EESM 6000C: LAB3 FIR DESIGN

Student Name: LIU Zhaoyang

E-mail: cliuel@connect.ust.hk

Student ID:21111010

1. INTRODUCTION

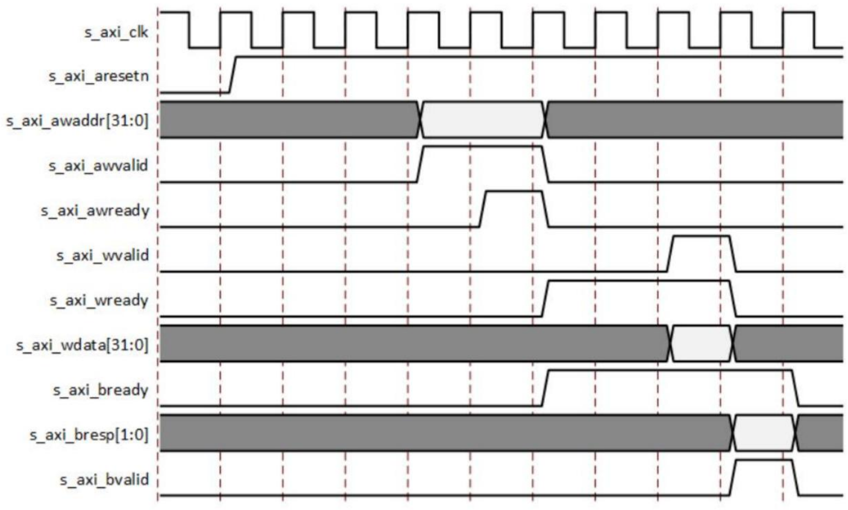
The project focuses on designing and implementing a FIR filter engine.The filter is based on the equation:

$$y[t] = \Sigma\ (h[i] * x[t - i])$$

with 11 taps (coefficients) and a 32-bit data width.
The design adopts the AXI Interfaces series protocols. Among these, the AXI-Lite protocol is used for transmitting tap coefficients, and AXI-Stream is used for transmitting data. It handles the streaming of input (X[n]) and output (Y[n]) data with flow control through TVALID/TREADY handshaking.

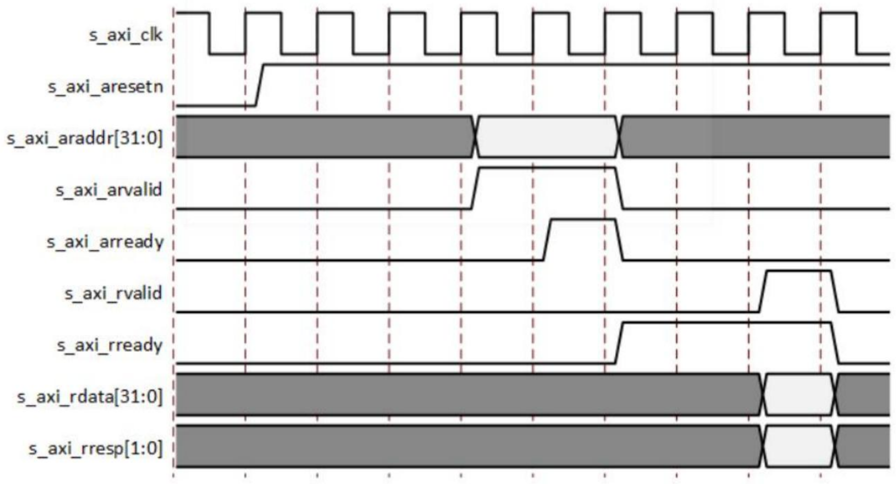
2. PROTOCOL

2.1 AXI-LITE



AXI_Lite Write Process

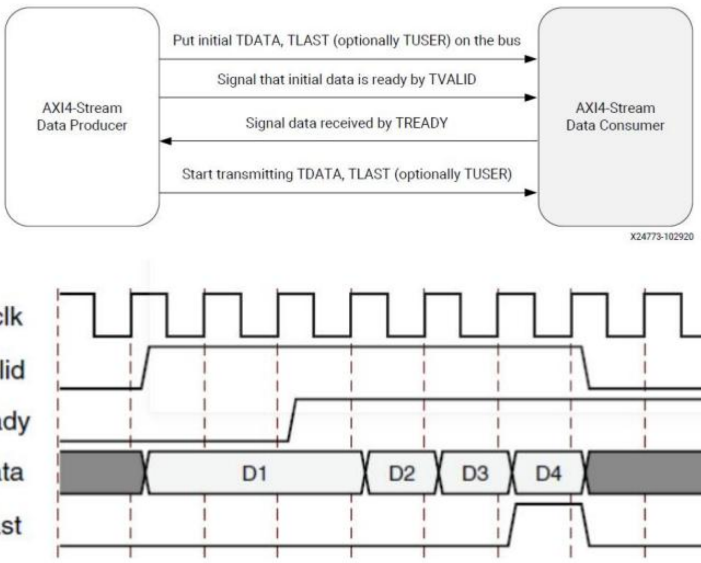
In this design, the AXI-Lite protocol is primarily used for transferring tap coefficients. During the write process, the external master writes the address into awaddr on the write address channel and writes the data into wdata on the write data channel. The external master asserts awvalid to indicate that the address on the respective channel is valid and asserts wvalid to indicate that the data on the respective channel is valid. When the Tap_RAM—that is, the slave—receives valid awvalid and wvalid signals, it raises the awready and wready signals, indicating that it is ready to receive the data write.



AXI_Lite Read Process

During the read process, the external master places the read address on the read address channel (araddr) and asserts arvalid, indicating that the master is ready to read data from the slave (i.e., Tap_RAM). Upon receiving the asserted arvalid signal, the slave raises the arready signal and then deasserts it in the next cycle, indicating that the slave has received the requested address. The slave then places the requested data on the read data channel (rdata) and asserts rvalid, indicating that the data on the channel is valid.

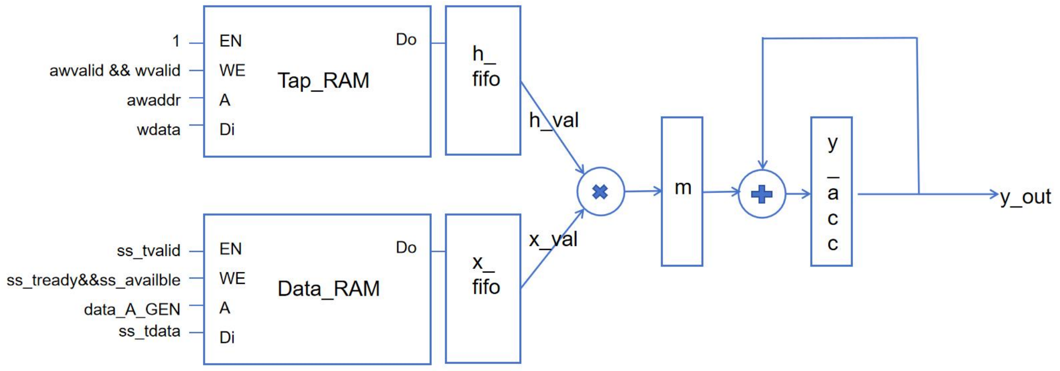
2.2 AXI-STREAM


AXI4-Stream Data Producer
Put initial TDATA, TLAST (optionally TUSER) on the bus
Signal that initial data is ready by TVALID
Signal data received by TREADY
Start transmitting TDATA, TLAST (optionally TUSER)
AXI4-Stream Data Consumer
X24773-102920
aclk
tvalid
tready
tdata
D1
D2
D3
D4
tlast

AXI_Stream Protocol

In this design, the AXI-Stream protocol is mainly used for data transmission. The external stimulus sends the ss_tvalid signal to Data_RAM to indicate that the data is valid, places the external data on ss_tdata, and asserts the ss_tlast signal to indicate the last byte, after which the data becomes invalid in the next cycle. Once the slave enters the computation running state, it sends the ss_tready signal to indicate that it is ready to receive data. For transmission to occur, both ss_tready and ss_tvalid must be asserted simultaneously; once tvalid is asserted, it must remain high until the handshake occurs. At the same time, when Data_RAM acts as the master, it transmits the computed y values externally. After a set of computations is completed, the sm_tvalid signal is asserted to indicate that the output y is valid, and the value on sm_tdata is the computed y value. When all the data have been transmitted, sm_tlast is asserted.

3. BLOCK DIAGRAM



Block Diagram

4. DESIGN

The design employs the AXI_Lite protocol for transferring tap coefficients and the AXI_Stream protocol for transmitting data. First, the registers are configured: address 0x00 is used as ap_ctrl to control the startup of the entire FIR program. A state machine manages the transitions and control of the ap_start, ap_done, and ap_idle states. The address range 0x10-0x14 is used to configure the number of data items, 0x14-0x18 to set the number of taps, and 0x80 to 0xFF to configure the tap coefficient values.

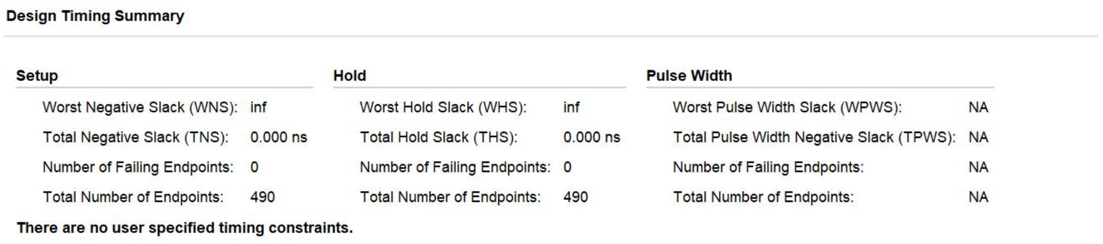
Next, a state machine controls the AXI_Stream, completing a series of transitions to ensure correct data input and output. Through these state transitions and handshaking, data from external sources is input into both Tap_RAM and Data_RAM. Additionally, because the data and tap coefficients do not arrive simultaneously, a FIFO mechanism is used to manage them: data_do is fed into x_fifo and tap_do into h_fifo. By controlling the read and write pointers, the coefficients and data are paired one-to-one, ultimately completing the computation and output of 11 sets of data.

5. RESOURCE USAGE

```
+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |  948 |     0 |     53200 |  1.78 |
|   LUT as Logic          |  948 |     0 |     53200 |  1.78 |
|   LUT as Memory         |    0 |     0 |     17400 |  0.00 |
| Slice Registers         |  885 |     0 |    106400 |  0.83 |
|   Register as Flip Flop |  882 |     0 |    106400 |  0.83 |
|   Register as Latch     |    3 |     0 |    106400 | <0.01 |
| F7 Muxes                |   64 |     0 |     26600 |  0.24 |
| F8 Muxes                |    0 |     0 |     13300 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

```
+----------------+------+-------+-----------+-------+
|   Site Type    | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| Block RAM Tile |    0 |     0 |       140 |  0.00 |
|   RAMB36/FIFO* |    0 |     0 |       140 |  0.00 |
|   RAMB18       |    0 |     0 |       280 |  0.00 |
+----------------+------+-------+-----------+-------+
```

6. TIMING REPORT

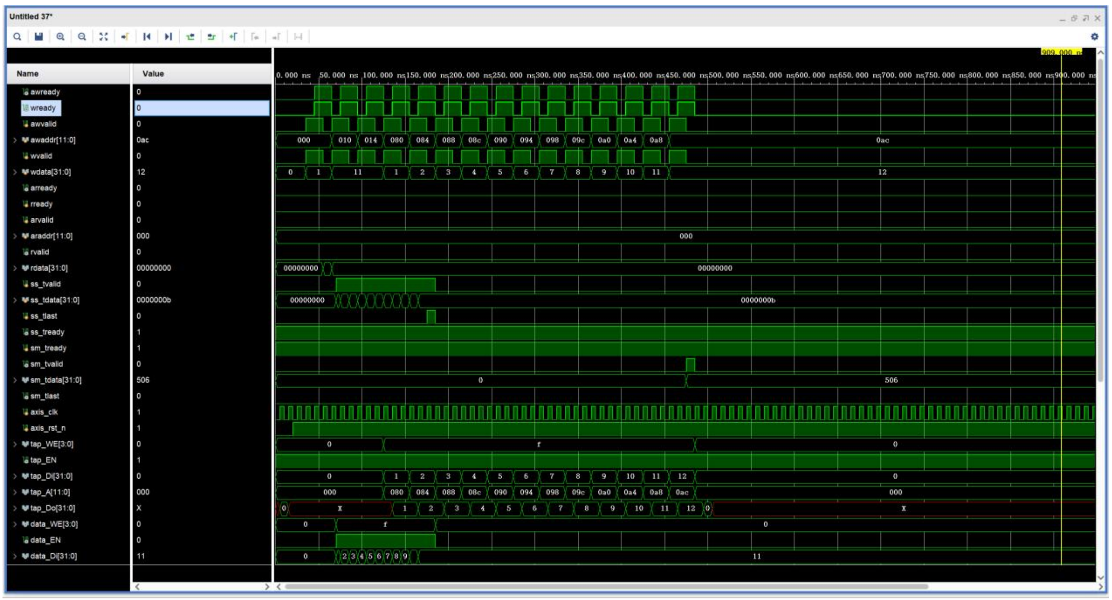


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	490	Total Number of Endpoints:	490	Total Number of Endpoints:	NA

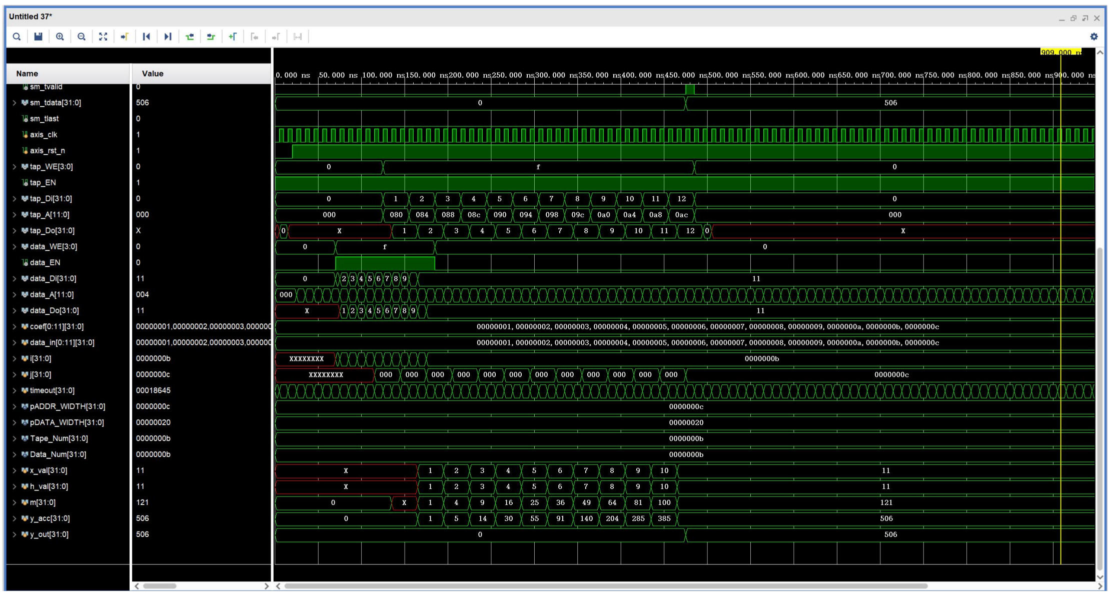
There are no user specified timing constraints.

7. SIMULATION WAVEFORM



Simulation Waveform1

In the testbench of this design, a set of data (11 data values and 11 tap coefficients) is used for testing. In the testbench, the value 1 is set at address 0x00 to indicate the FIR start, the value 11 is set at 0x10 to denote that the number of data is 11, and the value 11 is set at 0x10 to indicate that there are 11 tap coefficients. Then, the tap coefficients are sequentially written to addresses 0x80, 0x84, and so on. As shown in the waveform above, the corresponding address is passed to the Tap_RAM's tap_A signal, and the corresponding value is passed to Tap_Di of Tap_RAM, which in turn transfers the data to Tap_Do. The value of ss_tdata is set and transmitted to the Data_RAM's Di through the AXI-Lite protocol, and Data_Di then passes the data to Data_Do. Additionally, the address for Data_A is automatically generated by an internal generator and is passed sequentially.



Simulation Waveform2

From the above waveform, it can be seen that the output Tap_Do from Tap_RAM and the output Data_Do from Data_RAM do not arrive simultaneously. Therefore, the design employs FIFOs to temporarily store and then output the data. Tap_Do is passed into h_fifo, and Data_Do is passed into x_fifo. Synchronous output is achieved through the changes in the FIFO pointers. The register m collects the product of the corresponding x and h values, and y_acc accumulates the sum of m. After processing 11 sets of data, y_out outputs the final result of the 11 products added together.