SOC Design Lab3 FIR

112061620 郭邑哲

1. Overview

這次的實驗為設計一個 Tap (Coefficients)數目為 11 的 FIR，其運算公式如下:

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

而此次實驗輸入資料(ss_tdata) x[t]為 600 筆，因此在與係數做相乘及相加後，會輸出 600 筆輸出資料(sm_tdata)y[t]到 testbench(host)端，其運算規則如下。其中資料(data)的傳輸是藉由 AXI-Stream interface 進行；block level protocol 中的 ap_start,ap_idle,ap_done 則是由 AXI-Lite interface 進行。

$$y_0 = x_0 h_0 + 0 \cdots 0 \quad (10個0)$$

$$y_1 = x_1 h_0 + x_0 h_1 + 0 + \cdots 0 \quad (9個)$$

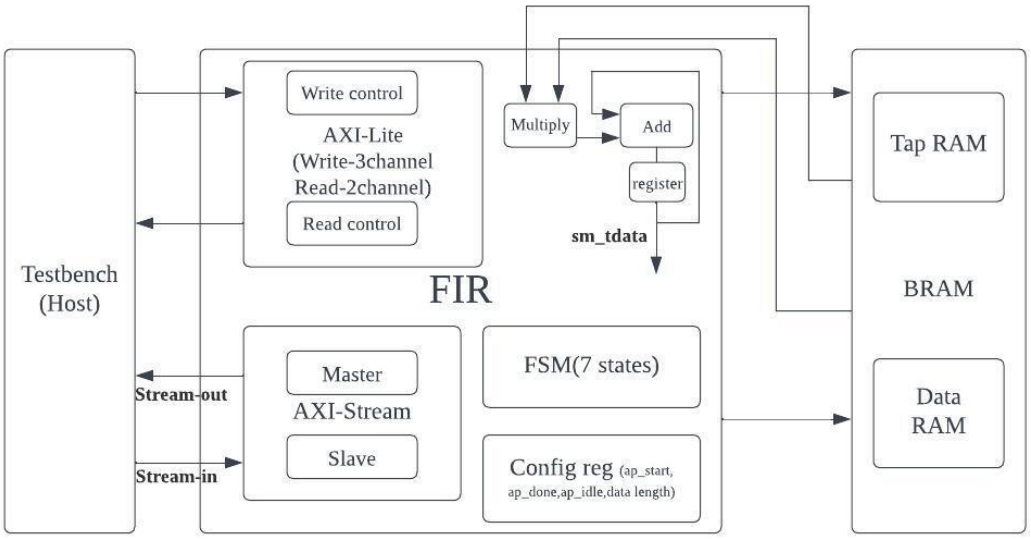
$$y_2 = x_2 h_0 + x_1 h_1 + x_0 h_2 + \cdots 0 \cdots \quad (8個)$$

$$\vdots$$

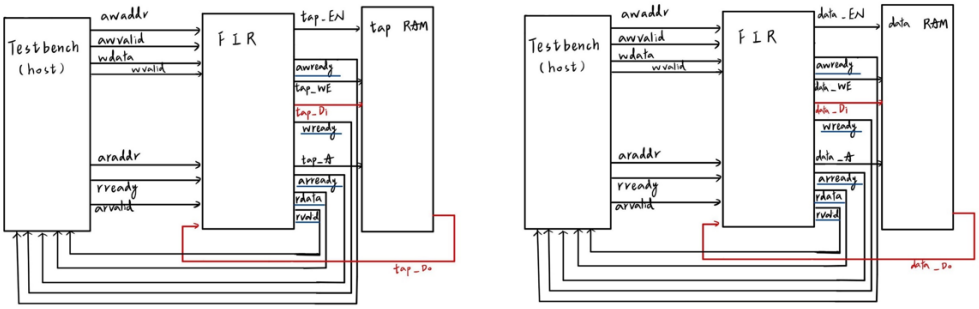
$$y_{10} = x_0 h_{10} + x_1 h_9 + x_2 h_8 + x_3 h_7 + x_4 h_6 + x_5 h_5 + x_6 h_4 + x_7 h_3 + x_8 h_2 + x_9 h_1 + x_{10} h_0$$

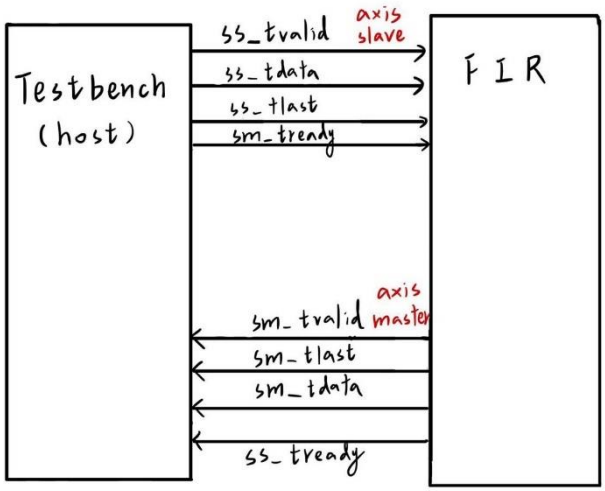
此次實驗也要求只能使用 1 個乘法器及 1 個加法器，因此我們必須先將輸入資料(x[t])及係數(coefficients)透過 RAM 存起來，當我們需要用到時，再取出做運算。

2. Block diagram



Control signals





SRAM Control Signals:

1. Tap/Data WE:為 Write enable signal，當 WE 為 1 時，資料才能寫入 Block RAM。
2. Tap/Data EN:當 EN 為 1 時，BRAM 才能進行寫入及讀出，相當於 SRAM 的電源。
3. Tap/Data Di:為寫入 SRAM 的資料
4. Tap/Data Do:為讀出 SRAM 的資料
5. Tap/Data Address:為寫入資料及讀出資料的 SRAM 之地址

AXI-Lite Control Signals:

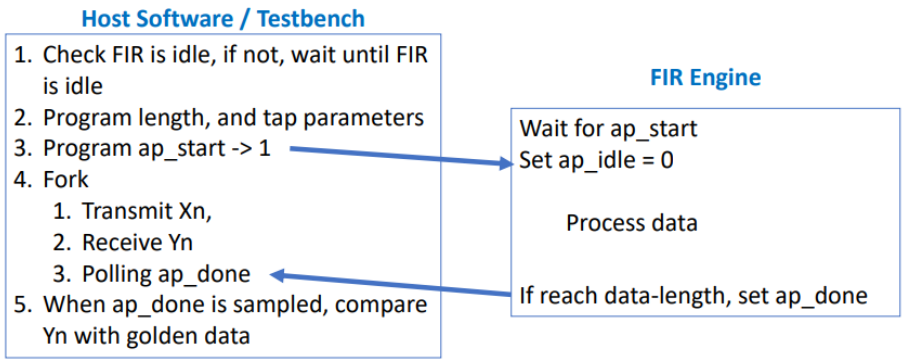
1. Valid:Master is ready to output data to Slave
2. Ready:Slave is ready to receive data from Master
3. Last:表最後一筆資料傳輸完成
4. Data:Master 和 Slave 間傳輸的資料

AXI-Stream Control Signals:

1. awready:Address write ready，Slave 端傳出此訊號，表示它準備好接收 write address 的訊號。
2. wready:Write data ready，Slave 端傳出此訊號，表示它準備好接收 data。

3. awvalid: Address write valid:Master 端傳出此訊號，表示它準備好發送 write address 的訊號。
4. awaddr:Write address，寫入的地址，此次實驗為 32bits。
5. wvalid:Write data valid，Master 端傳出此訊號，表示它準備好發送 data。
6. wdata:Write Data，寫入的資料。
7. arready: Address read ready，Slave 端傳出此訊號，表示它準備好接收 read address 的訊號。
8. rready:Read data ready，Master 端傳出此訊號，表示它準備好接收讀出的資料。
9. arvalid: Address read valid，Master 端傳出此訊號，表示它準備好發送要讀出的地址。
10. araddr: Read address，讀出的地址，此次實驗為 32bits
11. rvalid: Read data valid，Slave 端傳出此訊號，表示它準備好發送讀出的資料。
12. rdata: Read Data，讀出的資料。

3. Describe operation



Note: Transmit Xn (stream-in), Receive Yn (stream-out) and Polling ap_done (axilite) are running concurrently. They are using different interface and do not interfere each other

設計流程:先將 Tap 存進 Tap RAM 中，等 ap_start 為 1 後，開始將資料 x[t]存進 Data RAM 中，進行 shift，並與 tap(coefficients)相乘相加，輸出 Data(y[t])，當輸出數目為 data_length 時，則 ap_done 為 1，並與 golden data(正解)進行比較。

- How to receive data-in and tap parameters and place into SRAM?

  首先，Tap parameters 是用 AXI-Lite 寫入及讀出，當 awvalid 及 wvalid 為 1 且 awready 及 wready 為 1 時，此時可將資料及地址寫入 SRAM 中。而 data-in(x[t])則是利用 AXI-Stream 來接收 testbench(host)端傳入的資料，當 ss_tvalid 及 ss_tready 時，此時可接收資料(x[t])。

- How to access ShiftRAM and TapRAM to do computation?

  一開始我先將第一筆資料存入，並將 Datan RAM 的其他地址填入 0，以防止出現 unknown 的情形，此時只需考慮寫入，因此只有經過 11

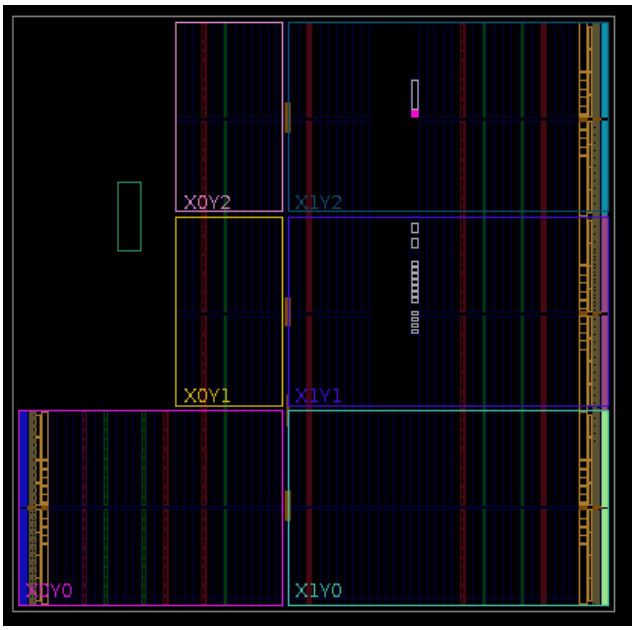
個 CLK。之後有第二筆資料要進入時，需先將前一筆資料先做讀出再做寫入(shift data)，如此先前的資料才不會被覆蓋掉。讀出需 2CLK；寫入則需 1CLK，11 個地址共需 33CLK，當資料移位完後，我們需將 DATA 及 TAP 讀出相乘後存到一暫存器，再與後面的地址之 DATA 及 TAP 讀出相乘後再相加，以此類推，總共有 11 筆數字相加，需 22CLK。

- How ap_done is generated?

  我利用 config-register 儲存 ap_start 及 ap_done 及 ap_idle 及 data_length。並利用一個加法器 i 來進行運算，每輸出一筆資料 i 會加 1，當 i=600 時表示已輸出 600 筆資料，此時 ap_done 會為 1，及 config-register 為 2。

4. Resource usage

下圖為合成後的電路圖:



1.LUT and FF

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	231	0	0	53200	0.43
LUT as Logic	231	0	0	53200	0.43
LUT as Memory	0	0	0	17400	0.00
Slice Registers	230	0	0	106400	0.22
Register as Flip Flop	43	0	0	106400	0.04
Register as Latch	187	0	0	106400	0.18
F7 Muxes	0	0	0	26600	0.00
F8 Muxes	0	0	0	13300	0.00

2.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

3.DSP

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	3	0	0	220	1.36
DSP48E1 only	3				

4.Report Cell Usage

Report Cell Usage:

	Cell	Count
1	BUFG	6
2	CARRY4	12
3	DSP48E1	3
4	LUT1	2
5	LUT2	154
6	LUT3	38
7	LUT4	12
8	LUT5	44
9	LUT6	58
10	FDRE	42
11	FDSE	1
12	LD	183
13	LDC	1
14	LDCP	1
15	LDP	2
16	IBUF	157
17	OBUF	169

5. Timing report&&Synthesis report

- Try to synthesize the design with maimum frequency
  在尋找最大操作頻率(maximum frequency)時，須先找到最長路徑，從 Timing report 中可以看到在 FIR 進行 32bits 相乘相加的地方，能看到較大的 Data Path Delay，此時我的最大 Data Path Delay 為 15.285ns，理想上大約為 65.4M Hz。

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.679 ns	Worst Hold Slack (WHS):	0.148 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	123	Total Number of Endpoints:	123	Total Number of Endpoints:	44

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             1.679ns  (required time - arrival time)
  Source:                 data_Do[16]
                            (input port clocked by axis_clk  {rise@0.000ns fall@5.000ns
period=20.000ns})
  Destination:            sm_tdata[31]
                            (output port clocked by axis_clk  {rise@0.000ns fall@5.000ns
period=20.000ns})
  Path Group:             axis_clk
  Path Type:              Max at Slow Process Corner
  Requirement:            20.000ns  (axis_clk rise@20.000ns - axis_clk rise@0.000ns)
  Data Path Delay:        15.285ns  (logic 11.586ns (75.799%)  route 3.699ns (24.201%))
  Logic Levels:           12  (CARRY4=5 DSP48E1=2 IBUF=1 LUT2=2 LUT6=1 OBUF=1)
  Input Delay:            2.000ns
  Output Delay:           1.000ns
  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

    Location             Delay type                Incr(ns)  Path(ns)    Netlist
Resource(s)
```

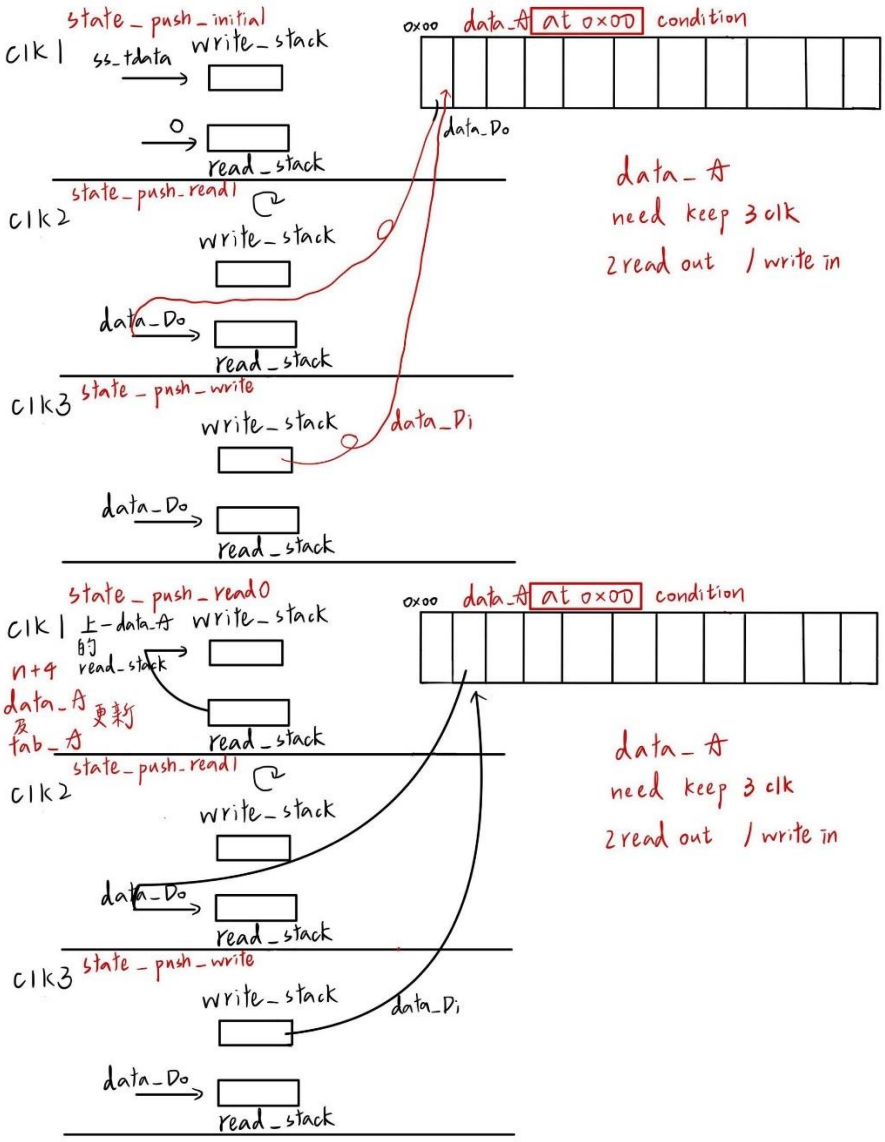
```
                         (clock axis_clk rise edge)
                                                      20.000    20.000
                         clock pessimism               0.000    20.000
                         clock uncertainty            -0.035    19.965
                         output delay                 -1.000    18.965
  -----------------------------------------------------------------------
                         required time                          18.965
                         arrival time                          -17.285
  -----------------------------------------------------------------------
                         slack                                   1.679
```

- Report timing on longest path,slack

  Timing report 報告中顯示，Data Path Delay 時間為 15.285ns，為我的 Data arrival time，而根據我所設置的 Input delay,Output Delay, Clock Uncertainty 時間分別為 2ns, 1ns, 0.035ns，我給定 clock cycle time 是 20ns，因此我的 Data required time 為 16.965ns(20-2-1-0.035)，因此我整個電路最長路徑的 slack 約為 1.679ns。符合題目要求，只用 1 adder 及 1multiplier。

```
Detailed RTL Component Info :
+---Adders :
      2 Input   32 Bit       Adders := 1
      2 Input   10 Bit       Adders := 1
      2 Input    6 Bit       Adders := 1
+---Registers :
                32 Bit    Registers := 1
                12 Bit    Registers := 1
                10 Bit    Registers := 1
                 6 Bit    Registers := 1
                 3 Bit    Registers := 1
                 1 Bit    Registers := 1
+---Multipliers :
               32x32  Multipliers := 1
+---Muxes :
      2 Input   32 Bit        Muxes := 3
      8 Input   32 Bit        Muxes := 5
      5 Input   32 Bit        Muxes := 1
      2 Input   12 Bit        Muxes := 3
      8 Input   12 Bit        Muxes := 1
      2 Input    6 Bit        Muxes := 2
      8 Input    6 Bit        Muxes := 1
      2 Input    4 Bit        Muxes := 1
      8 Input    4 Bit        Muxes := 1
      2 Input    3 Bit        Muxes := 3
```

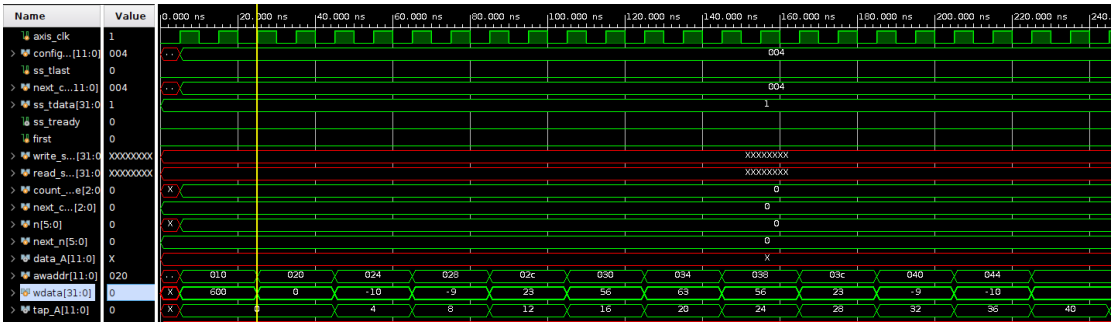
6. Data RAM FSM 相關推導圖

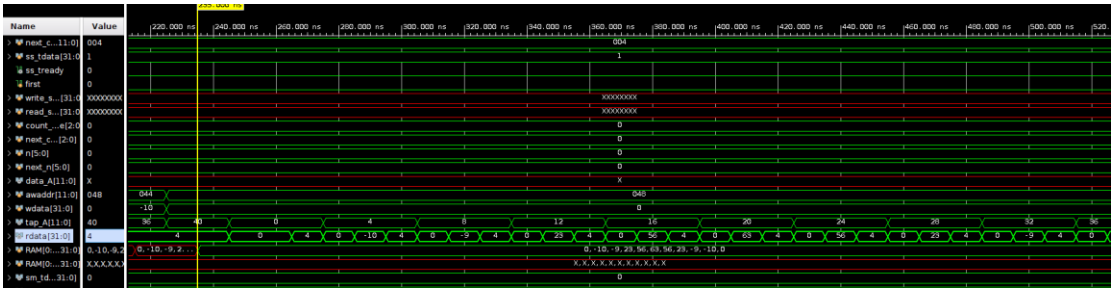


7. Simulation Waveform

- Coefficient program,and read back

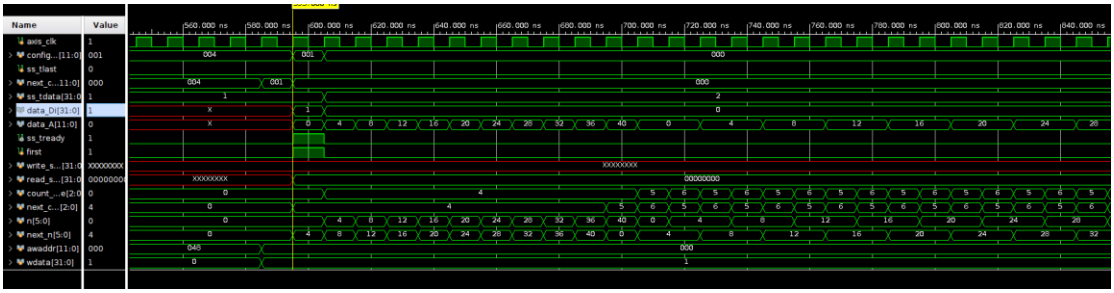
awaddr 為寫入地址，wdata 為寫入資料，可從圖中看到依序將資料存進 Tap RAM 中。



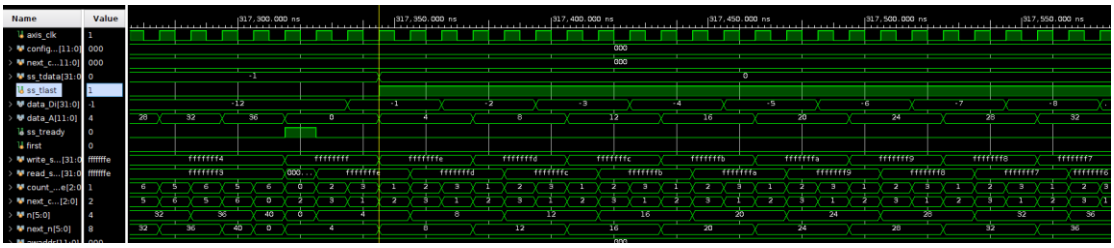


- Data-in, Stream-in

從圖中能看到 data_A 有對應的 data_Di。從 0x00 開始存入 ss_tdata 第一筆資料 1。

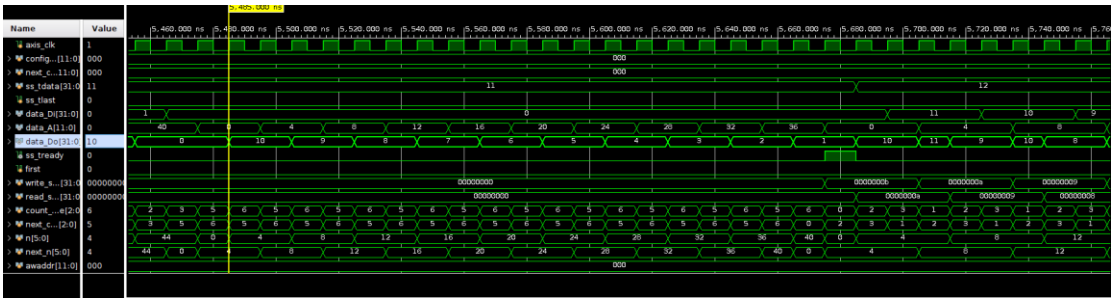


當最後一筆輸入資料傳完後，ss_tlast 升起為 1。

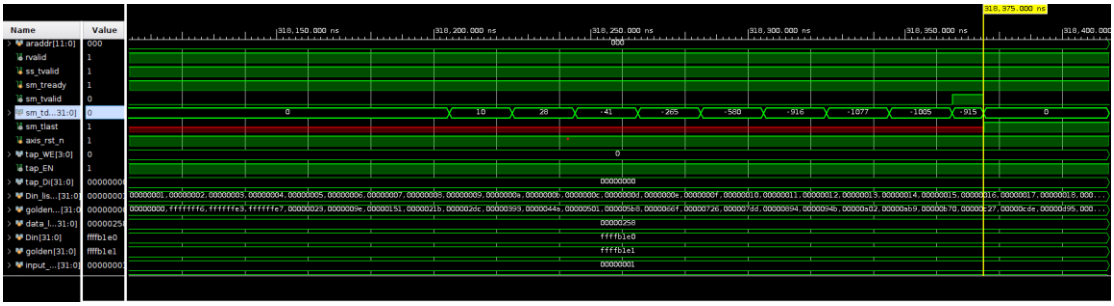


- Data-out,Stream-out

此時為讀取資料，可以從圖中看到 data_A 有對應的 data_Do。

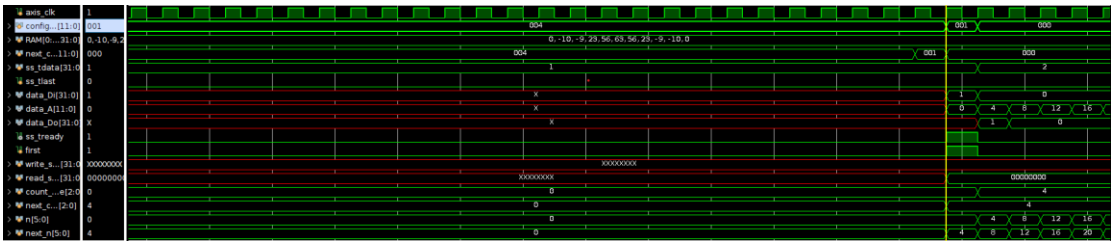


當最後一筆資料輸出完後，sm_tlast 為 1。

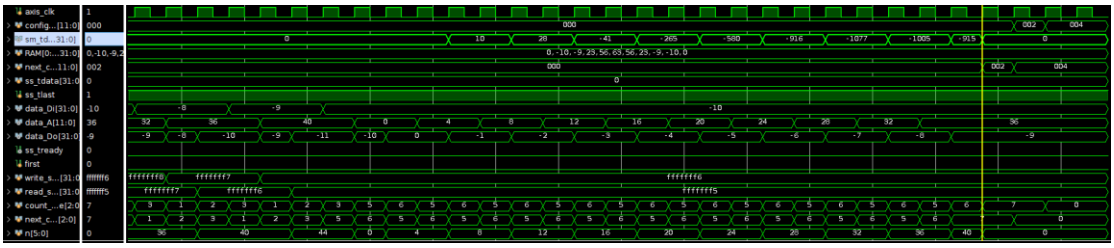


- RAM access control

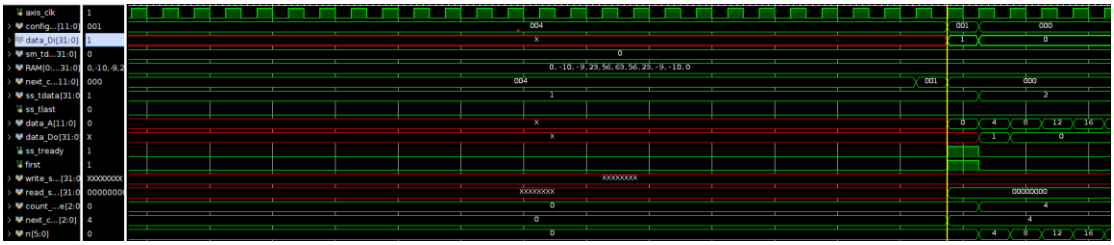
Tap 寫入完成後，ap_start=1，啟動 FIR



當輸出完最後一筆資料時，ap_done 為 1



一開始 FIR 為 ap_idle=1 表閒置，啟動 FIR 後 ap_idle 轉為 0 表正在工作。

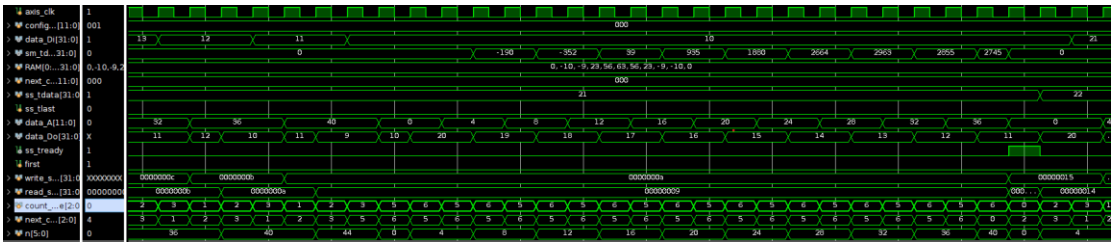


- Clock cycles from ap_start to ap_done

從圖中得知，花費(318385-595)/10=31779 個 Clock 才完成

- FSM during calculation

我的 FSM 設計為 7 個 states，因此能在圖中看到狀態的轉變。



Github link:https://github.com/816-allen/SOC-Lab-FIR

Reference:https://www.realdigital.org/doc/a9fee931f7a172423e1ba73f66ca4081