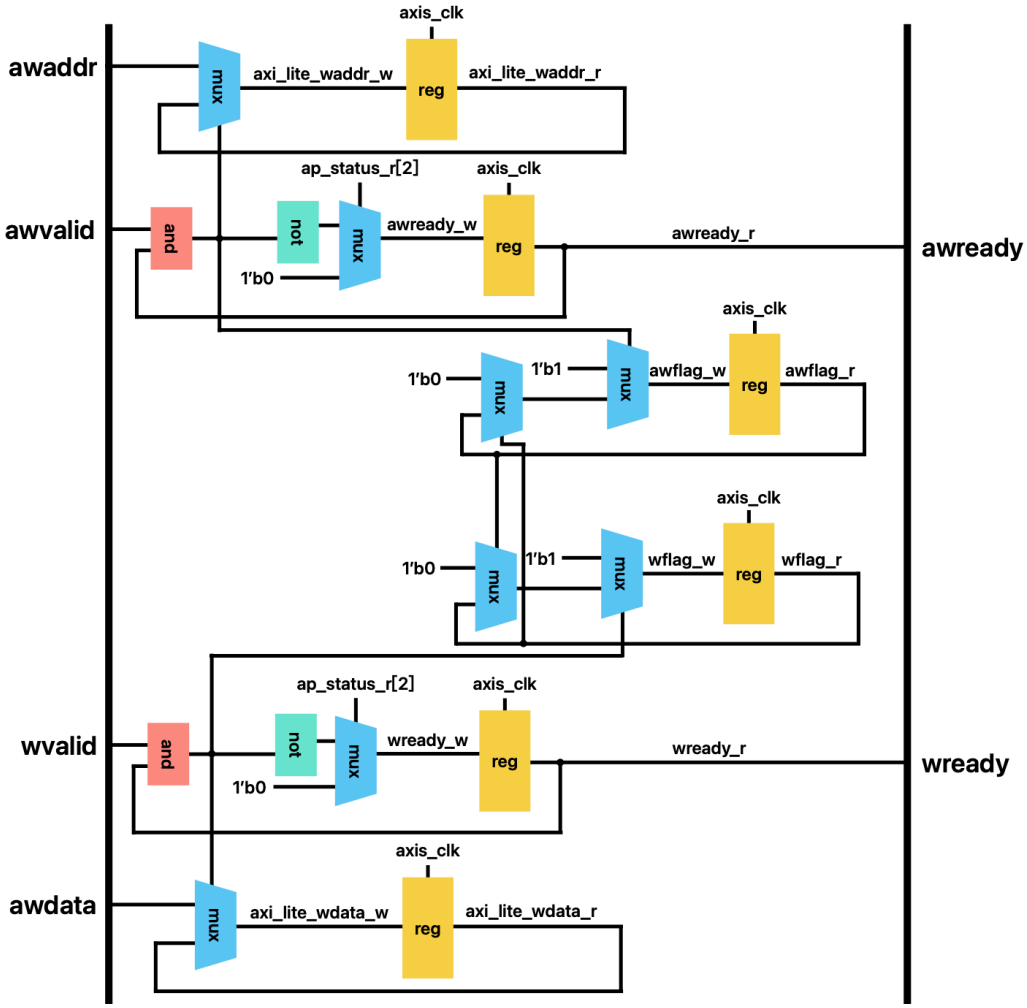
系統晶片設計 lab3 report

學號：R12943012 姓名：蔡東翰

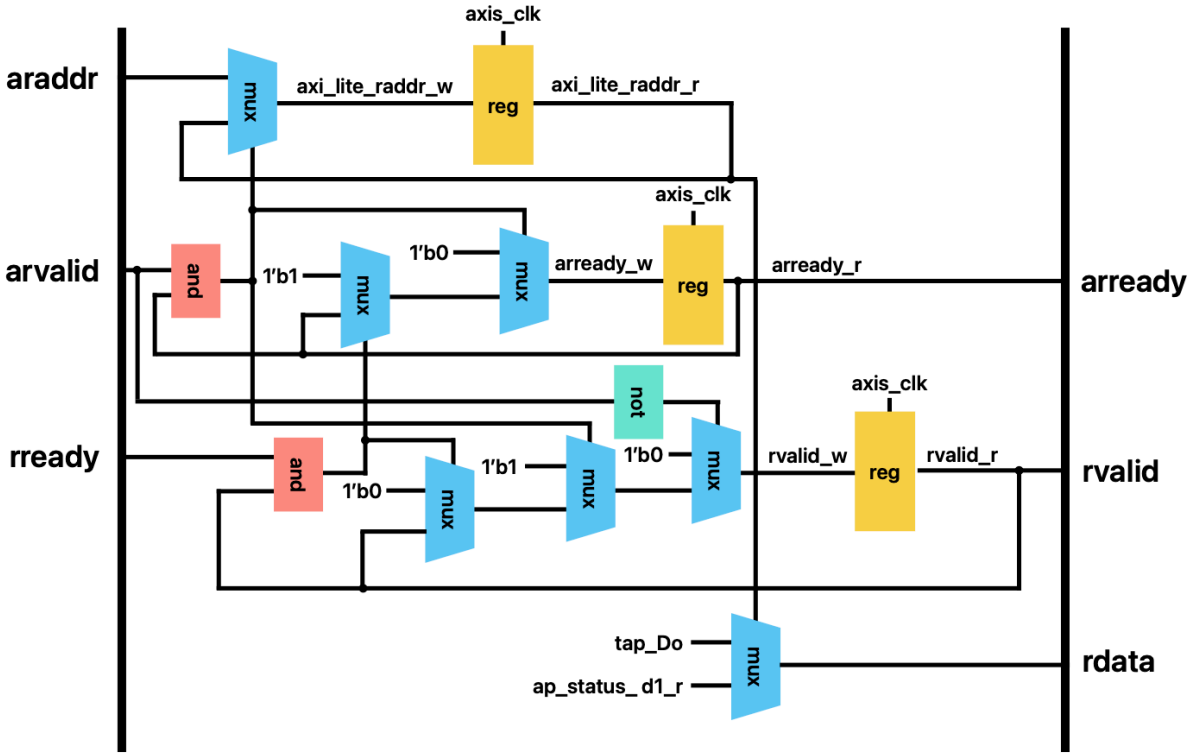
此report將會分為三部分，第一部分包含fir引擎中所有信號的設計與其相對應的block diagram，為求可讀性會將全部信號經分類後呈現，第二部分會依照時間順序說明各個執行階段的流程與波形圖，而第三部分將包含此設計的所有測試report截圖（rtl與synthesis），此三部分的聯集即包含lab3 submission guide中所提及在report中需繳交的內容。

一、信號設計與方塊圖

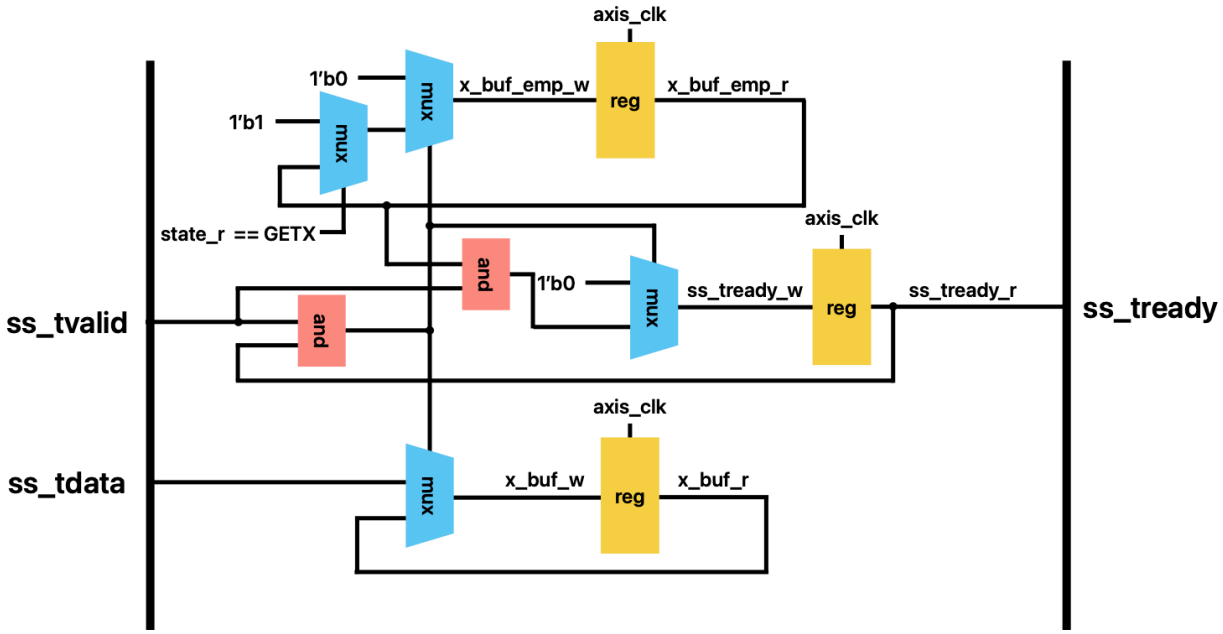
1. axi-lite寫入信號：
   - 信號設計：
     ① awready_r, awready_w：為儲存output awready之flip-flop，我們讓其在fir engine為idle時預設拉高為1以等待fir參數或ap_start地址的寫入，但若地址握手成功將歸零一週期以等待下一筆地址抵達。
     ② wready_r, wready_w：為儲存output wready之flip-flop，其特性與前者類似。
     ③ awflag_r, awflag_w, wflag_r, wflag_w：分別為儲存axi-lite地址與資料握手成功與否的flip-flop，由於axi-lite協議之特性，地址與資料未必將同時抵達，需暫存兩者握手成功與否的狀態，當兩個flag皆表示為1，則代表地址與資料皆到齊，可繼續進行寫入。
     ④ axi_lite_waddr_r, axi_lite_waddr_w：為儲存寫入地址的flip-flop，當地址握手成功時，其值將被更新為此筆新傳入的地址。
     ⑤ axi_lite_wdata_r, axi_lite_wdata_w：為儲存寫入資料的flip-flop，當資料握手成功時，其值將被更新為此筆新傳入的資料。
   - Block diagram：



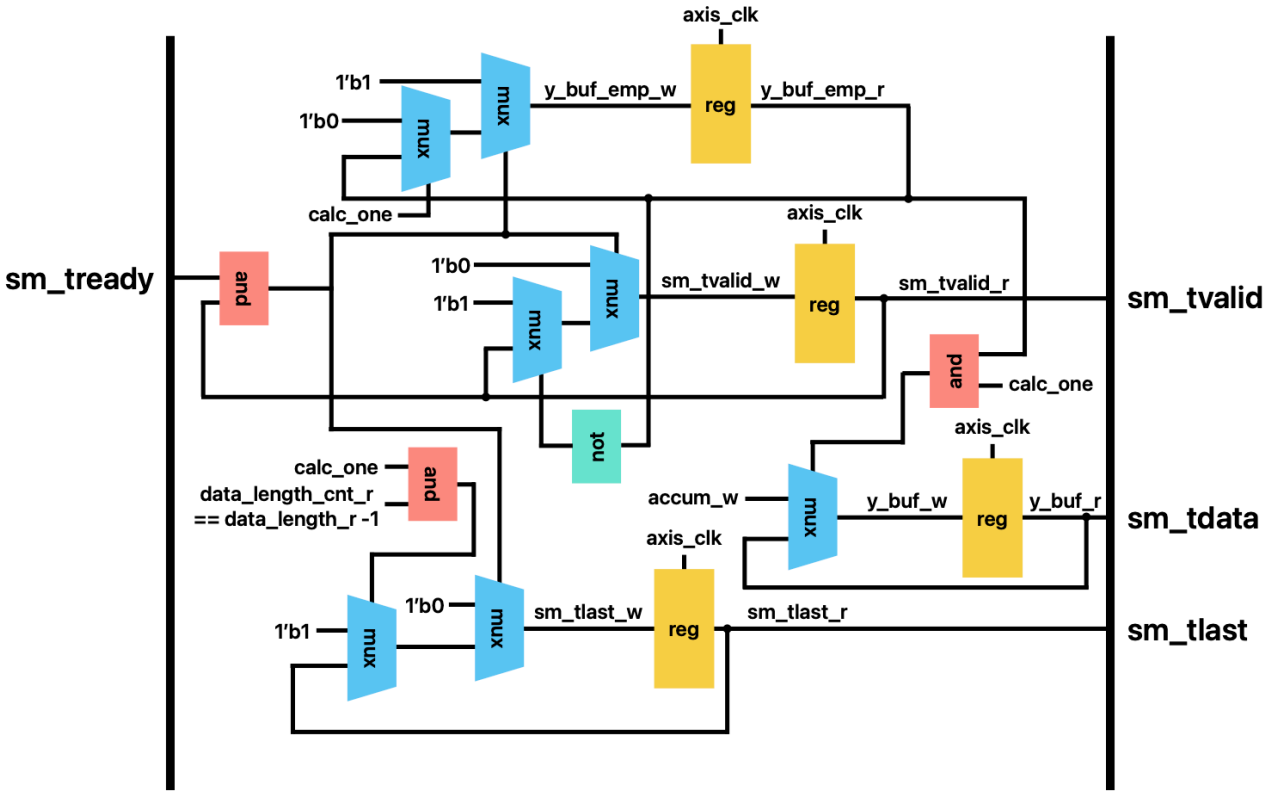
2. axi-lite讀出訊號：
   - 信號設計：
     ① arready_r, arready_w：為儲存output arready之flip-flop，預設拉高為1，當地址握手成功時將歸零，等待資料握手成功後拉回至1以接收下一筆讀出地址。
     ② axi_lite_raddr_r, axi_lite_raddr_w：為儲存讀出地址的flip-flop，當地址握手成功時，其值將被更新為此筆新傳入的地址。
     ③ rvalid_r, rvalid_w：為儲存output rvalid之flip-flop，由於無論要求讀出的是fir參數或系統狀態，皆保證能在下一週期進行輸出，故此訊號預設為0，當地址握手成功後一週期拉高為1。
     ④ rdata：為輸出之wire，由於所要求傳回的資料僅分兩種：fir參數或系統狀態，故在所儲存之讀出地址為0時被assign目前的系統狀態，而在其餘情況皆被assign tap sram的讀出資料tap_Do。
   - Block diagram：



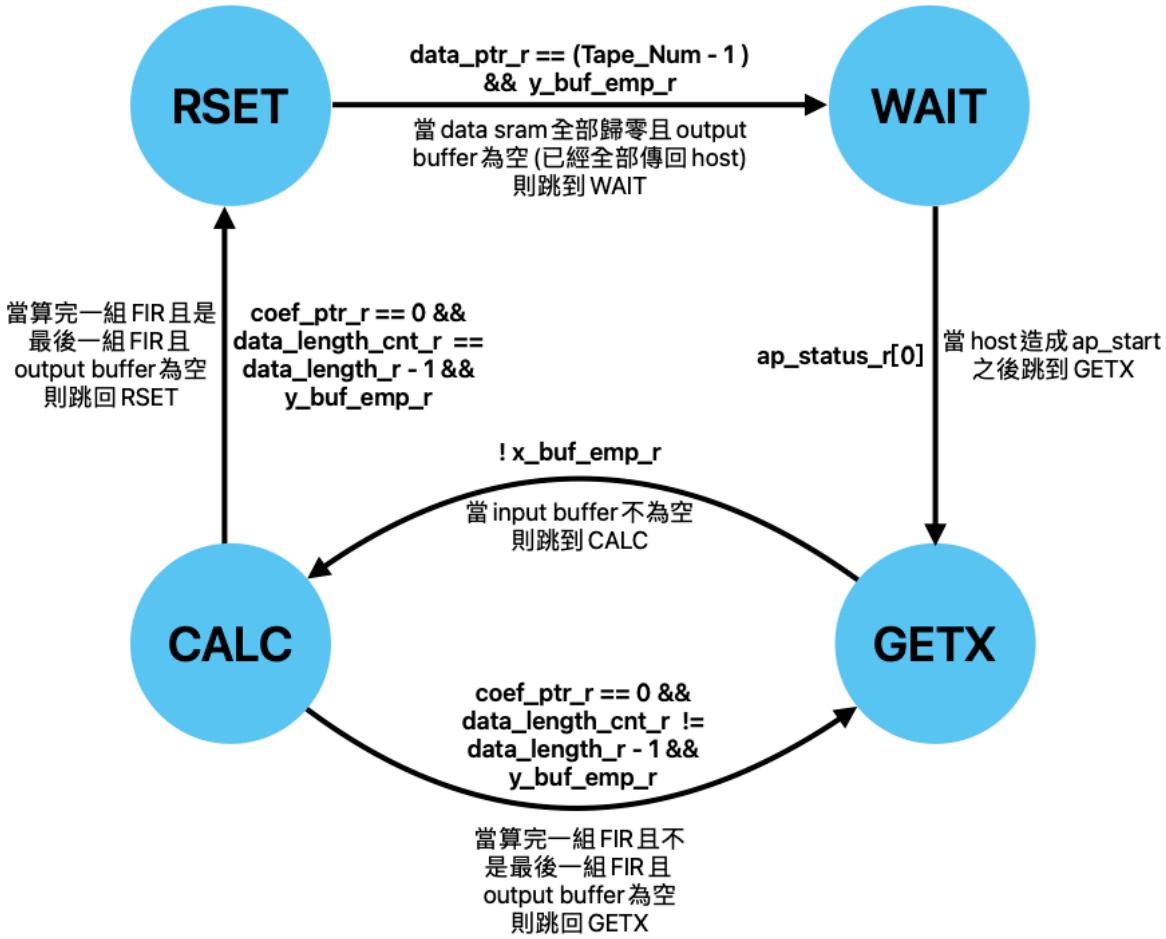
3. axi-stream寫入訊號：
   - 信號設計：
     ① ss_tready_r, ss_tready_w：為儲存output ss_tready之flip-flop，平時預設為0，當input buffer為空且ss_tvalid為1時拉高為1以接收訊號放進input buffer。
     ② x_buf_emp_r, x_buf_emp_w：為紀錄input buffer是否為空之flip-flop，當axi-stream握手成功時將有資料進入input buffer，故拉高為1，而當input buffer內的訊號被寫入data sram後input buffer被清空，故拉回0。
     ③ x_buf_r, x_buf_w：為充當input buffer的flip-flop，當axi-stream握手成功時其值將被更新為此筆新傳入的資料。
   - Block diagram：



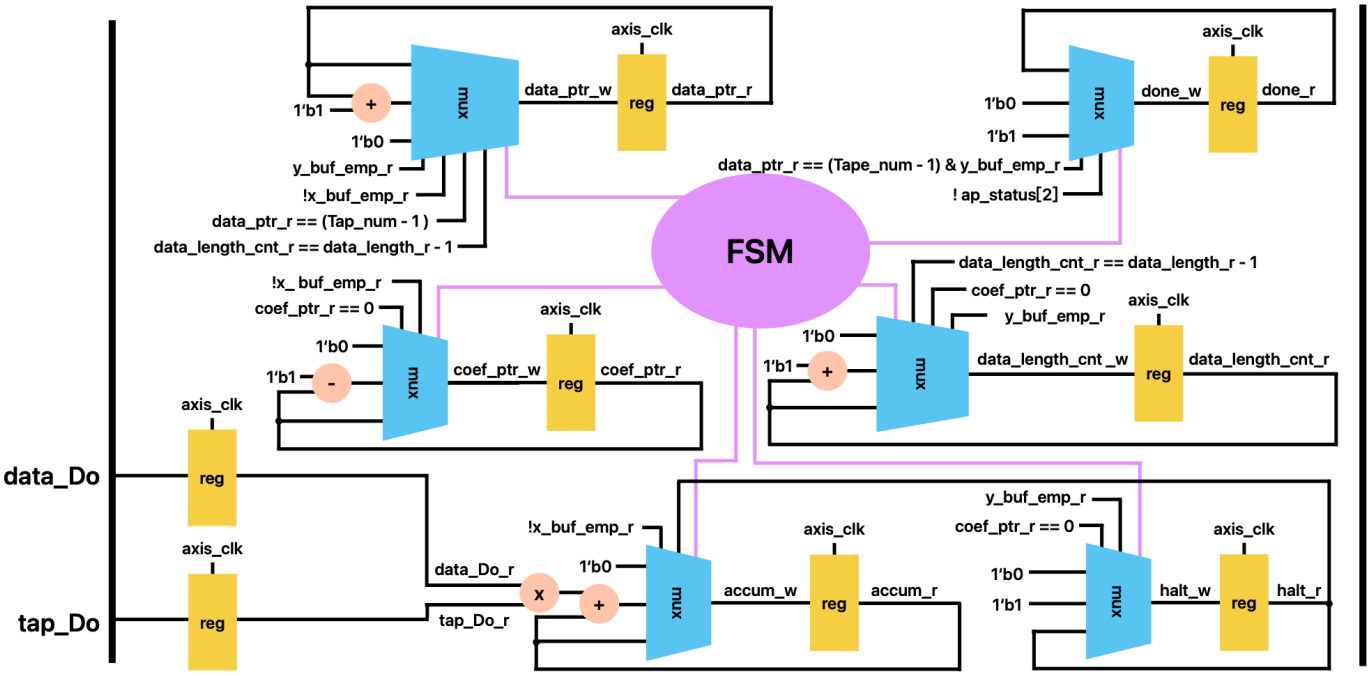
4. axi-stream讀出訊號：
   - 信號設計：
     ① sm_tvalid_r, sm_tvalid_w：為儲存output sm_tvalid之flip-flop，平時預設為0，當output buffer不為空時拉高為1以將output buffer內資料傳出。
     ② sm_tlast_r, sm_tlast_w：為儲存output sm_tlast之flip-flop，平時預設為0，當FSM算好一筆輸出且為最後一筆輸出時拉高為1。
     ③ y_buf_emp_r, y_buf_emp_w：為儲存output buffer是否為空之flip-flop，當axi-stream握手成功時拉至1，而當FSM算好一筆輸出時拉回0。
     ④ y_buf_r, y_buf_w：為充當output buffer的flip-flop，當output buffer為空且FSM算好一筆輸出時其值將被更新為此新算好的資料，因為axi-stream所傳出的資料必定來自此buffer，故y_buf_r直接與output信號sm_tdata對接。
   - Block diagram：



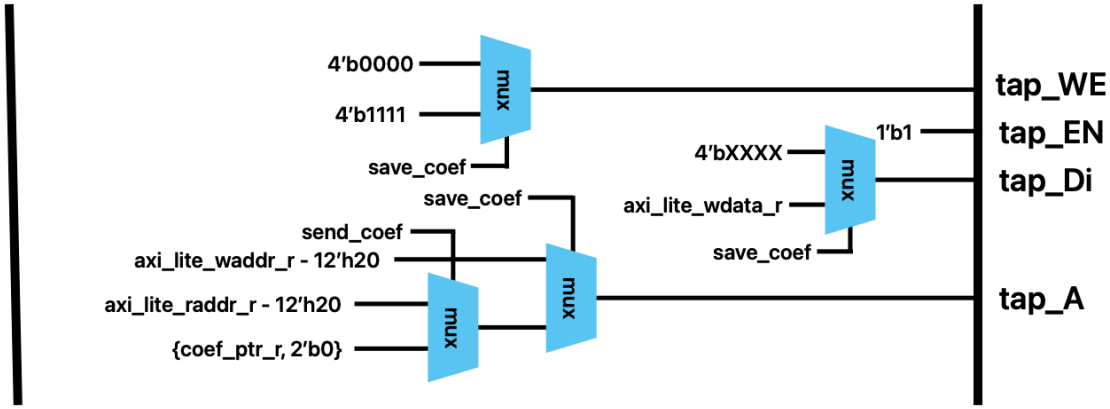
5. 運算FSM相關訊號：
   - 信號設計：
     ① state_r, state_w：為儲存FSM中state的flip-flop，總共分為4個state，第一個state為RSET，負責將data sram中儲存的資料歸零；第二個state為WAIT，負責等待host端傳遞ap_start以進行下一步運算；第三個state為GETX，負責將input buffer中的資料x存入data sram中；第四個state為CALC，負責在11個cycle內做完一組FIR運算，並將結果儲存於output buffer中，詳細見以下FSM圖與block diagram。
     ② data_ptr_r, data_ptr_w：為儲存data pointer的flip-flop，代表目前我們欲向data sram呼叫的地址，在RSET時，data pointer從0掃到10將data sram完全歸零，在GETX與CALC時，data pointer遞增以提供FIR運算所需要的data，在算完全部FIR運算後，data pointer歸零。
     ③ coef_ptr_r, coef_ptr_w：為儲存coefficient pointer的flip-flop，代表目前我們欲向tap sram呼叫的地址，在GETX與CALC時，coefficient pointer遞減以提供FIR運算所需要的fir參數，在算完一組FIR運算後，coefficient pointer歸零。
     ④ accum_r, accum_w：為儲存目前FIR運算累加結果的flip-flop，在CALC時，會與目前data sram與tap sram的輸出資料的積做累加，當算完一組FIR時歸零。
     ⑤ data_length_cnt_r, data_length_cnt_w：為儲存目前已計算的FIR組數的flip-flop，當算完一組FIR運算後加一，而當算完所有FIR運算後歸零。
     ⑥ done_r, done_w：為儲存目前該組FIR運算結束與否的flip-flop，在最後一組運算結果傳回且data sram被再度歸零後拉高為1一個週期。
   - Finite State Machine：



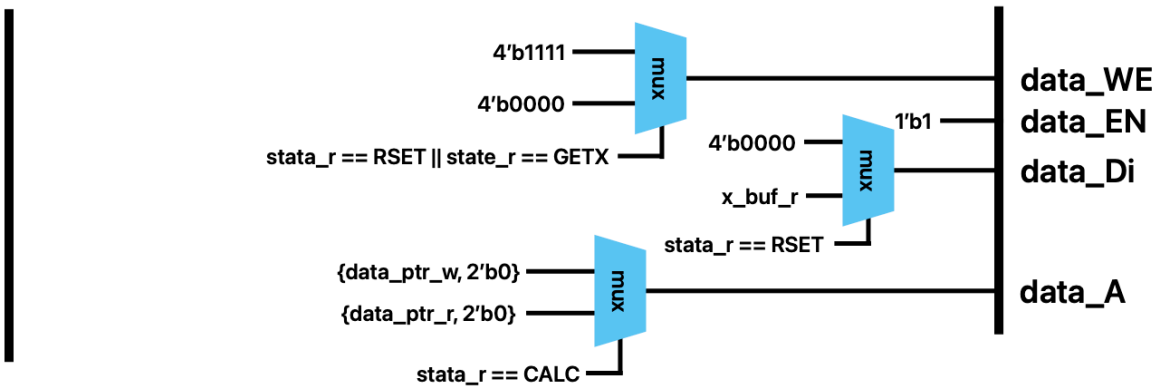
- Block diagram：



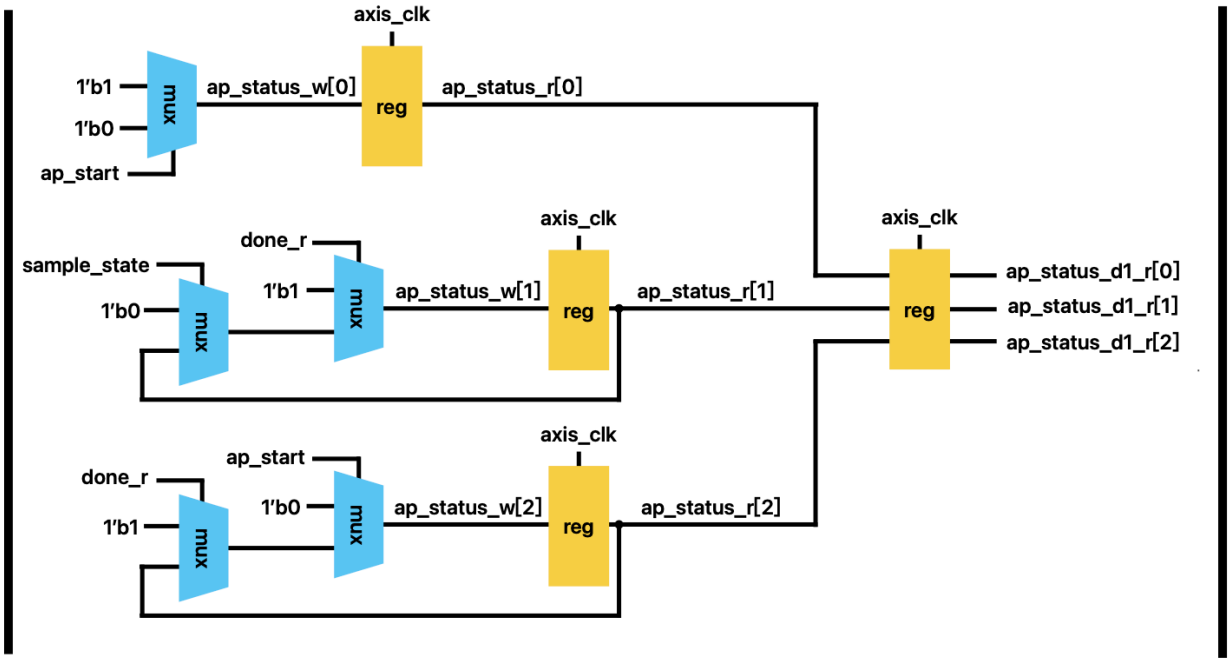
6. tap sram相關訊號：
   - 信號設計：
     ① tap_WE：為輸出之wire，平時預設為0，當要寫入coefficient時拉高為4'b1111。
     ② tap_EN：為輸出之wire，為了減少控制邏輯，永久設1。
     ③ tap_Di：為輸出之wire，平時預設為X，當要寫入fir參數時變為在axi-lite寫入時儲存的axi_lite_wdata_r。
     ④ tap_A：為輸出之wire，平時預設為coef_ptr_r往左shift 2 bit的值，當要寫入fir參數時變為在axi-lite寫入時儲存的axi_lite_waddr_r減去12'h20（offset），而當要讀出fir參數時變為在axi-lite讀出時儲存的axi_lite_raddr_w減去12'h20（offset）。
   - Block diagram：



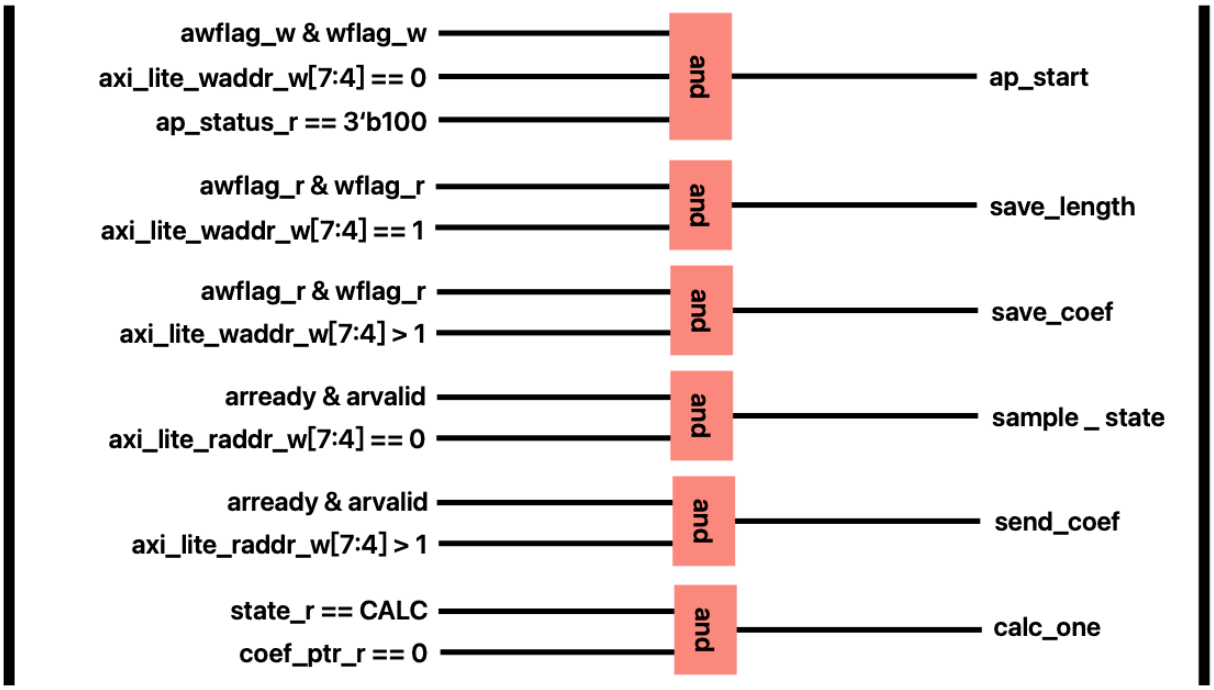
7. data sram相關訊號：
   - 信號設計：
     ① data_WE：為輸出之wire，平時預設為0，當要將data sram歸零或寫入data時拉高為4’b1111。
     ② data_EN：為輸出之wire，為了減少控制邏輯，永久設1。
     ③ data_Di：為輸出之wire，預設為x_buf_r（要將input buffer內的資料寫入data sram），當要將data sram歸零時變為0。
     ④ data_A：為輸出之wire，根據目前的state設為data_ptr_r或data_ptr_w往左shift 2 bit的值。
   - Block diagram：



8. ap_state相關訊號：
   - 信號設計：
     ① ap_state_r[0], ap_state_w[0]：為儲存ap_start狀態的flip-flop，只有當host端寫入ap_start後被拉高為1一個週期，其餘時候均拉回0。
     ② ap_state_r[1], ap_state_w[1]：為儲存ap_done狀態的flip-flop，在每次fir運算結束後被拉高為1，且在host端訪問一次其狀態後拉回0。
     ③ ap_state_r[2], ap_state_w[2]：為儲存ap_idle狀態的flip-flop，在host端寫入ap_start後拉至0，在fir運算結束後拉高回1。
     ④ ap_state_d1_r, ap_state_d1_w：為儲存以上三個狀態delay一個cycle的flip-flop。
   - Block diagram：

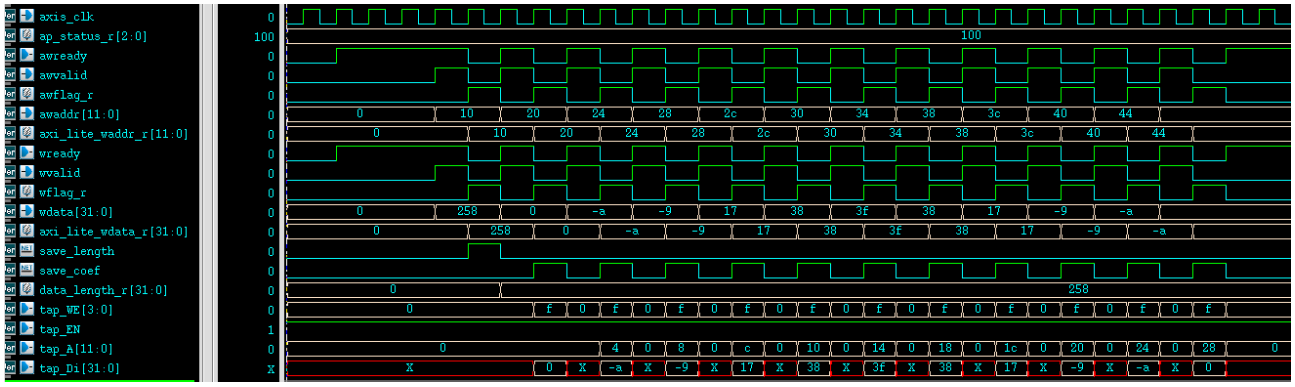


9. 其餘控制訊號：
   - 信號設計：
     ① ap_start：為wire，代表host端傳進一筆ap_start，故在axi-lite寫入成功且寫入地址為0且ap_state_r為3'b100(idle)時assign其為1。
     ② save_length：為wire，代表host端傳入一筆data length，故在axi-lite寫入成功且寫入地址為1開頭時assign其為1。
     ③ save_coef：為wire，代表host端傳入一筆fir參數，故在axi-lite寫入成功且寫入地址為2開頭以上時assign其為1。
     ④ sample_state：為wire，代表host端要求回傳目前狀態，故在axi-lite讀出地址握手成功且讀出地址為0時assign其為1。
     ⑤ send_coef：為wire，代表host端要求傳回fir參數，故在axi-lite讀出地址握手成功且讀出地址為2開頭以上時assign其為1。
     ⑥ calc_one：為wire，代表fir engine剛算好一筆fir輸出，故在FSM為CALC且coef_ptr_r == 0時assign其為1。
   - Block diagram：

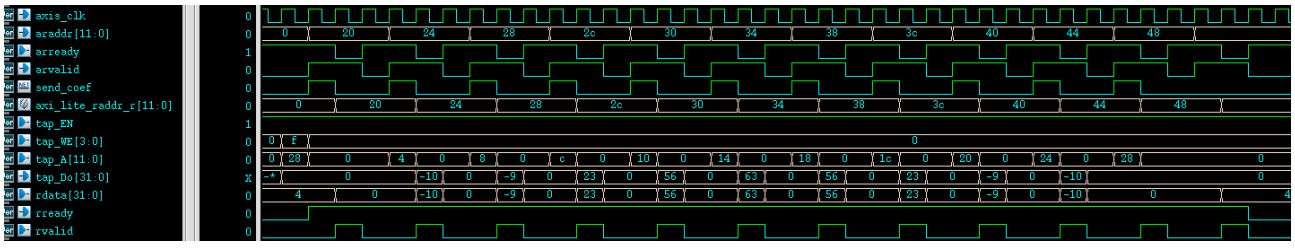


二、執行流程與波形圖

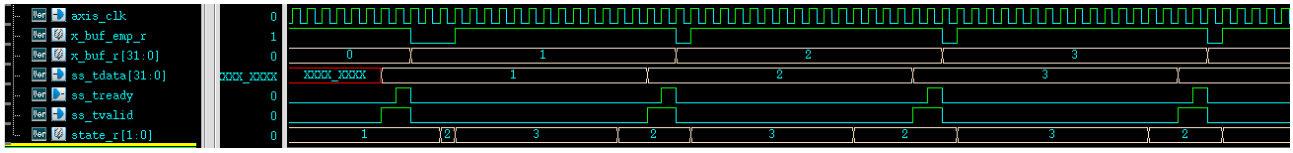
1. axi-lite寫入fir參數並存入tap sram：
   - 流程說明：
     ① 當目前狀態是idle時(ap_status_r[2] == 1)，awready與wready會被拉高成以接收寫入地址與人寫入資料。
     ② 在寫入地址握手成功後(awready & awvalid == 1)，awflag_r會被拉高為1一個週期，並且輸入的寫入地址(awaddr)會被存入暫存器(axi_lite_waddr_r)，同理，在寫入資料握手成功後(wready & wvalid == 1)，wflag_r會被拉高為1一個週期，輸入的寫入資料(wdata)會被存入暫存器(axi_lite_wdata_r)。
     ③ 當awflag_r與wflag_r同時為1，代表寫入條件到齊，若寫入地址不小於12'h20，則代表寫入的資料為fir參數，此時將tap_WE拉高，tap_A設為暫存的寫入地址(axi_lite_waddr_r)減去12'h20，tap_Di設為暫存的寫入資料(axi_lite_wdata_r)進行寫入。
     ④ awready與wready在每次握手成功後會被拉回0一個週期，但只要目前狀態仍是idle(ap_status_r[2] == 1)，則會持續觸發步驟①直到存入所有fir參數。
   - 波形圖：



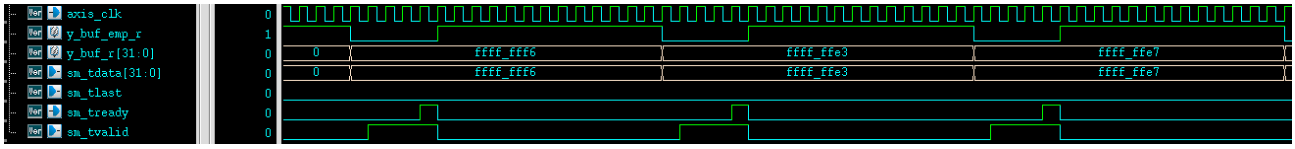
2. axi-lite讀出fir參數：
   - 流程說明：
     ① arready預設為1以接收讀出地址。
     ② 讀出地址握手成功後(arready & arvalid == 1)，將讀出地址存入暫存器(axi_lite_raddr_r)，arready此時被拉至0以避免接收下一筆讀出地址。
     ③ 若讀出地址不小於12'h20，則代表讀出的資料為fir參數，故在握手成功的同時將讀出地址減去12'h20的值傳給tap_A，用以在下一個週期獲得需要的參數。
     ④ 在讀出地址握手成功的下一個週期令rvalid為1，rdata為步驟③從tap sram取出的參數，等待rvalid與rready握手成功即可將此筆參數送回host。
     ⑤ 讀出資料握手成功後(rready & rvalid == 1)，arready被拉回1以接收下一筆讀出地址。
   - 波形圖：



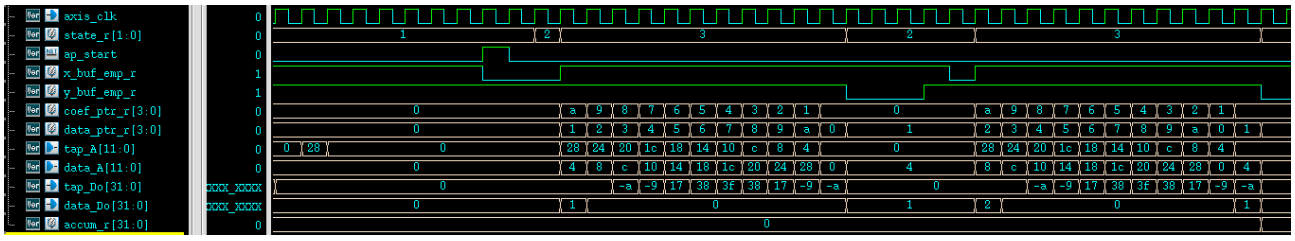
3. axi-stream將x寫入buffer：
   - 流程說明：
     ① 當input buffer為空且ss_valid為1時ss_tready會拉高以接收新的資料放入input buffer。
     ② 在資料握手(ss_tready & ss_tvalid == 1)後輸入資料被存入input buffer中，此時input buffer不再為空。
     ③ 等待FSM將input buffer中資料取出後(state_r == GETX)，input buffer再度為空，回到步驟①。
   - 波形圖：



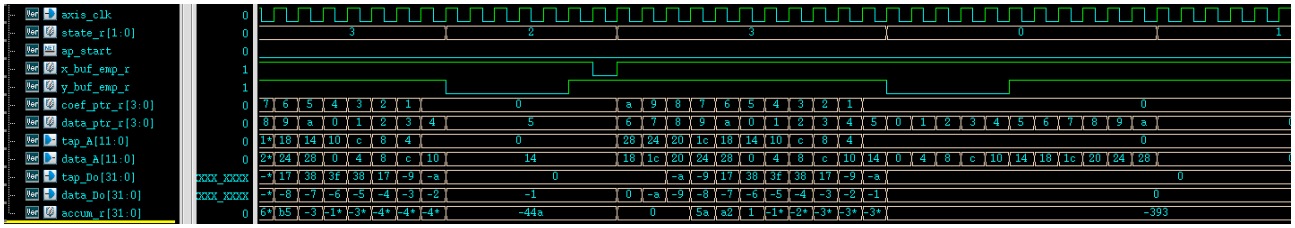
4. axi-stream將y傳回host：
   - 流程說明：
     ① 當output buffer不為空時將sm_valid拉高以將資料傳出。
     ② 當是最後一筆資料時sm_tlast會預先拉高，且在資料握手成功後拉回0。
   - 波形圖：



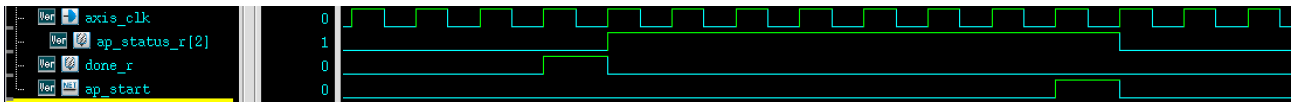
5. 利用FSM進行fir運算：
   - 流程說明：
     ① 在RSET state時，用11個週期將data sram內的值進行歸零，當output buffer為空（上一筆fir result傳回host）後進入WAIT state以等待下一筆ap_start。
     ② 在WAIT state時，等待host端寫入ap_start後進入GETX state並開始進行fir運算。
     ③ 在GETX state時，等待input buffer不為空時，從input buffer中取出資料存進目前data_ptr_r指向的data sram位置，並進入CALC state。
     ④ 在CALC state時，用11個週期將data sram與tap sram傳出之值進行相乘累加，當output buffer為空（上一筆fir result傳回host）後將累加值存入output buffer，再依照全部fir運算結束與否回到RSET state或GETX state。
   - 波形圖：
     ① 全部fir運算剛開始(600筆中的第1, 2筆)



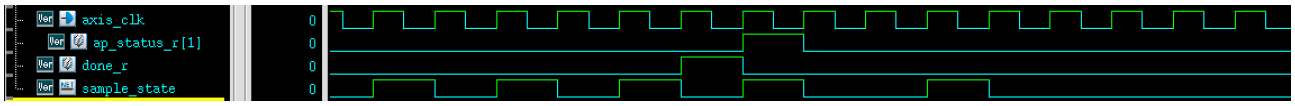
     ② 全部fir運算將結束(600筆中的第599, 600筆)



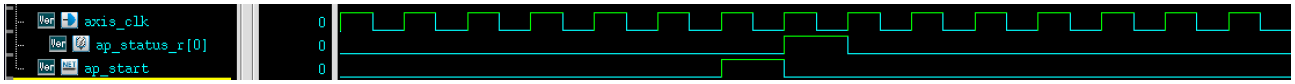
6. ap_idle, ap_done, ap_start：
   - 流程說明：
     ① 此三者的set與reset條件詳見第一部分ap_state相關訊號的設計。
   - 波形圖：
     ① ap_idle (下圖為先被set再被reset的過程)



     ② ap_done (下圖為先被set再被reset的過程)



     ③ ap_start (下圖為先被set再被reset的過程)



三、測試截圖

1. 資源用量：
   - Adders / Registers / Multipliers / Muxes / LUTs / BRAMs：

LUT	FF	BRAM	URAM	DSP
263	265	0	0	3

```
Detailed RTL Component Info :
+---Adders :
      2 Input   32 Bit       Adders := 1
      2 Input   12 Bit       Adders := 2
      2 Input   10 Bit       Adders := 2
      2 Input    4 Bit       Adders := 2
+---Registers :
                32 Bit    Registers := 6
                12 Bit    Registers := 2
                10 Bit    Registers := 2
                 4 Bit    Registers := 2
                 3 Bit    Registers := 2
                 1 Bit    Registers := 13
+---Multipliers :
              32x32  Multipliers := 1
+---Muxes :
      2 Input   32 Bit        Muxes := 5
      4 Input   32 Bit        Muxes := 1
      2 Input   12 Bit        Muxes := 7
      2 Input   10 Bit        Muxes := 1
      2 Input    6 Bit        Muxes := 1
      2 Input    4 Bit        Muxes := 9
      4 Input    4 Bit        Muxes := 2
      4 Input    2 Bit        Muxes := 1
      2 Input    2 Bit        Muxes := 1
      2 Input    1 Bit        Muxes := 16
      4 Input    1 Bit        Muxes := 4
```

2. 合成截圖：
   - 合成cycle：12 (ns)
   - Design timing summary：

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.174 ns	Worst Hold Slack (WHS):	0.137 ns	Worst Pulse Width Slack (WPWS):	6.000 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:	814	Total Number of Endpoints:	814	Total Number of Endpoints:	266

All user specified timing constraints are met.

   - Critical path：

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	0.174	7	8	18	data_length_r_reg[1]/C	data_A[3]	8.334	4.436	3.898	13.0	axis_clk	axis_clk
Path 2	0.174	7	8	18	data_length_r_reg[1]/C	data_A[5]	8.334	4.436	3.898	13.0	axis_clk	axis_clk
Path 3	0.758	6	7	18	data_length_r_reg[1]/C	data_A[2]	7.750	4.312	3.438	13.0	axis_clk	axis_clk
Path 4	0.758	6	7	18	data_length_r_reg[1]/C	data_A[4]	7.750	4.312	3.438	13.0	axis_clk	axis_clk
Path 5	1.331	10	11	2	tap_Do_r_reg[16]/C	accum_r_reg[31]/D	11.532	8.633	2.899	13.0	axis_clk	axis_clk
Path 6	1.331	10	11	2	tap_Do_r_reg[16]/C	y_buf_r_reg[31]/D	11.532	8.633	2.899	13.0	axis_clk	axis_clk

3. 運作時長：
    - 如下圖，從第一筆ap_start輸入(ap_status_r[0]變為1)至第一筆ap_done被讀取(ap_status_r[1]變為1)，總共耗時(260226000 – 130506000)ps，也就是129720ns，因為我們的clock cycle為12ns，故一組FIR運算的花費時長為10810cycles。