SOC HW3 FIR Design

m111061549

張耀明

Outline:

Date:2023/10/13

1. Overview

This project aims to design a 11-tap Fir filter that interfaces with the host(testbench) through AXI-interface, including AXI-LITE and AXI-Stream. The AXI-LITE module sends configuration write(block level protocol: ap_start) from the testbench to the FIR filter and reads the tap and configuration (block level protocol: ap_done, ap_idle). Meanwhile, the AXI-Stream is responsible for sending Fir input x[t] from the testbench to the FIR filter and receiving the FIR output as soon as an FIR calculation is completed.

Therefore, the design can be divided into two parts: Configuration part and Fir Dataflow part. Configuration part manages the AXI-LITE interface to ahndle the configuration inputs, while the Fir Dataflow part is responsible for the AXI-Stream interface and the calculation unit that generates FIR output.

In addition, the FIR can access two BRAM module that are designed in Behavior level, Tap_ram and Data_ram. Tap_ram stores the filter tap, while Data_ram stores the previous inputs required for calculation.

Moreover, the execution time can be divided into two stages: configuration stage and calculation stage. The execution begins with configuration stage, where host(testbench) sends FIR tap to the FIR filter to store them in tap_ram. Subsequently, the ap_start signal is set to activate the calculation stage, where FIR filter receives FIR input from host and performs the calculation to generate FIR output.

The overall structure is shown in Figure 1.

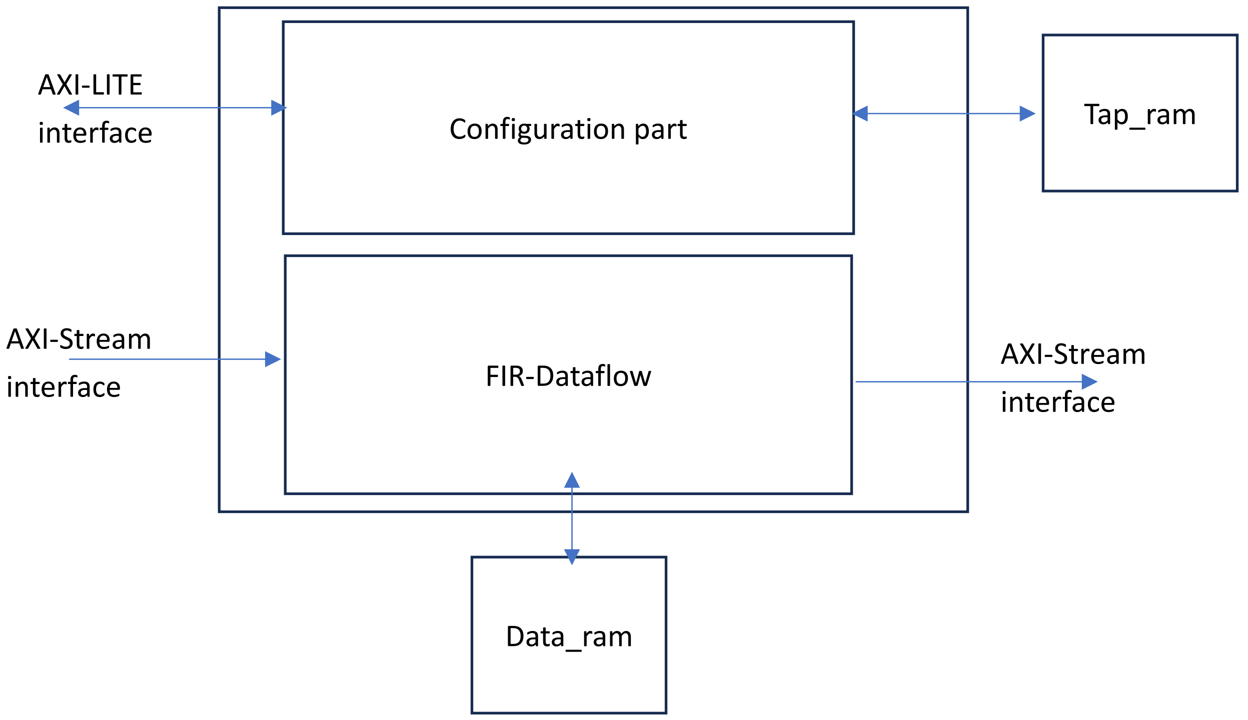


Figure 1 Overall structure

The next section will provide a detailed explanation of the design.

2. Block diagram

The block diagram is shown in Figure 2

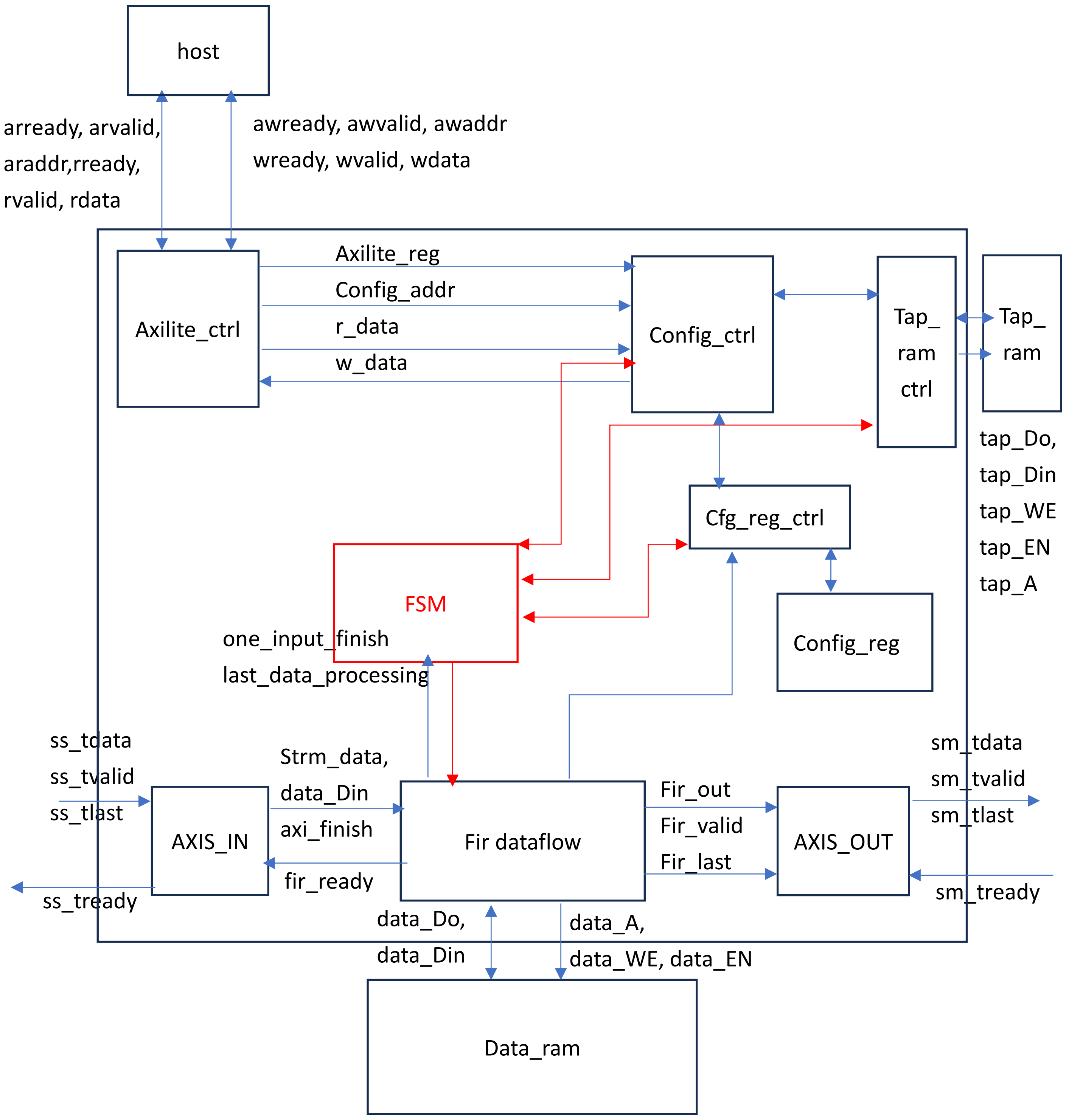


Figure 2 Block diagram

This FIR designed contains 9 blocks:

Configuration Part:

This part contains Axilite_ctrl, Config_ctrl, Tap_ram_ctrl, Cfg_reg_ctrl, config_reg. The primary purpose is to manage the AXI-LITE read write operation, including the handling of block level protocol signals and the tap transmission of tap value via AXI-LITE.

Axilite_ctrl: The Axilite_ctrl is responsible for managing write and read requests from the host. It also handles the transmission of data to and from the host., which includes the transfer of tap values and configuration write(block level protocol signal). The Axilite_ctrl module reads the register address and forward it to the Config_ctrl module.

Config_ctrl: The Config_ctrl module receives the awaddr and araddr from Axilite_ctrl, it process those request based on the address value, such as store/load the tap, write/read the block level protocol.

Tap_ram_ctrl:

Determine which module can write and read the tap_ram in each state.

Cfg_reg_ctrl:

Determine which module can write and read the configuration registers

Config_reg:

Stores the block level protocol

Fir Dataflow part:

AXIS_IN :

Receive fir input from axi-stream interface and send to Fir Dataflow module when Fir dataflow asks for a new data.

Fir_dataflow:

Perform the Fir calculation, when an output generated, it asks AXIS_IN module for a new data.

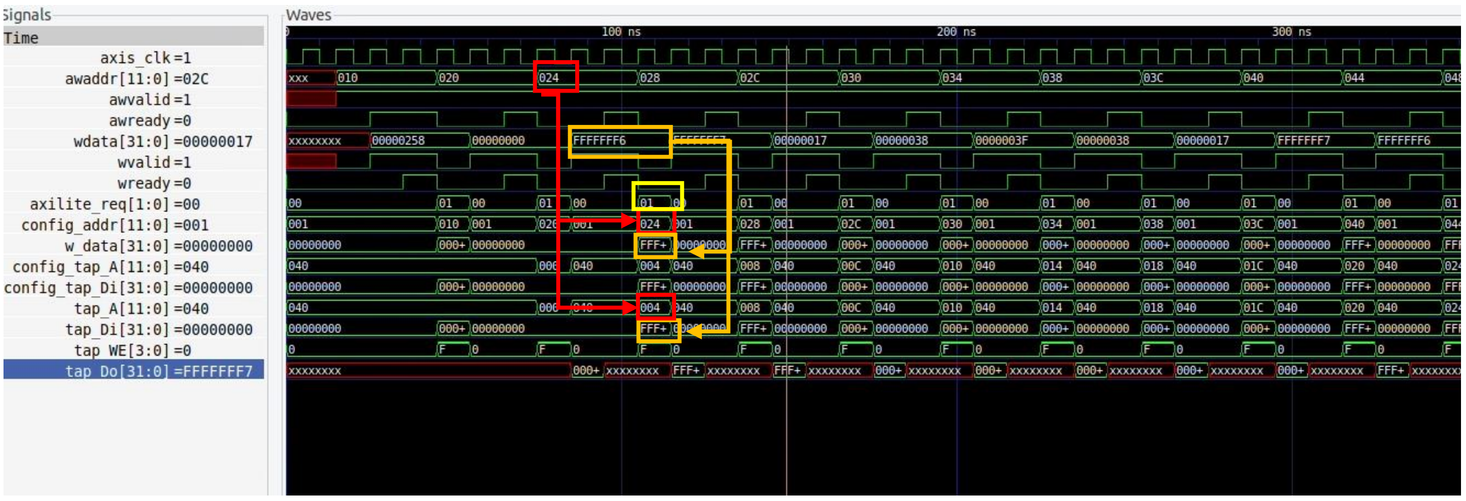
AXIS_OUT

Send fir output to axis interface.

3. Operation

Configuration read:

The Axilite module initiates a handshake with the axilite_interface and receives tap data. Subsequently, the Axilite module sends an axilite_request, 'awaddr,' and 'wdata' to the Config_ctrl module. The Config_ctrl module checks the request and the 'awaddr' to determine where to store the 'wdata. If awaddr is 0x00, it indicates that the wdata is ap_start. Config_ctrl write the wdata to the config_reg module through the cfg_reg_ctrl module. This module determines which module can write and read the configuration registers. If 'awaddr' is 0x10, it indicates that 'wdata' represents the 'data_length.' It's important to note that, in my design, 'data_length' will not be stored in either 'tap_ram' or 'config_reg. This is because my AXIS_IN module can determine if it has received the last input data by checking the 'tlast' signal. If 'awaddr' is equal to or greater than 0x20, the 'wdata' represents the FIR taps. The Config_ctrl module then stores 'wdata' through the tap_ram_ctrl module, which manages read and write access to the tap_ram and assigns it to the corresponding RAM address.



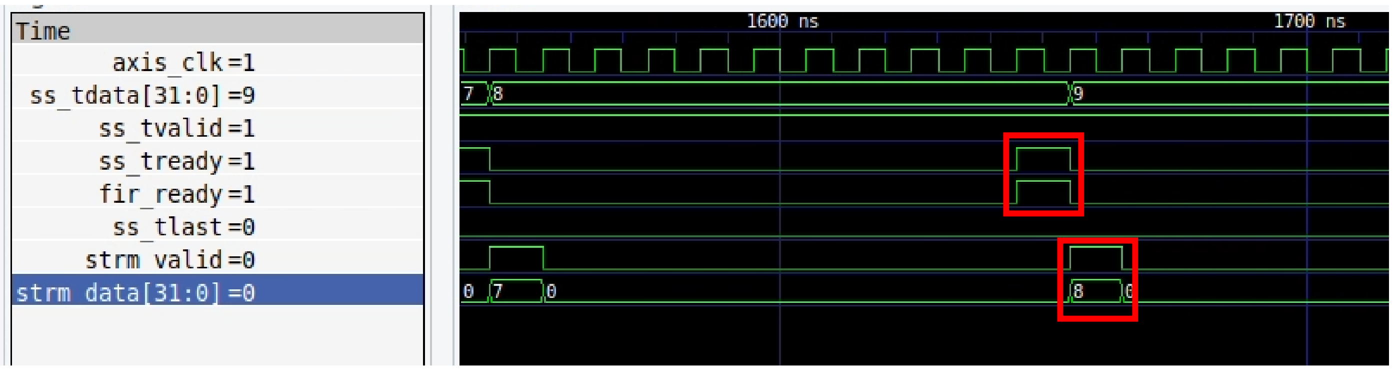
```
Signals
Time
axis_clk=1
awaddr[11:0]=02C
awvalid=1
awready=0
wdata[31:0]=00000017
wvalid=1
wready=0
axilite_req[1:0]=00
config_addr[11:0]=001
w_data[31:0]=00000000
config_tap_A[11:0]=040
config_tap_Di[31:0]=00000000
tap_A[11:0]=040
tap_Di[31:0]=00000000
tap_WE[3:0]=0
tap_Do[31:0]=FFFFFFF7
```

Fir_caluculation:

In my design, the AXIS_IN module handles the axis input, the fir_dataflow performs the fir calculations, and the AXIS_OUT module manages the axis output.

1.AXIS_IN:

This module manages the axis input. Whenever the Fir_dataflow sends the 'fir_ready' signal to request new input, the AXIS_IN module initiates a handshake with the Axi_stream interface to receive the new data.



```
Time
axis_clk=1
ss_tdata[31:0]=9
ss_tvalid=1
ss_tready=1
fir_ready=1
ss_tlast=0
strm_valid=0
strm_data[31:0]=0
```

As shown above, when 'fir_ready' is set, the AXI_IN module sets 'ss_tready.' When 'ss_tready' and 'ss_tvalid' are both set, the module receives 'ss_tdata' and sends it to the fir_dataflow module through 'strm_data' and 'strm_valid,' where 'strm_valid' indicates valid data for the fir_dataflow module.

2.Fir_dataflow:

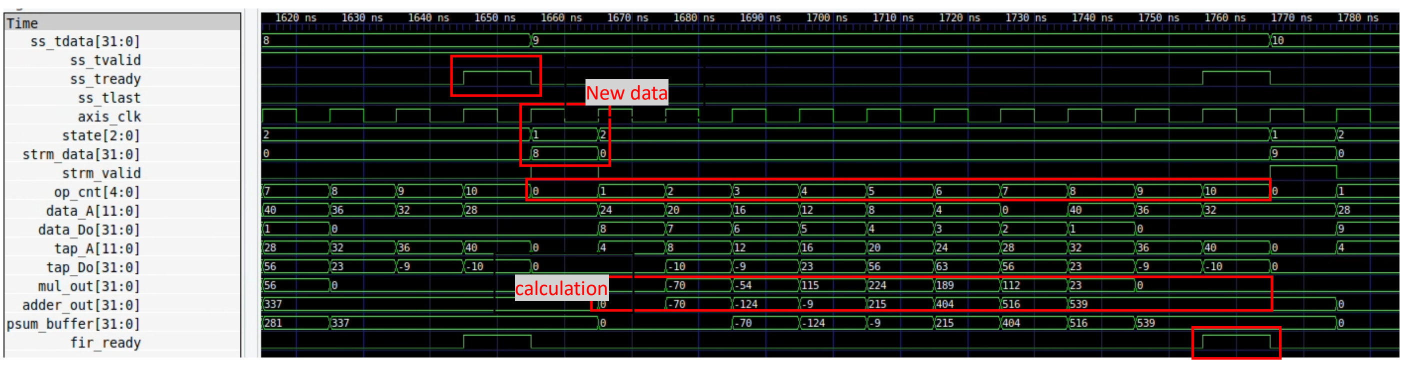
This module performs the FIR calculation. It has an FSM to manage the workflow.

State	Actions	Next_state
Idle	Wait for axi_stream data. At this state, the configuration read and write is available.	When ap_start == 1, go to Store_1_input
Store_1_input	Receive a new input from AXI_IN module and store it in data_ram. At the same time, a new output is generated and send to AXI_OUT module.	Calculation
Calculation	Perform the calculation.	When one_input_finish and last_data_processing are both set, go to Finish state. If only one_input_finish is set, go to store_1_input. Otherwise, remain in same state.
Finish	Set ap_done	When rvalid is set, go to AP_DONE
AP_DONE	Clean ap_done and set AP_IDLE	When rvalid is set, go to AP_IDLE
AP_IDLE	Delay one cycle	Back to IDLE

Fir_dataflow sets the 'fir_ready' signal to request new data at the same time as a new output is generated. In the next clock cycle, the new input is send via 'strm_data' with 'strm_valid' signal set to 1. At that moment, the state transitions to “store_1_input”.

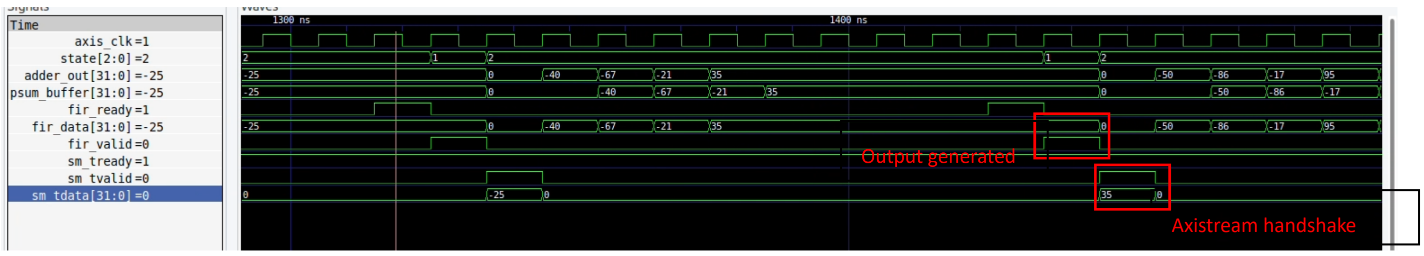
At 'store_1_input,' the data is prepared for storage in Data_ram for future use, and 'op_cnt' is reset to 0. The state transitions to 'calculation' in the next cycle. During this state, 'op_cnt' starts counting from 1 to 10, generating the 'ram_addr' to access the FIR taps and previous inputs from tap_ram and data_ram. The output of data_ram and tap_ram are sent to the calculation unit, with contains a multiplier and adder

There is only one adder and one multiplier. To generate an output data, the calculation unit must perform calculations 11 times, requiring 11 clock cycles. When 'op_cnt' counts to 10, it indicates the completion of one calculation, resulting in a new output. To fully utilize the FIR_Dataflow, the fir_ready signal set when op_cnt reaches 10, and the next calculation begins



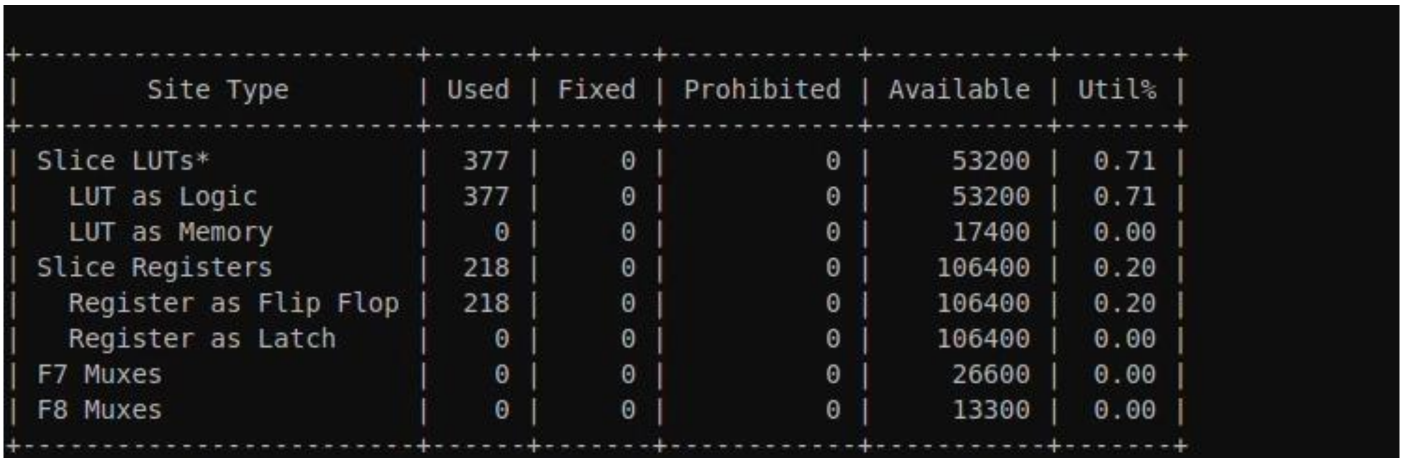
3.AXIS_OUT:

This module manages the axi_stream output. When a new output data is generated, the 'fir_valid' signal is set by the fir_dataflow module. The AXIS_OUT module receives the 'fir_data' when 'fir_valid' is set, and it initiates a handshake with the axi_stream interface to send the output to the host.

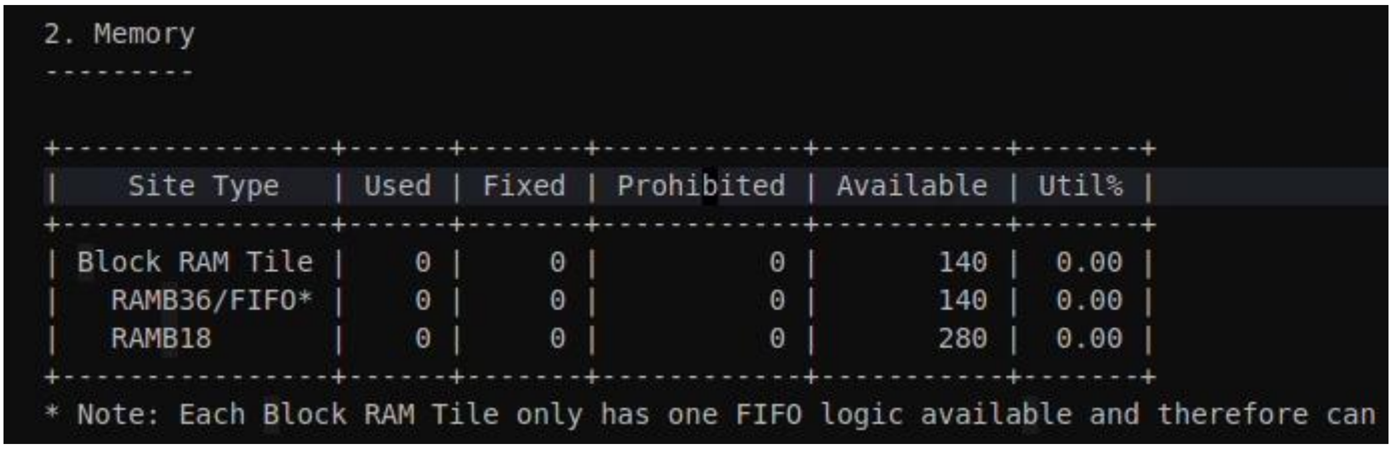


4.Resource usage

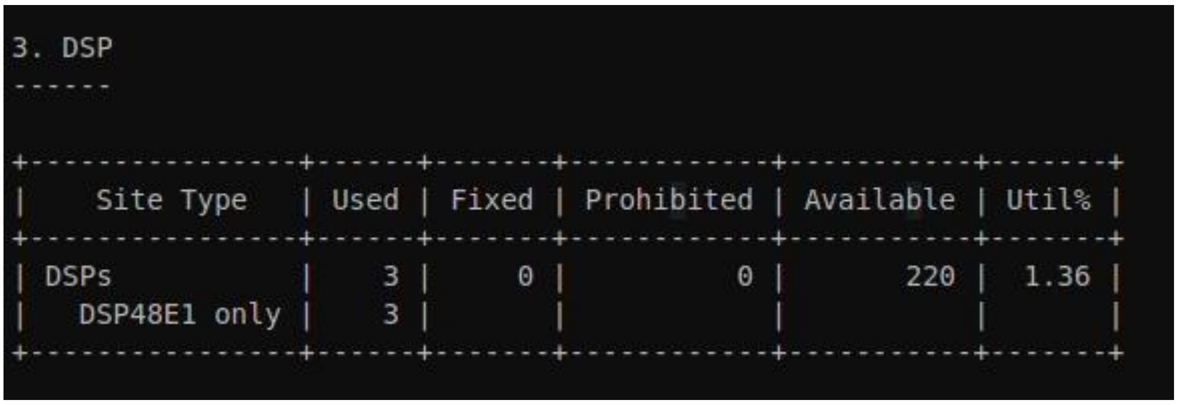
FF and LUT:


```
+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |  377 |     0 |          0 |     53200 |  0.71 |
|   LUT as Logic          |  377 |     0 |          0 |     53200 |  0.71 |
|   LUT as Memory         |    0 |     0 |          0 |     17400 |  0.00 |
| Slice Registers         |  218 |     0 |          0 |    106400 |  0.20 |
|   Register as Flip Flop |  218 |     0 |          0 |    106400 |  0.20 |
|   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 |
+-------------------+------+-------+------------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can
```

DSP:

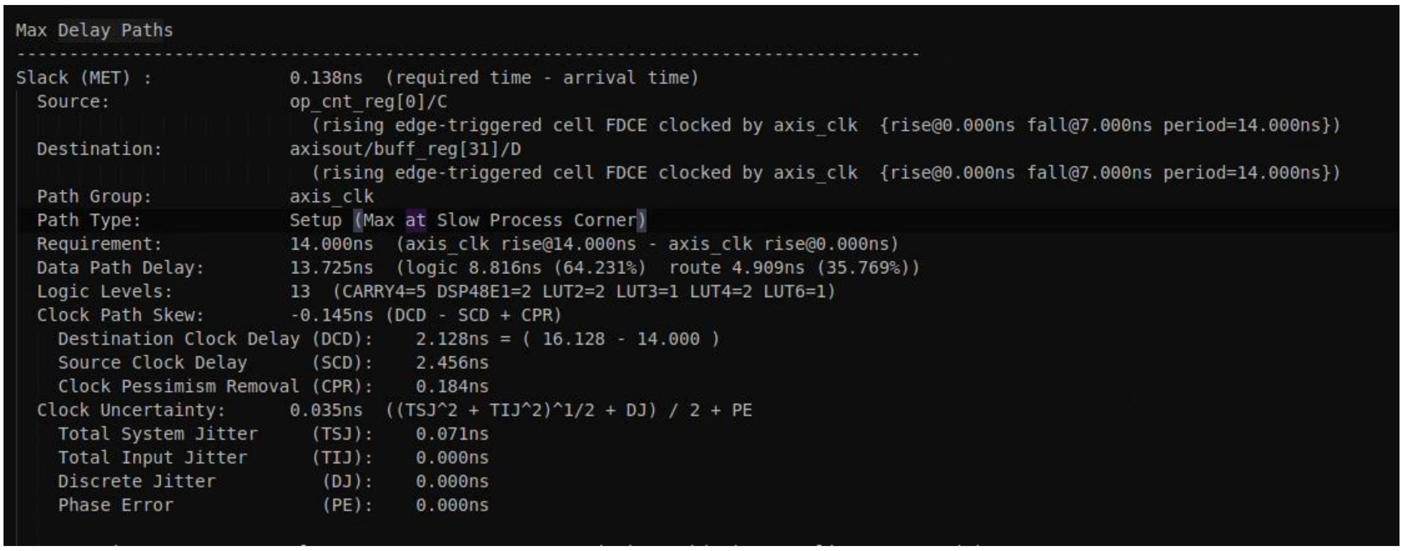

```
3. DSP
------

+----------------+------+-------+------------+-----------+-------+
|    Site Type   | Used | Fixed | Prohibited | Available | Util% |
+----------------+------+-------+------------+-----------+-------+
| DSPs           |    3 |     0 |          0 |       220 |  1.36 |
|   DSP48E1 only |    3 |       |            |           |       |
+----------------+------+-------+------------+-----------+-------+
```

5.Report

Timing

Clk cycle:14ns

Critical path:



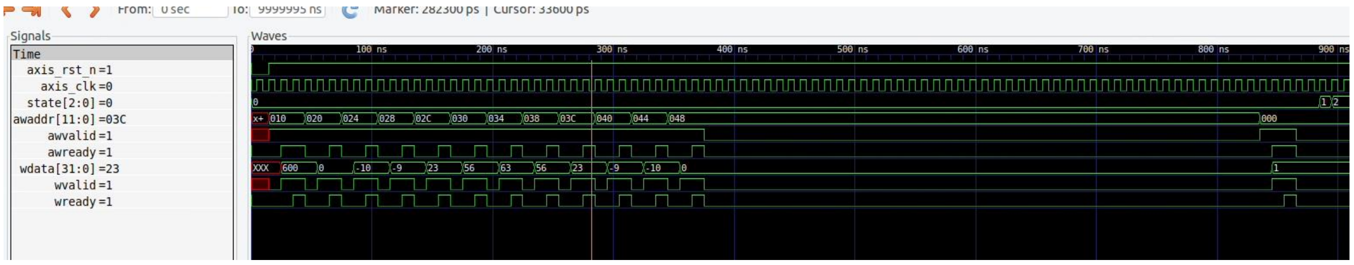
```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.138ns  (required time - arrival time)
  Source:                 op_cnt_reg[0]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@7.000ns period=14.000ns})
  Destination:            axisout/buff_reg[31]/D
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@7.000ns period=14.000ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            14.000ns  (axis_clk rise@14.000ns - axis_clk rise@0.000ns)
  Data Path Delay:        13.725ns  (logic 8.816ns (64.231%)  route 4.909ns (35.769%))
  Logic Levels:           13  (CARRY4=5 DSP48E1=2 LUT2=2 LUT3=1 LUT4=2 LUT6=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 16.128 - 14.000 )
    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
```

6.Simulation

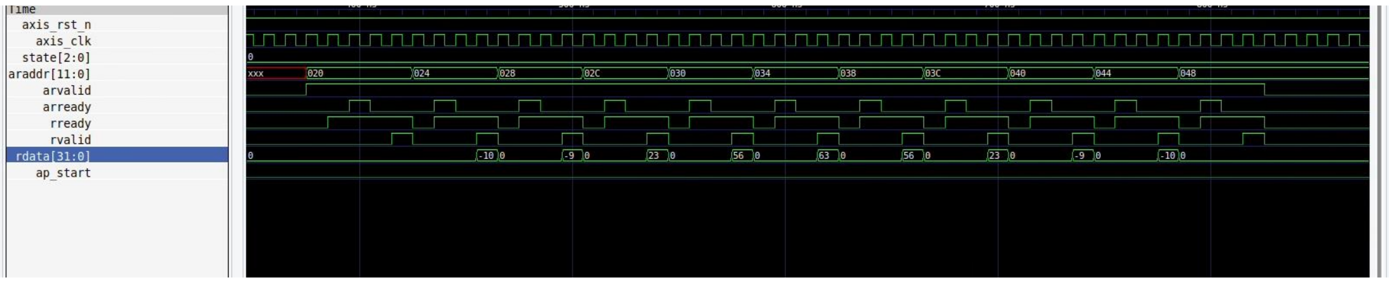
Waveform :

(1) coefficient program, and read back

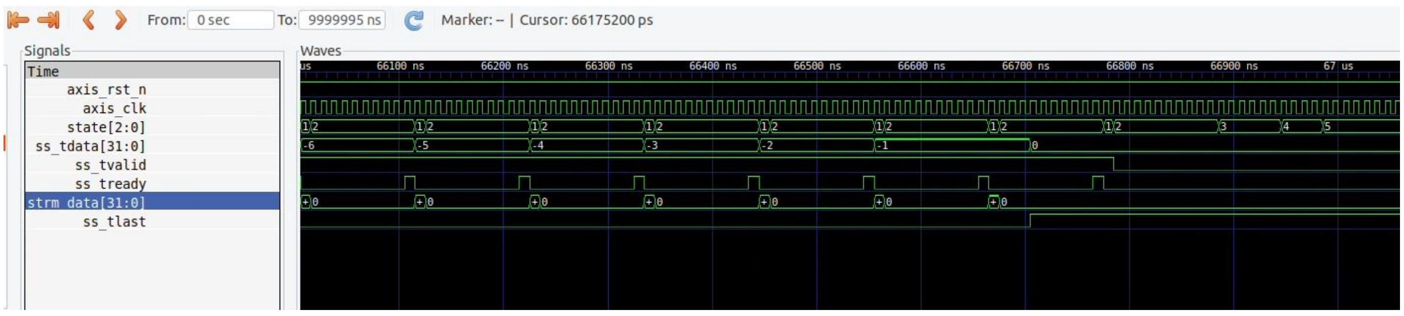
coefficient program



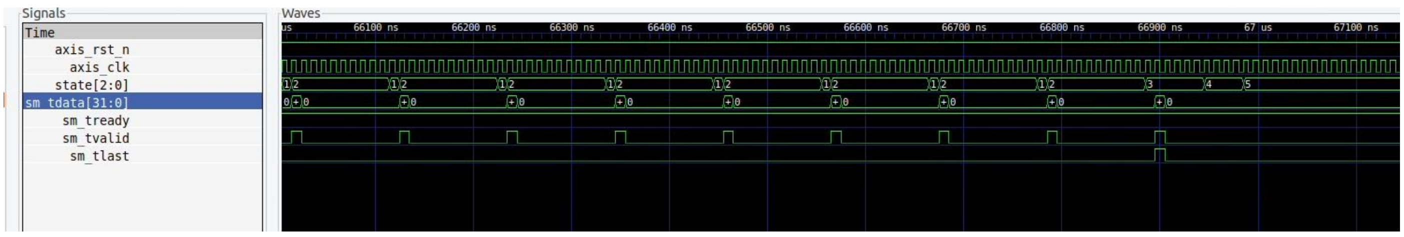
read back



(2) Data-in stream-in

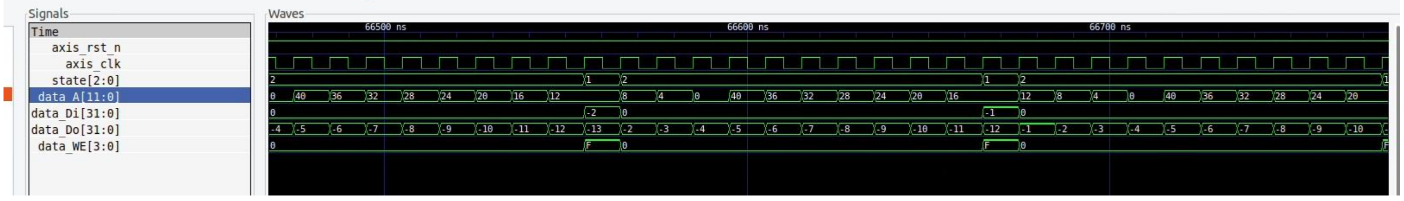


(3) Data-out stream-out



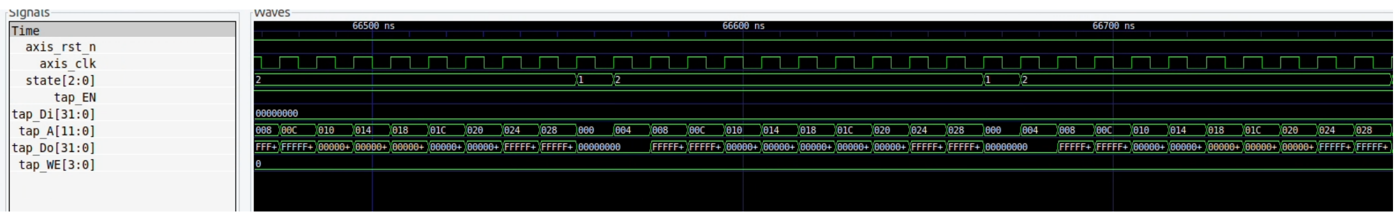
(4) RAM access control

Data_ram:

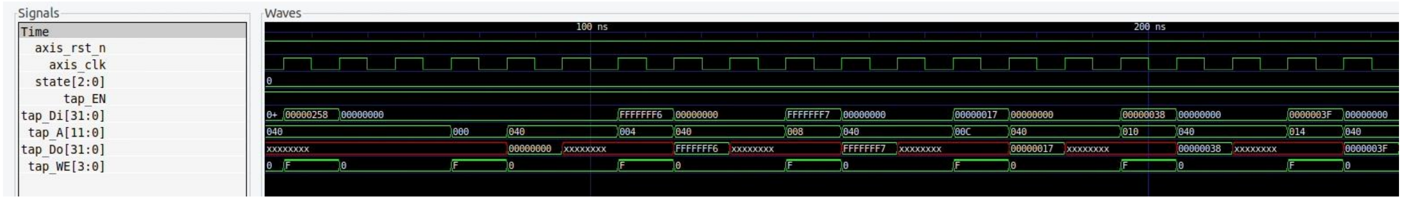


Tap_ram:

Write

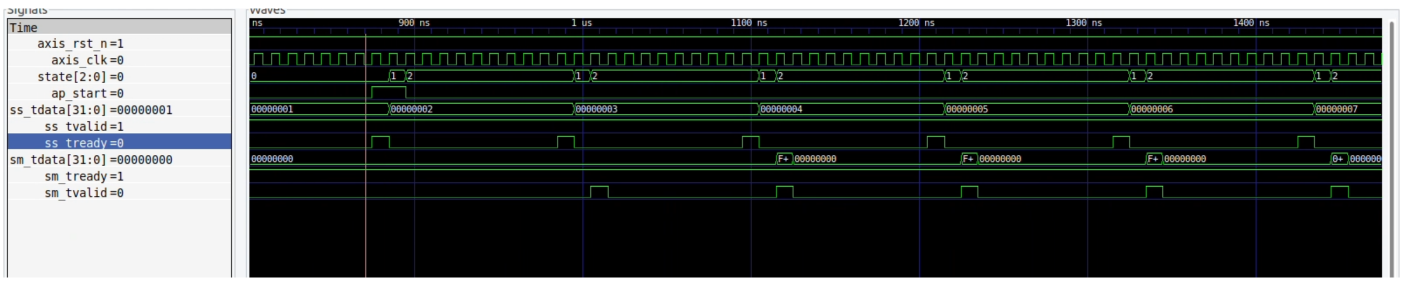


Read from axilite



(5) FSM

FSM during calculation



Github link: https://github.com/s095339/SOC_lab_fir