
52 // E.g. For 8 channels:
53 // Array Order :  0  1  2  3  4  5  6  7  8     9     10    etc. 16       etc...
54 // Sample Order:  A0 B0 C0 D0 E0 F0 G0 H0 A1    B1    C2    etc. A2       etc...
55 // Output Order:  A0 B0 C0 D0 E0 F0 G0 H0 A0+A1 B0+B1 C0+C2 etc. A0+A1+A2 etc...
56
57 void axi_interfaces (dout_t d_o[N], din_t d_i[N]) {
58     int i, rem;
59
60     // Store accumulated data
61     static dacc_t acc[CHANNELS];
62
63     // Accumulate each channel
64     For_Loop: for (i=0;i<N;i++) {
65         rem=i%CHANNELS;
66         acc[rem] = acc[rem] + d_i[i];
67         d_o[i] = acc[rem];
68     }
69 }
70
```

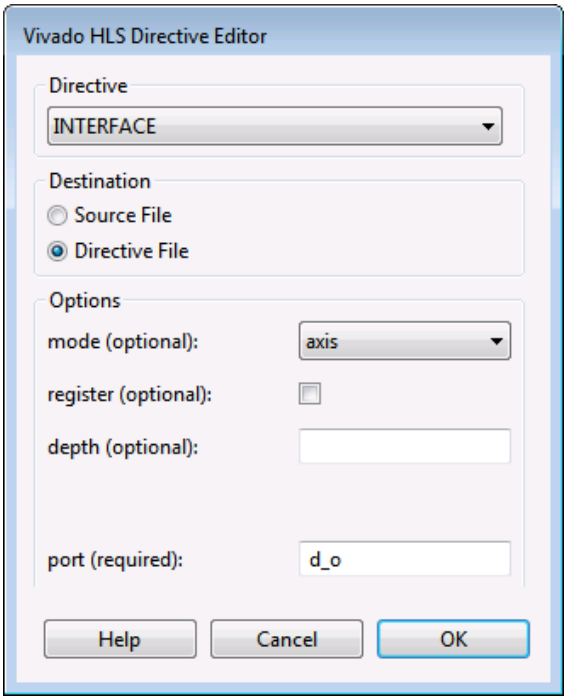
- Apply the directive below by the using the Vivado HLS Directive Editor:

```
set_directive_array_partition -type cyclic -factor 8 -dim 1 "axi_interfaces" d_i
set_directive_array_partition -type cyclic -factor 8 -dim 1 "axi_interfaces" d_o
```

Since there are eight channels, partition both the input and output arrays (d_i & d_o) cyclically with a factor of 8.

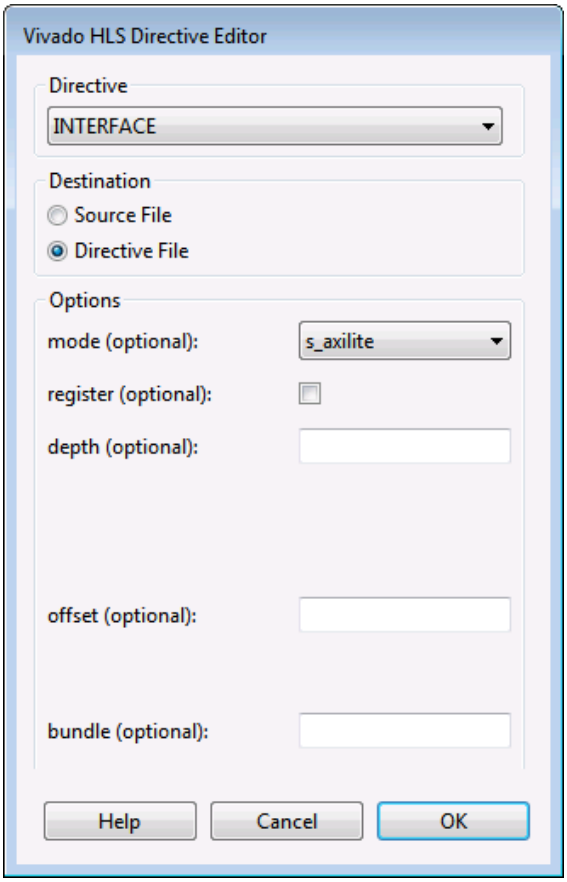


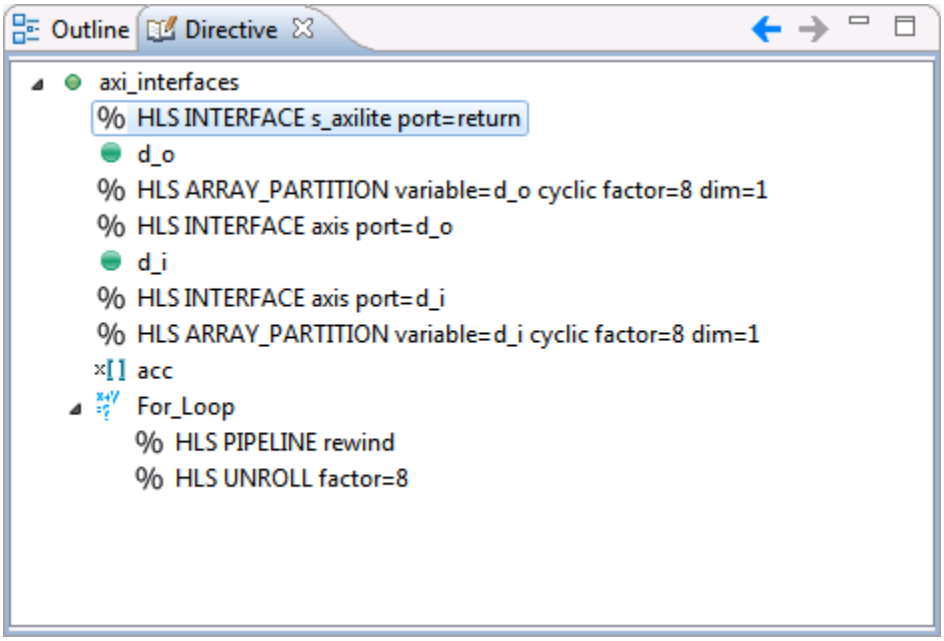
Action with Description	Point of Emphasis and Key Takeaway
• Partially unroll and pipeline the for loop: **`set_directive_unroll -factor 8 "axi_interfaces/For_Loop"`**	The Vivado HLS tool does not unroll the loops by default. Since there are eight channels in the design, partially unroll with a factor of 8. This is equivalent to rewriting the C code to execute eight copies of the loop body in each iteration of the loop (where the new loop only executes for four iterations in total, not 32).

• Specify an axis interface on the d_o port:
  • In the Directive tab, select **d_o** again and right-click to open the Directives Editor dialog box.
  • Select **INTERFACE** from the Directive drop-down list.
  • Select **axis** from the mode drop-down list.
  • Click **OK**.
• Similarly, specify an axis interface on the d_i port.
	AXI-4 Streaming interfaces are not native to the Vivado HLS tool.
	 

• Pipeline **FOR_Loop** with loop rewinding enabled. **`set_directive_pipeline -rewind "axi_interfaces/For_Loop"`**	When the top-level of the design is a loop, you can use the pipeline rewind option. This informs the Vivado HLS tool that when implemented in RTL, this loop runs continuously (with no end of function and function restart cycles).

Action with Description	Point of Emphasis and Key Takeaway

• Group the block-level I/O protocol ports into a single AXI4-Lite interface:
  • In the Directive tab, select the top-level **axi_interfaces** function and right-click to open the Directives Editor dialog box.
  • Select **INTERFACE** from the Directive drop-down list.
  • Select **s_axilite** from the mode drop-down list.
  • Click **OK**.
	This allows these block-level control signals to be controlled and accessed from a CPU.
	 

Action with Description	Point of Emphasis and Key Takeaway
The Directive pane should look like the figure below. 	
• Run C synthesis. • Review the **Interface Summary** in the Synthesis report.	Note the AXI Lite Slave interface and AXI STREAM interface in the Synthesis report.

Action with Description	Point of Emphasis and Key Takeaway
• Run C/RTL Cosimulation with the *Dump Trace all* option selected.	Note that the C/RTL Cosimulation passed. With Dump trace selected, the tool will generate a simulation trace file that is saved to the *<solution>/sim/<RTL>* folder. This file is useful for verifying the waveform. To open this in the Vivado Design Suite: • Launch the Vivado Design Suite and open any example project (or) open the hardware manager. • Enter the below commands in the Tcl Console. **`load_feature simulator`** **`open_wave_database <name>.wdb`** **`open_wave_config <name>.wcfg`** **Note:** These options can also be selected via the GUI.
• Export RTL for **IP Catalog** format (do not select the Evaluate option to save time).	The *\solution1\impl\ip* folder will contain the exported IP.
• In the Explorer pane, select the **\solution1** > **impl** > **ip** > **drivers** > **axi_interfaces_v1_0** > **src** folder. • Double-click the **xaxi_interfaces_hw.h** file.	When you add an AXI4-Lite interface to the design, the IP packaging process also creates software driver files to enable an external block, typically a CPU, to control this block (you can start it, stop it, set port values, and review the interrupt status).

Action with Description	Point of Emphasis and Key Takeaway
• In the Explorer pane, select the **\solution1** > **impl** > **ip** > **example** folder. • Double-click the **ipi_example.tcl** file to open it.	You can use ipi_example to create an example design that contains the Vivado HLS tool exported IP. **Usage:** `vivado -notrace -source ipi_example.tcl -tclargs <part> <zipfile>`

Summary

In this demo, you learned how to specify AXI Streaming and AXI Lite interfaces for I/O ports. In addition to showing how to add the AXI4 interfaces, this exercise also demonstrated how to create an optimal design by using interface and logic directives together.

References:

- Supporting materials
  - *Vivado Design Suite Tutorial: High-Level Synthesis* (UG871)
  - *Vivado Design Suite User Guide: High-Level Synthesis* (UG902)