

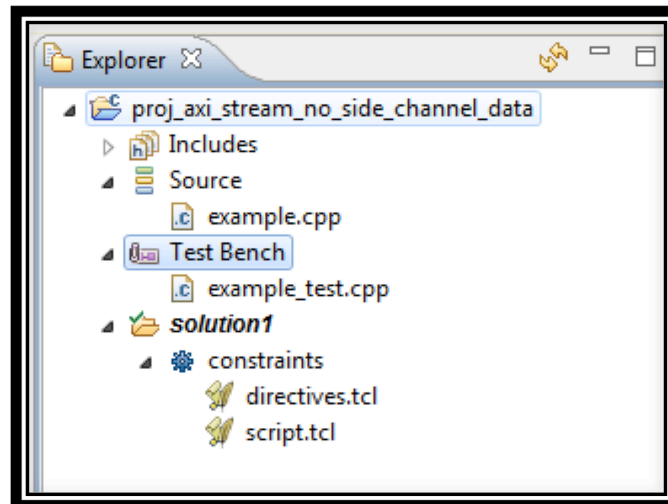
A Simple AXI-Stream Example Using HLS

This example will take you through the process of building a complete system that includes an AXI-Stream interface compatible IP using the tools (Vivado HLS, Vivado Design Suite and Xilinx SDK).

Open the Vivado HLS and make sure that you have Welcome screen visible by default. If it is not visible by default, click on HELP menu and select Welcome to show the screen as shown below.




Click on “Open Example Project” and select “axi_stream_no_side_channel_data” under “Design Examples”. Click next and finish to create the sample project.



The project created already includes the example code (example.cpp) and the corresponding testbench (example_test.cpp).

```
void example(int A[50], int B[50]) {  
    #pragma HLS INTERFACE axis port=A  
    #pragma HLS INTERFACE axis port=B  
  
    int i;  
  
    for(i = 0; i < 50; i++){  
        B[i] = A[i] + 5;  
    }  
}
```

Function “example” has two input arguments A[] and B[]. We are using #pragma directives to specify it to C synthesizer that we would like to have two AXI-Stream interfaces for these two inputs.

Run C simulation  to make sure that the software and hardware function have comparable results.

```
1  Compiling ../../../../example_test.cpp in debug mode  
2  Compiling ../../../../example.cpp in debug mode  
3  Generating csim.exe  
4 HLS AXI-Stream no side-channel data example  
5 Success HW and SW results match  
6 @I [SIM-1] CSim done with 0 errors.  
7
```

Run C synthesizer to view the performance estimates.

1. Timing (ns):

Timing (ns)			
Summary			
Clock	Target	Estimated	Uncertainty
default	13.33	2.00	1.67

2. Latency (clock cycles):

Latency (clock cycles)				
Summary				
Latency		Interval		
min	max	min	max	Type
51	51	52	52	none

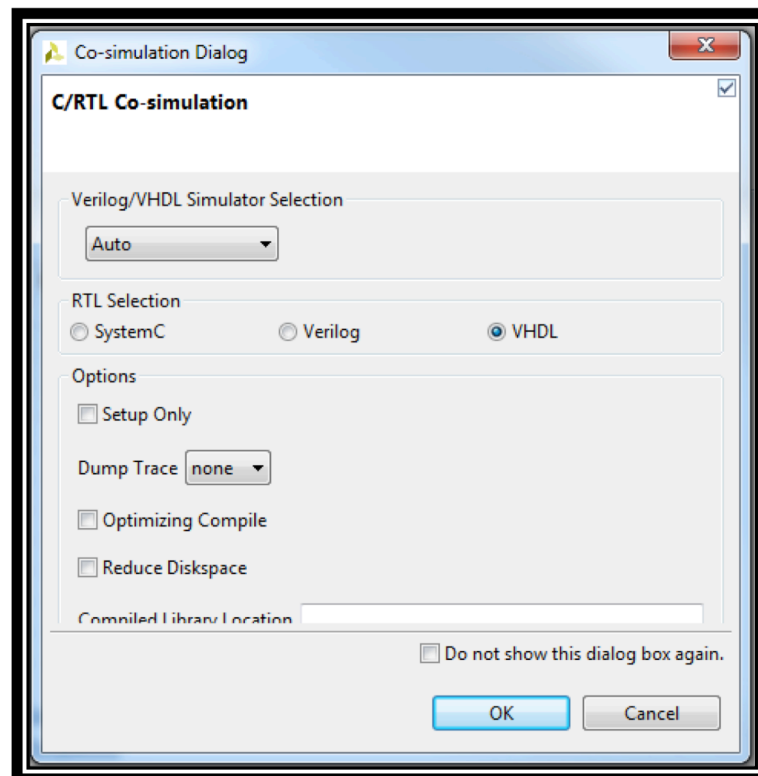
3. Utilization Estimates:

Utilization Estimates				
Summary				
Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	46
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	8
Register	-	-	9	-
Total	0	0	9	54
Available	280	220	106400	53200
Utilization (%)	0	0	~0	~0

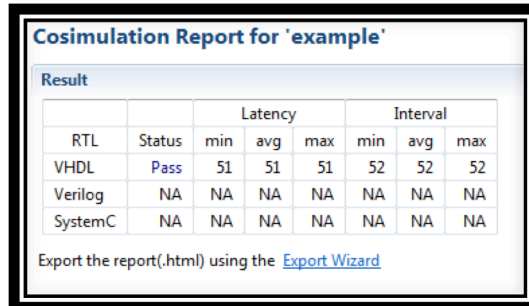
4. Interface:

Interface					
Summary					
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	example	return value
ap_rst_n	in	1	ap_ctrl_hs	example	return value
ap_start	in	1	ap_ctrl_hs	example	return value
ap_done	out	1	ap_ctrl_hs	example	return value
ap_idle	out	1	ap_ctrl_hs	example	return value
ap_ready	out	1	ap_ctrl_hs	example	return value
A_TDATA	in	32	axis	A	pointer
A_TVALID	in	1	axis	A	pointer
A_TREADY	out	1	axis	A	pointer
B_TDATA	out	32	axis	B	pointer
B_TVALID	out	1	axis	B	pointer
B_TREADY	in	1	axis	B	pointer

Click on “Run C/ RTL Cosimulation” ☒ . Make sure to select VHDL as your RTL selection.



Make sure that the RTL VHDL status is PASS in your Cosimulation report.




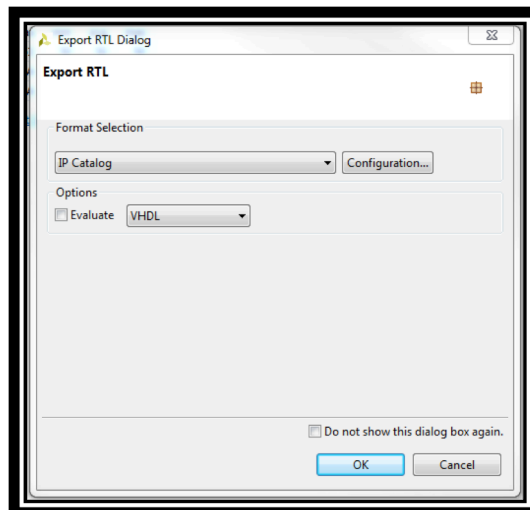
Cosimulation Report for 'example'

Result

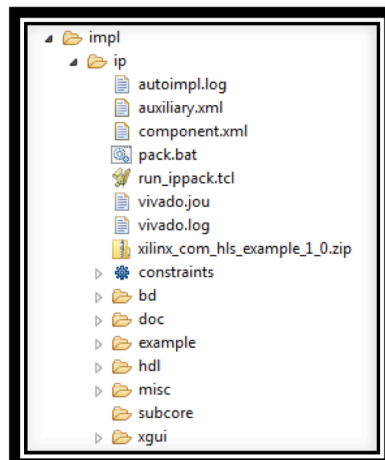
RTL	Status	Latency			Interval		
		min	avg	max	min	avg	max
VHDL	Pass	51	51	51	52	52	52
Verilog	NA	NA	NA	NA	NA	NA	NA
SystemC	NA	NA	NA	NA	NA	NA	NA

Export the report(.html) using the [Export Wizard](#)

Click on “Export RTL” icon  and make sure to choose VHDL from the drop down menu.



It will take some time for the tool to generate the implementation data including the IP directory. Make sure that you can see the IP and all related files.



Create a new application project in SDK and then include the file “example_c_code.c” into your project. Program FPGA, run the code, and based on the equation ($B[i] = A[i] + 5$), see the output on the console as shown below:

Output:

--- Entering main() ---

=====

TxBuffer[0] = 0
TxBuffer[1] = 1
TxBuffer[2] = 2
TxBuffer[3] = 3
TxBuffer[4] = 4
TxBuffer[5] = 5
TxBuffer[6] = 6
TxBuffer[7] = 7
TxBuffer[8] = 8
TxBuffer[9] = 9
TxBuffer[10] = 10
TxBuffer[11] = 11
TxBuffer[12] = 12
TxBuffer[13] = 13
TxBuffer[14] = 14
TxBuffer[15] = 15
TxBuffer[16] = 16
TxBuffer[17] = 17
TxBuffer[18] = 18
TxBuffer[19] = 19
TxBuffer[20] = 20
TxBuffer[21] = 21
TxBuffer[22] = 22
TxBuffer[23] = 23
TxBuffer[24] = 24
TxBuffer[25] = 25
TxBuffer[26] = 26
TxBuffer[27] = 27
TxBuffer[28] = 28
TxBuffer[29] = 29
TxBuffer[30] = 30
TxBuffer[31] = 31
TxBuffer[32] = 32
TxBuffer[33] = 33
TxBuffer[34] = 34
TxBuffer[35] = 35
TxBuffer[36] = 36
TxBuffer[37] = 37
TxBuffer[38] = 38
TxBuffer[39] = 39
TxBuffer[40] = 40
TxBuffer[41] = 41
TxBuffer[42] = 42

TxBuffer[43] = 43
TxBuffer[44] = 44
TxBuffer[45] = 45
TxBuffer[46] = 46
TxBuffer[47] = 47
TxBuffer[48] = 48
TxBuffer[49] = 49

=====

RxBuffer[0] = 5
RxBuffer[1] = 6
RxBuffer[2] = 7
RxBuffer[3] = 8
RxBuffer[4] = 9
RxBuffer[5] = 10
RxBuffer[6] = 11
RxBuffer[7] = 12
RxBuffer[8] = 13
RxBuffer[9] = 14
RxBuffer[10] = 15
RxBuffer[11] = 16
RxBuffer[12] = 17
RxBuffer[13] = 18
RxBuffer[14] = 19
RxBuffer[15] = 20
RxBuffer[16] = 21
RxBuffer[17] = 22
RxBuffer[18] = 23
RxBuffer[19] = 24
RxBuffer[20] = 25
RxBuffer[21] = 26
RxBuffer[22] = 27
RxBuffer[23] = 28
RxBuffer[24] = 29
RxBuffer[25] = 30
RxBuffer[26] = 31
RxBuffer[27] = 32
RxBuffer[28] = 33
RxBuffer[29] = 34
RxBuffer[30] = 35
RxBuffer[31] = 36
RxBuffer[32] = 37
RxBuffer[33] = 38
RxBuffer[34] = 39
RxBuffer[35] = 40
RxBuffer[36] = 41
RxBuffer[37] = 42
RxBuffer[38] = 43
RxBuffer[39] = 44
RxBuffer[40] = 45
RxBuffer[41] = 46
RxBuffer[42] = 47
RxBuffer[43] = 48


```
RxBuffer[44] = 49  
RxBuffer[45] = 50  
RxBuffer[46] = 51  
RxBuffer[47] = 52  
RxBuffer[48] = 53  
RxBuffer[49] = 54
```

```
=====  
AXI DMA interrupt example test passed
```

Modify the C code in Vivado HLS to create your own hardware accelerator that can perform Matrix multiplication, and follow the same design flow to implement and run your code.