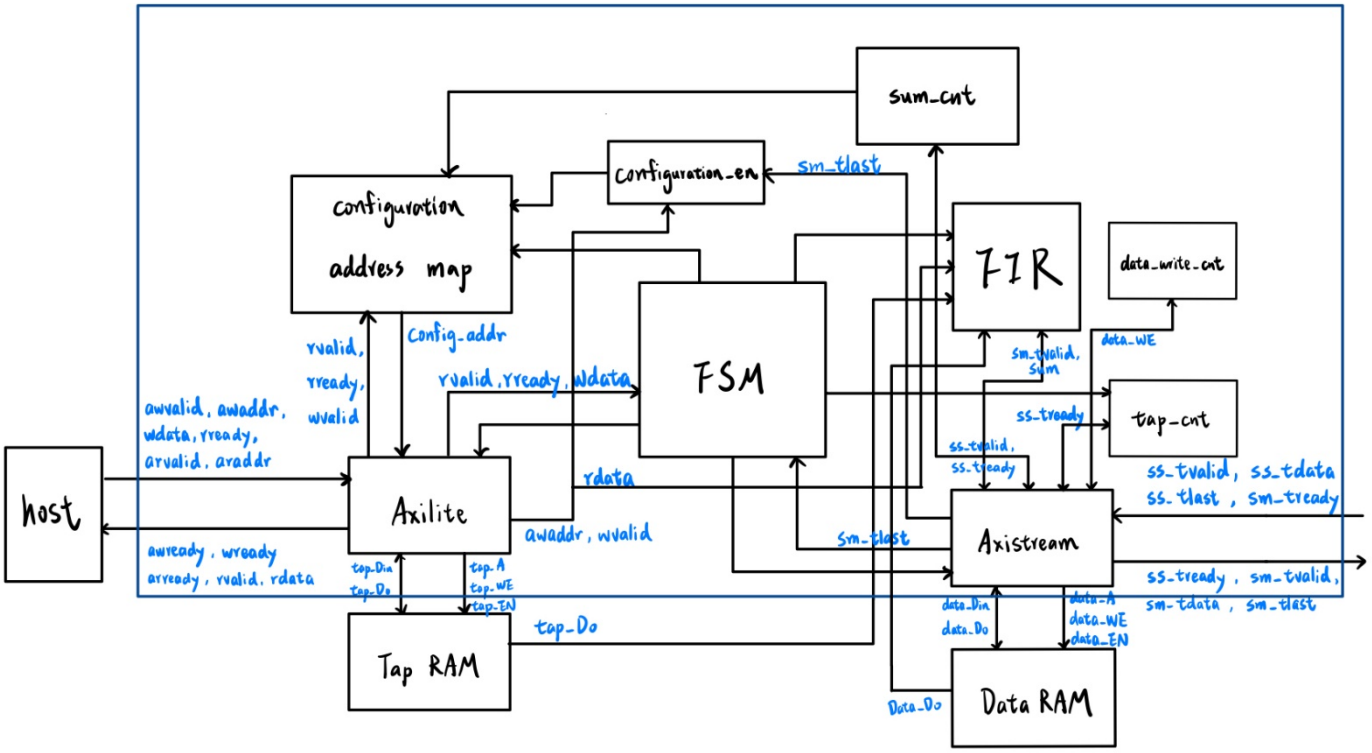
Lab 3 (Lab-FIR) Report

Student ID: 111061606 Name: 李柏辰

Overview

In this lab, we have to design a FIR engine with one multiplier and one adder. And using specified interface to deal with data transfer handshake. Tap coefficient is transferd to tap RAM by axilite interface , input and output data are transferd to data RAM by axi-stream interface. In addition, we should assert some control signals through configuration address map, such as : ap_start, ap_done, etc. To confirm the engine operate fluently.

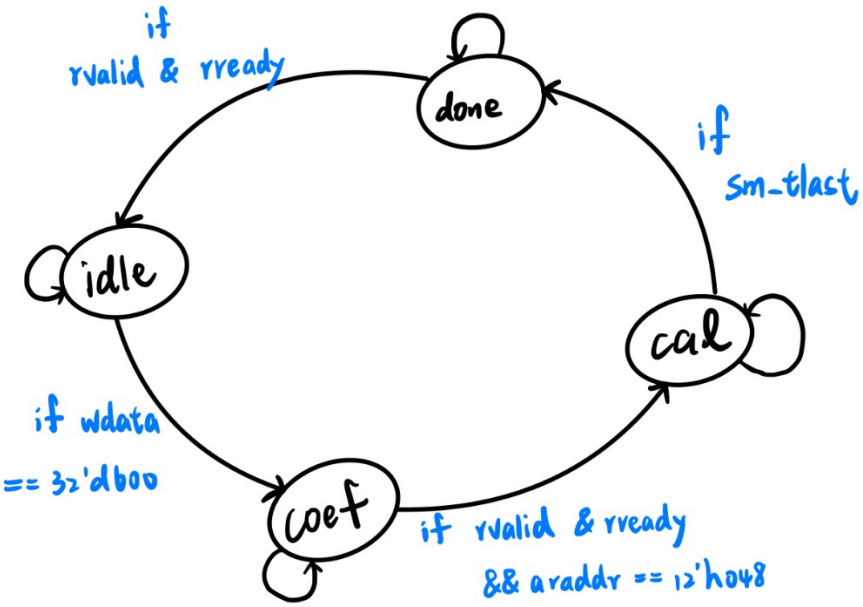
Block Diagram



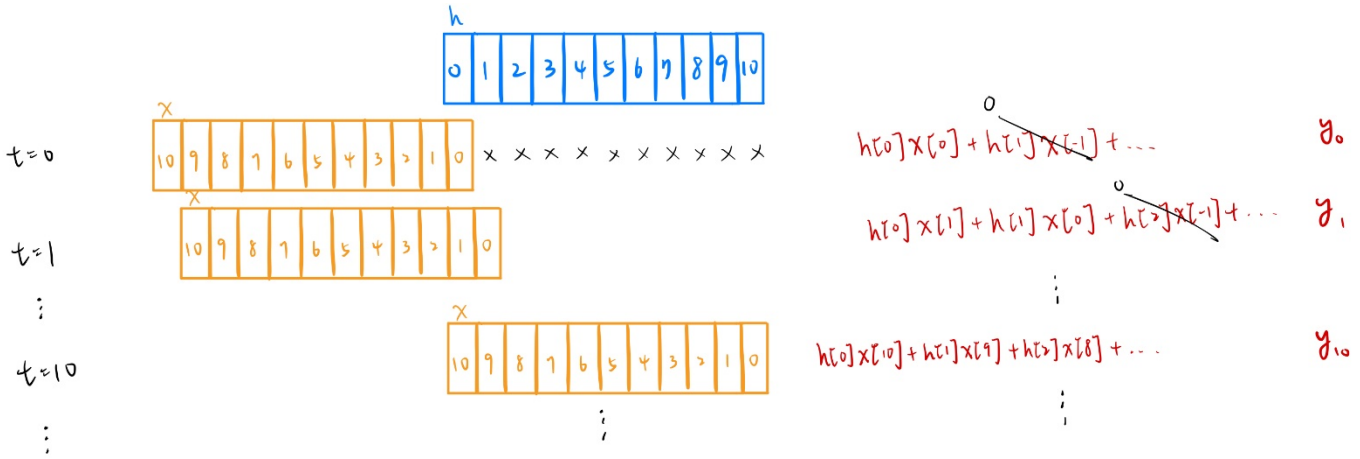
As shown above, we can divide the diagram into left-hand part and right-hand part. In the left-hand part, it mainly handles the data transfer handshake of tap coefficients and the control signal from the configuration address map. Between the host and tap RAM, the axilite interface would be the bridge to connect. In the other words, no matter what kind of data want to communicate with host, it should be go through axilite interface.

In the right-hand part, it's in charge of fir calculation and data input/output. Through the axi-stream interface, data RAM receives the first group of input data from the host. Then the data would be sent to fir block. After calculating with tap coefficients, the output would be sent back to the host through axi-stream interface. Subsequently, the host transfers the other input data to data RAM until the whole calculation finish.

Describe Operation



Between these two part, there is a finite state machine which manages all of the block's operation. I divide the process into four state. The first state is idle, the engine stays at this state until the wdata given by host become 32'd600, which is actually the length of input data. Then it would turn to the second state : COEF. At this state, host transfers tap coefficient to tap RAM and reads the data from tap RAM to check whether it is correct through axilite interface. After all the tap coefficient is settled down, the state converts to CAL state. Host starts to transfer input data into data RAM. As long as an output occurs, it would be sent back to host through axi-stream interface. Then host also transfers other input data for next output. Doing the process repeatly, when all the output finishes, host would assert a sm_tlast signal to announce. The state turns to the final one: DONE. If host wants to execute the engine again, it would turn back to IDLE state and do all the procedure above.



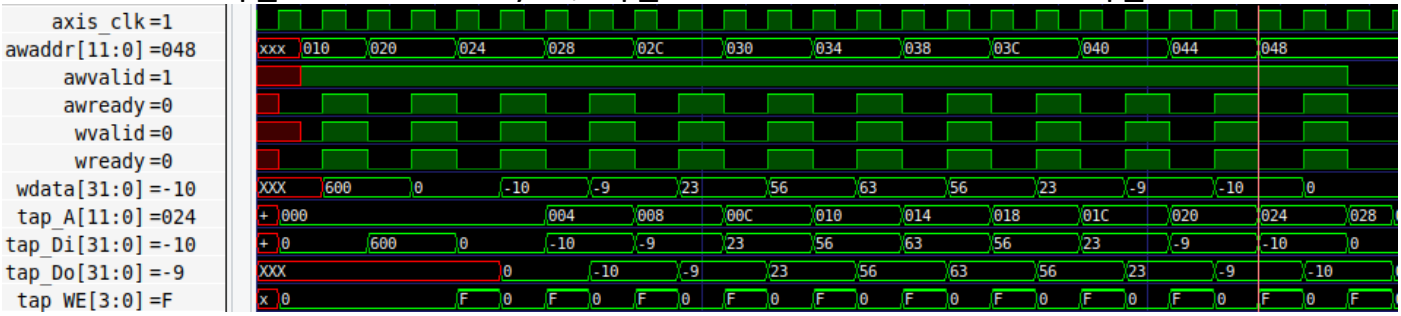
About FIR computation, we know that there are 11 tap cofficients. Thus, data RAM has to receive 11 input data from host in order to get one output. In addition, we can only use one multiplier and one adder in the calculation engine, so it takes 12 cycle(1 cycle to sum all the product) to get one output. As figure above, “h” is the tap RAM, “x” is the data RAM and “y” is the output data. As calculating the first output to 9th output, the input data in data RAM are actually not full. Therefore I initialize the storage space of data RAM before receiving any data. In this way, the first outpu0 y0 t would be h[0] multiply with x[0]. After receive output data, host transfers the next input data to data RAM. The previous input data would be shift right to x[1], the new input data is stored at x[0]. Then the second output y1 become h[0]*x[1]+h[1]*x[0] and so on.

To determine when to generate ap_done, I design a sum counter. As long as host transfers a new input data to data RAM, the counter number would plus one. In other words, the number represents which output is calculating. When the last output finish, the counter number becomes 601. At this time, ap_done is asserted.

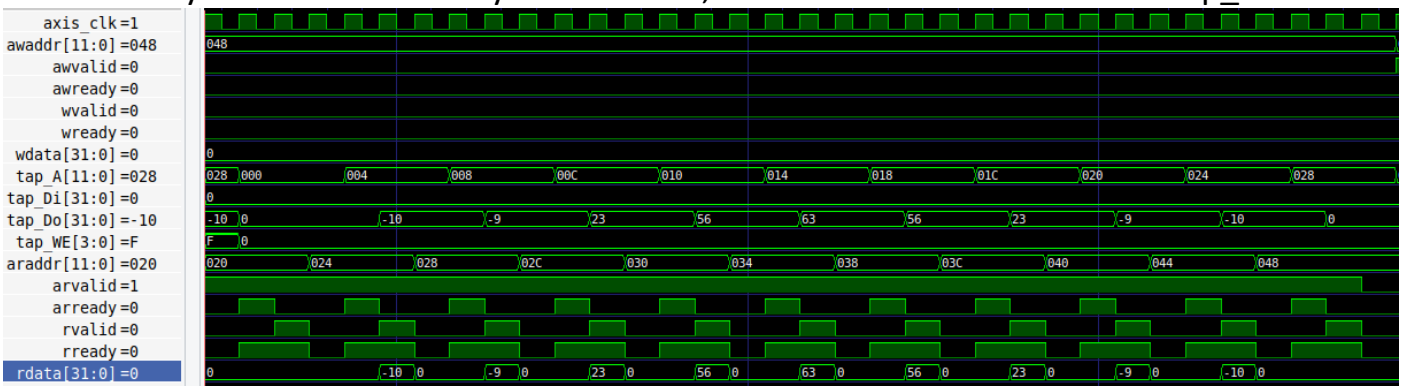
Simulation Waveform

- Coefficient

  According to awaddr, tap_A is awaddr minus 12’h020. When awready and wready are asserted, it would trigger tap_WE. As tap_WE stay high, bram would set wdata to tap_Di. In the next cycle, tap_Do is written as the data of tap_Di.

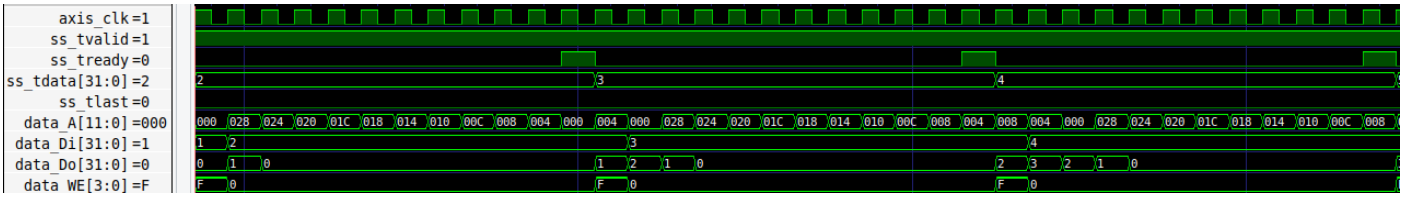


As reading check part, rvalid is asserted when arvalid and arready stay high. Only as rvalid and rready are asserted, rdata would take the value of tap_Do.

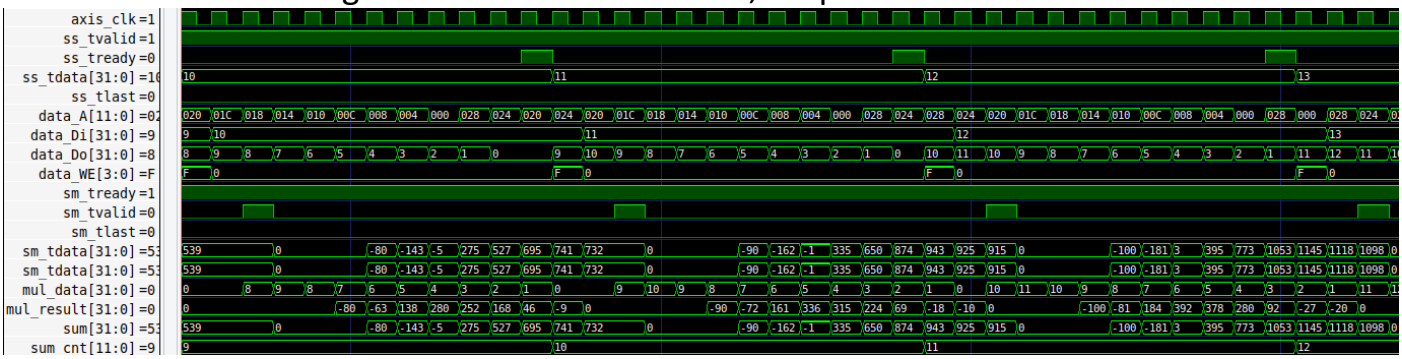


- Data

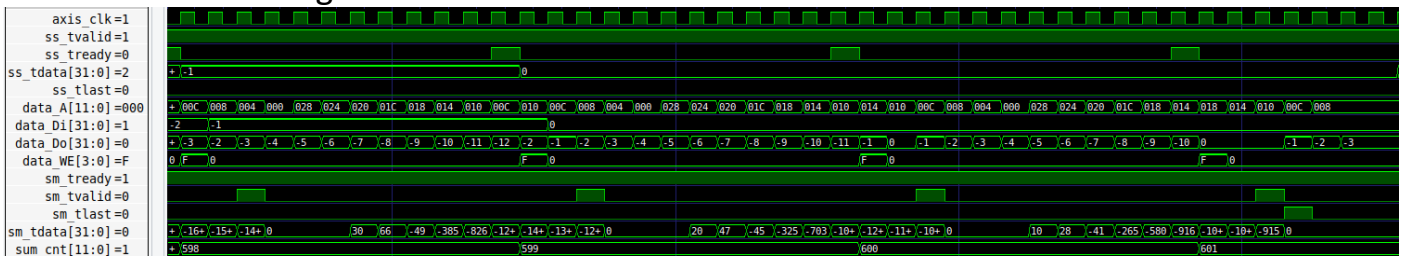
When host transfer input data to data RAM, we use ss_tvalid and ss_tready to do data transfer handshake. As both signals are asserted, it would trigger data_WE. In next cycle, data_Di is covered by ss_tdata.



As one output finished, sm_tvalid is asserted. The sm_tdata become the result of sum. Through the axi-stream interface, output is transferred to host.

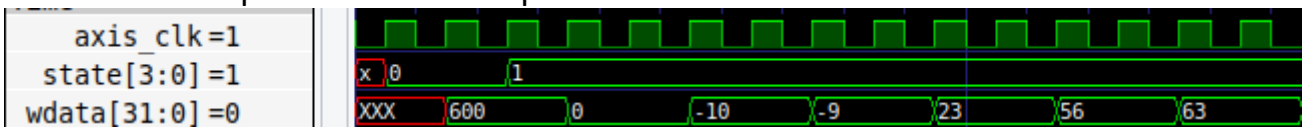


When FIR engine transfer the last output, it triggers sm_tlast signal. Host would know FIR engine finished the calculation.

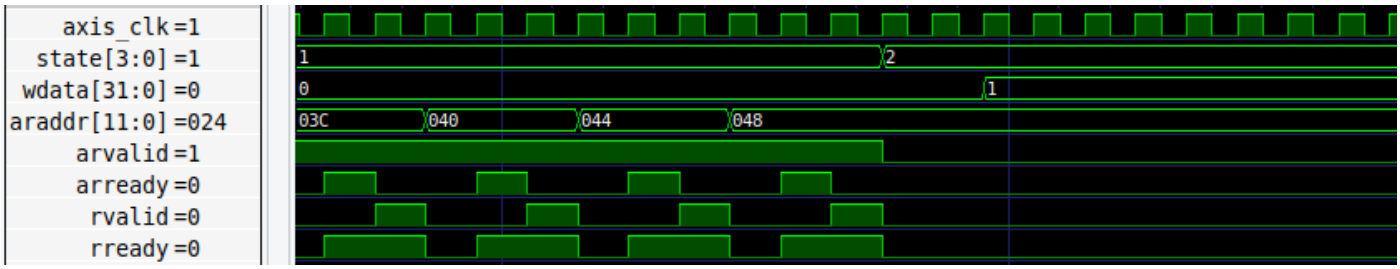


- FSM

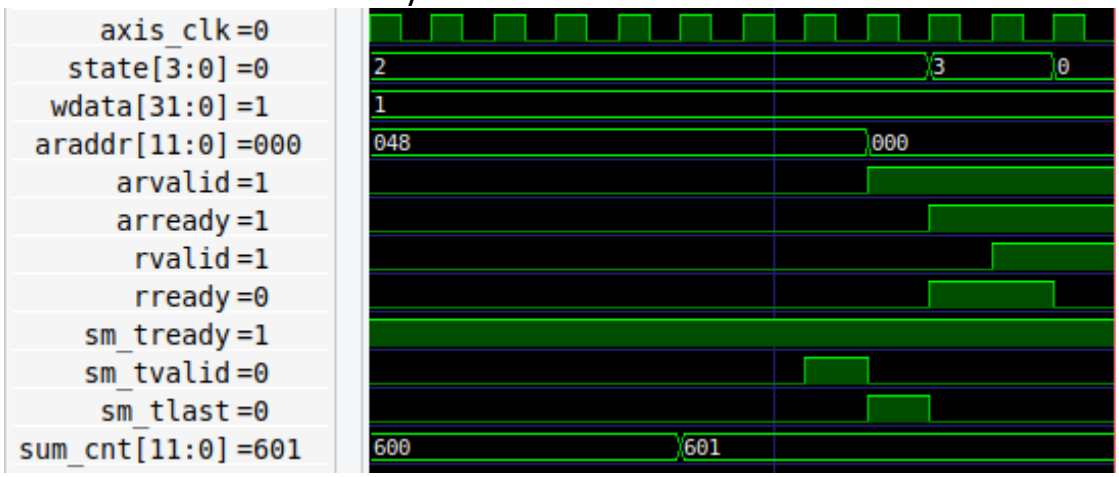
  State IDLE turns to state COEF when wdata is 600(data_length), host starts to transfer tap coefficients to tap RAM.



  As araddr is 12'h024, rvalid and rready are asserted, it means reading check finishes. Then state COEF turns to state CAL.



  When sm_tlast asserted, the last output finishes, state would turn to state DONE from state CAL. In state DONE, host would read configuration address map to check situation of engine. When configuration address becomes ap_done, state turns to state IDLE in next cycle.



Timing Report

Critical path (7ns)

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.076ns  (required time - arrival time)
  Source:                 mul_data_reg[16]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@3.600ns period=7.200ns})
  Destination:            mul_result0__1/PCIN[0]
                            (rising edge-triggered cell DSP48E1 clocked by axis_clk  {rise@0.000ns fall@3.600ns period=7.200ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            7.200ns  (axis_clk rise@7.200ns - axis_clk rise@0.000ns)
  Data Path Delay:        5.544ns  (logic 4.689ns (84.582%)  route 0.855ns (15.418%))
  Logic Levels:           1  (DSP48E1=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 9.328 - 7.200 )
    Source Clock Delay      (SCD):    2.456ns
    Clock Pessimism Removal (CPR):    0.184ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

- **Resource Usage**

FF and LUT

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	298	0	0	53200	0.56
LUT as Logic	298	0	0	53200	0.56
LUT as Memory	0	0	0	17400	0.00
Slice Registers	339	0	0	106400	0.32
Register as Flip Flop	339	0	0	106400	0.32
Register as Latch	0	0	0	106400	0.00
F7 Muxes	0	0	0	26600	0.00
F8 Muxes	0	0	0	13300	0.00

BRAM

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	140	0.00
RAMB36/FIFO*	0	0	0	140	0.00
RAMB18	0	0	0	280	0.00

- **Github Link**

https://github.com/nelson1216/SoC_fir