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.

![Vivado Welcome Screen](image)

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).

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.

Run C synthesizer to view the performance estimates.
1. **Timing (ns):**

   ![Timing (ns) summary](image1)

<table>
<thead>
<tr>
<th>Clock</th>
<th>Target</th>
<th>Estimated</th>
<th>Uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>13.33</td>
<td>2.00</td>
<td>1.67</td>
</tr>
</tbody>
</table>

2. **Latency (clock cycles):**

   ![Latency (clock cycles) summary](image2)

<table>
<thead>
<tr>
<th>Latency</th>
<th>Interval</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>max</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>51</td>
<td>none</td>
</tr>
</tbody>
</table>

3. **Utilization Estimates:**

   ![Utilization Estimates summary](image3)

   ![Utilization Estimates details](image4)

<table>
<thead>
<tr>
<th>Name</th>
<th>BRAM_18K</th>
<th>DSP48E</th>
<th>FF</th>
<th>LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>46</td>
</tr>
<tr>
<td>FIFO</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiplexer</td>
<td>-</td>
<td>-</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Register</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>54</td>
</tr>
<tr>
<td>Available</td>
<td>280</td>
<td>220</td>
<td>106400</td>
<td>53200</td>
</tr>
<tr>
<td>Utilization (%)</td>
<td>0</td>
<td>0</td>
<td>~0</td>
<td>~0</td>
</tr>
</tbody>
</table>
4. Interface:

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.

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 Vivado project and click on Project settings in the Flow Navigator.

Click on the IP settings and include your Example IP into the current project.

Create the block design and include the “Example” IP along with all other component IPs required to build your design.

**Note:** Make sure to keep your “ap_start” signal HIGH for at least one clock cycle for the hardware accelerator to start working.

Synthesize and implement the design and generate the bitstream. Export the hardware, include the bitstream, and then launch the Xilinx SDK.
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[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
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.