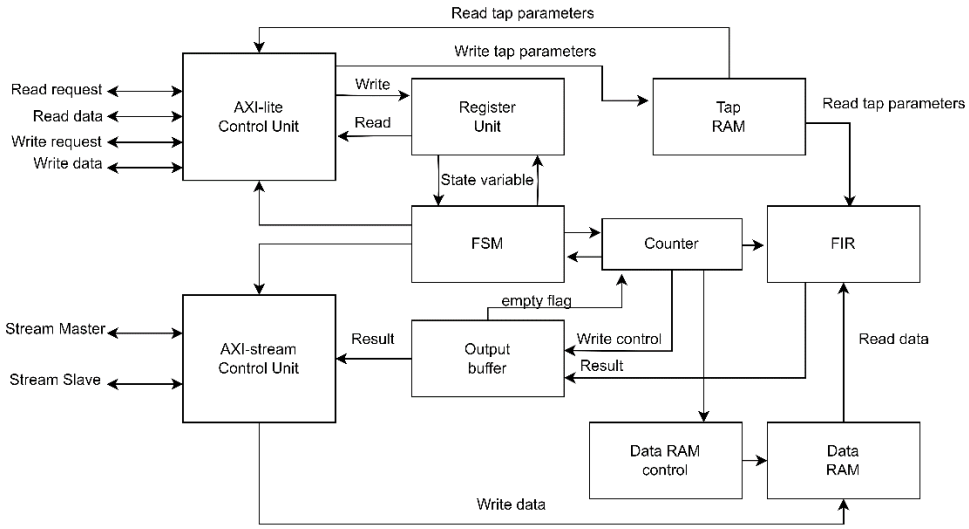
SOC lab3_report r11921073 李丞峰

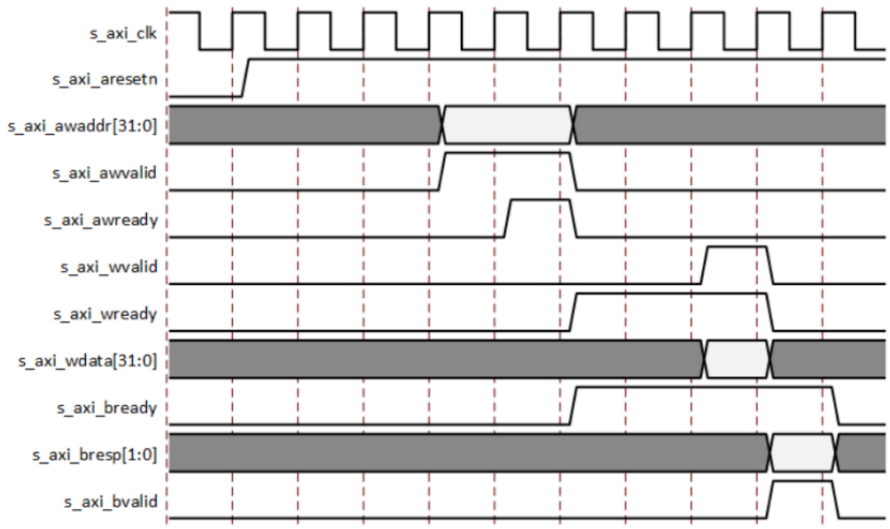
1. Block Diagram



2. Describe operation

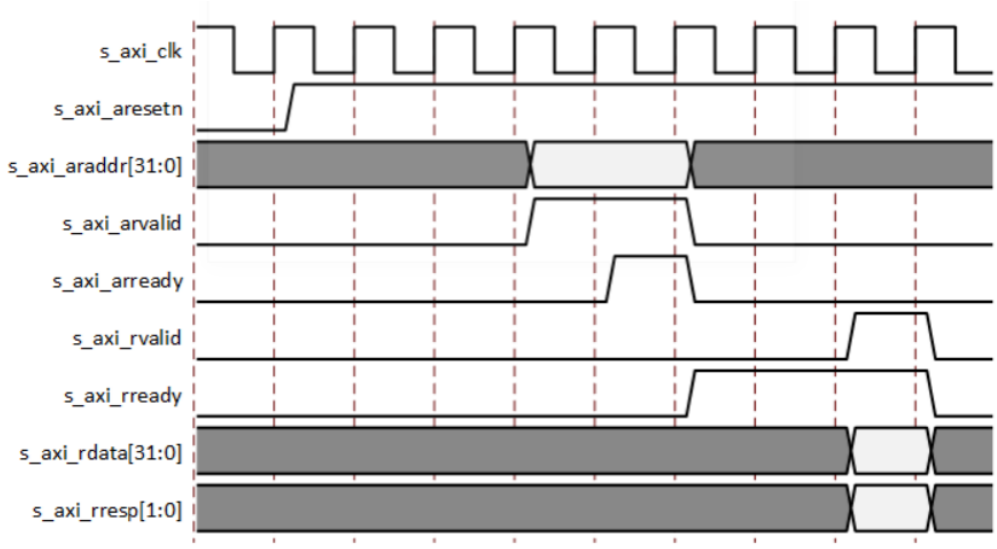
- **AXI-lite**

a. Write request and Write data channel



在實作 Write Channel 上，我在邏輯上做了點調整，將 write ready 和 awrite ready 的控制分離，兩者的邏輯相同處是在於只要沒發生 handshake，例如 write request channel 的 handshake 尚未發生，那麼就會允許 ready 訊號拉起，如此一來就可以達到 request channel 和 data channel 的獨立性，等到兩者的 handshake 都發生後才會將 address 和 data 保存於 IP 之中，之後再視情況是對於 ap register 做寫入或是輸入 data length 和 tap ram 數值。

b. Read request and Read data channel



在實作 Read Channel 上，read request channel 的邏輯與 write channel 相似，都是判斷 handshake 是否成立來決定 ready 訊號的拉起，不過因為 AXI literead channel 的 data 與 valid 會固定在 read request channel 的 handshake 發生後才會進行輸出，因此若 read request channel 的 handshake 未發生，read channel 的 valid 就不會被拉高，藉此來保證控制的正確性，另外，read data 的選擇也是會根據 read address 的數值，去選擇對應的結果輸出。
等到 read data channel 的 handshake 發生後，再重置兩個通道的 handshake 狀態來迎接下一個指令輸入。

- **AXI-stream**

在這次的 lab 中，AXI-stream 是用於接收計算資料以及輸出計算結果，因此在設計上我是利用 counter 決定是否將接收端的 ready 訊號拉高，例如在 0 的時候拉高 ready 來準備接收資料，如果 handshake 發生時，就會允許 counter 往上計數，來控制後續的 bram 以及 fir 來做計算，等到最終計算至 11 時，就將計算結果存儲至 output buffer 之中，因此若 output buffer 有值，就會設定 sm channel 的 valid 以及 data 輸出，直到 hand shake 成立後，就會記錄 output buffer 為 empty，就能允許下一筆資料放入等待 sm 通道傳送。

- **FSM**

我的狀態機具有 4 種狀態，idle、start、complete、done

a. Idle : 此時可以輸入 tap data 以及一些 data length 參數來做設定，直到 ap_start 被拉高後進入 start 狀態。
b. Start : 一進入此狀態時，會先將 data ram 中的所有數值清空為 0，之後才會開始配合 counter 進行計算，例如 counter 為 0 時會接收 stream

通道的數據進入 FIR 運算，中間也會不斷的與 tap ram 做讀取以及 data ram 做讀寫，最終 counter 為 11 且 output buffer 為 empty 狀態就會將結果寫入 output buffer，否則就會將 counter 維持於 11 直到 buffer 狀態改變成 empty，之後將用於計算次數的 counter + 1，直到最後一筆 data 計算完畢就會進入 complete 狀態。

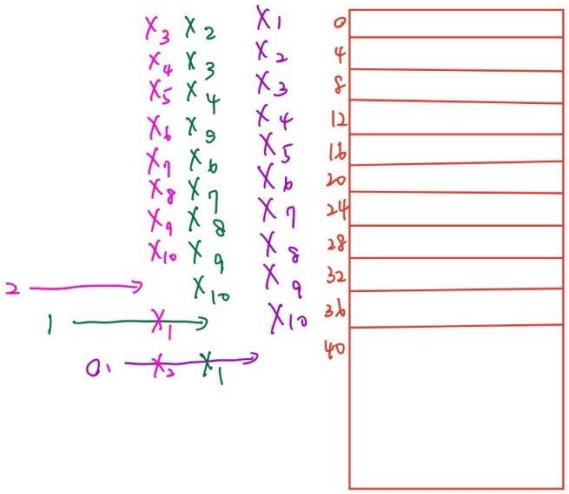
c. Complete : 進入此狀態代表 fir 已經完成所有的計算，但是最後一筆 data 尚未被取走，因此若 handshake 發生後，就會離開此狀態進入 done。

d. Done : 此狀態代表 fir 已經完成所有計算，且所有資料都被讀取，因此下一個 cycle 就會回復到 idle 狀態等待外部控制命令輸入。

- **FIR**

  根據數學運算式子以及作業的限制，因此使用了一個乘法器以及一個加法器，透過控制乘法器的輸入來使每一個 cycle 只進行一次的乘法以及與結果相加，直到 counter 為 11 時就會將結果放入 output buffer。

- **Address Control**



這個部份我是實作了兩個 pointer，分別是 write pointer 和 read pointer，write pointer 主要會根據 counter 變動，初始階段會固定在 40，每一次 counter 來到 11 完成一次完整的 fir 運算後，write pointer 就會減 4，而 read pointer 會根據 write pointer 和 counter 當前的數值，決定要讀取 data ram 和 tap ram 的某一筆資料來送入 fir 之中進行運算。所以若是第一次的 fir 運算，write pointer 就會固定在 40，而 read pointer 就會由 40 再往下，也就是回到頂部的 0 開始依序讀取所需數值，從新資料依序讀取到舊資料，最終 counter 來到 11 就會將 ss channel 的輸入寫入 write pointer 處，然後 write poiner 就會減 4 來到 36 來繼續下一次的 fir 計算。

而這個控制方式有助於未來的擴展性上，因為若是 bram 的 size 過大，會

不利於讀寫控制，因此選用 pointer 來協助控制即可達到更多的通用性和延展性。

- **Ap_register 與 data length**

  這個部分我是分別使用 3 個 flip-flop 和 32 個 flip-flop 來實現。只有 ap_start 和 data length 可以做讀寫，其他的 ap_done 和 ap_idle 只允許被外部讀取。

  a. ap_start :
     如果目前 FSM 的狀態是 idle，才會允許寫入 ap_start，且寫入後若進入了 start state，就會自動的設置為 0。

  b. ap_done :
     如果 FSM 的 state 進入了 done，代表 fir 所有的計算已經完成，且所有 data 都被送出，因此就會被設置為 1，之後如果被讀取，或是進入了 start state，ap_done 就會被設置為 0。

  c. ap_idle :
     如果系統被 reset，ap_idle 就會被設定為 1，若離開了 idle 狀態而來到 start 開始 fir 運算，就會被設置為 0。
     如果 fir 計算完畢最後一筆 data，也就是用來計算 data length 的 counter 為 599 且控制 fir 的 counter 來到 11，就會將 ap_idle 設定為 1 來表示目前 fir 已經完成所有運算，處於閒置的狀態，除非 ap_idle 被讀取，或是回到 idle 狀態後 ap_start 被設置為 1 準備進入 start start，此時就會將 ap_idle 設置為 0。

  d. data length :
     這個 register 負責記錄本次要運算的數據資料量，可以被 AXI-lite 的 channel 讀寫。

- **Output buffer**

  在 axi-stream 的輸出通道，我實作了一個 output buffer 來承接計算結果，因此若外部 sm_ready 來不及拉起，而是在 11 個 cycle 之內才拉起時，也不會影響我的 fir 進行下一筆資料的計算，除非超過了 11 個 cycle，sm_ready 尚未被拉起，代表前一筆資料尚未被取走，但新的資料以計算完畢等待輸出，此時才會將 counter 給凍結，直到 buffer 狀態為 empty 後才會寫入新數值進入 output buffer，並且將 counter 歸零進行下一筆計算。

- **Testbench**

  在 testbench 總共會經歷兩次的測試，首先會檢查 ap_idle = 1，之後開始使用 AXI-lite write 進行 tap 數據傳輸，再傳輸完畢 11 筆數據後

開始 AXI-lite read 來檢測寫入 tap ram 的數據，檢測完畢後 module 開始將 ss_ready 拉 high 來接收 AXI_stream slave 的 data，在傳輸的同時，testbench 也透過 AXI-stream master 來接收與比對 fir 的計算結果，直到 ss channel 送出倒數第二筆 input data 後，會利用 AXI-lite 檢測 ap_idle 和 ap_done 是否為 0，接下來才送出最後一筆 input data，然後 sm channel 在接收了 599 筆計算輸出後，也會馬上透過 AXI-lite 檢測 ap_idle 和 ap_done 同時為 0，因為這時候 fir 仍然在計算最後一筆數據，最後接收完畢最後一筆 fir 輸出後，再使用 AXI-lite 檢測 ap_idle 和 ap_done 同時為 1，到此結束第一次的模擬。

之後第二次的模擬幾乎與第一次相同，差別只在於直到 ss channel 送出倒數第二筆 input data 後，會利用 AXI-lite 檢測 ap_idle 和 ap_done 是否為 0，再來 sm channel 讀取倒數第二筆數據，再次利用 AXI-lite 檢測 ap_idle 和 ap_done 是否為 0，接下來會延遲 15 個 cycle 後發起 AXI-lite 檢測 ap_idle 是否為 1，因為理論上在經過這麼多的延遲後 fir 已經完成最後一筆數據的計算，應該會將 ap_idle 設定為 1 且等待最後一筆的結果被 sm channel 給讀取，所以最後 sm channel 讀走最後一筆資料時，就會 AXI-lite 檢測 ap_idle 是否為 0 且 ap_done 為 1，因為 ap_idle 被讀取，所以應該被設定為 0，而且因為數據已經全部發送完畢，ap_done 應該被設定為 1。

第一次模擬 ：

```
-----------Start simulation-----------
----Start the data input(AXI-Stream)----
Check ap start, done and idle
OK: exp =           4, rdata =           4
----Start the coefficient input(AXI-lite)----
 Check Coefficient ...
OK: exp =           0, rdata =           0
OK: exp =         -10, rdata =         -10
OK: exp =          -9, rdata =          -9
OK: exp =          23, rdata =          23
OK: exp =          56, rdata =          56
OK: exp =          63, rdata =          63
OK: exp =          56, rdata =          56
OK: exp =          23, rdata =          23
OK: exp =          -9, rdata =          -9
OK: exp =         -10, rdata =         -10
OK: exp =           0, rdata =           0
 Tape programming done ...
 Start FIR
----End the coefficient input(AXI-lite)----
Set ap_start, total cycle =          96
[PASS] [Pattern         0] Golden answer:           0, Your answer:           0
[PASS] [Pattern         1] Golden answer:         -10, Your answer:         -10
[PASS] [Pattern         2] Golden answer:         -29, Your answer:         -29
[PASS] [Pattern         3] Golden answer:         -25, Your answer:         -25
[PASS] [Pattern       584] Golden answer:       -3660, Your answer:       -3660
[PASS] [Pattern       585] Golden answer:       -3477, Your answer:       -3477
[PASS] [Pattern       586] Golden answer:       -3294, Your answer:       -3294
[PASS] [Pattern       587] Golden answer:       -3111, Your answer:       -3111
[PASS] [Pattern       588] Golden answer:       -2928, Your answer:       -2928
[PASS] [Pattern       589] Golden answer:       -2745, Your answer:       -2745
[PASS] [Pattern       590] Golden answer:       -2562, Your answer:       -2562
[PASS] [Pattern       591] Golden answer:       -2379, Your answer:       -2379
[PASS] [Pattern       592] Golden answer:       -2196, Your answer:       -2196
[PASS] [Pattern       593] Golden answer:       -2013, Your answer:       -2013
[PASS] [Pattern       594] Golden answer:       -1830, Your answer:       -1830
[PASS] [Pattern       595] Golden answer:       -1647, Your answer:       -1647
[PASS] [Pattern       596] Golden answer:       -1464, Your answer:       -1464
[PASS] [Pattern       597] Golden answer:       -1281, Your answer:       -1281
OK: exp =           0, rdata =           0
[PASS] [Pattern       598] Golden answer:       -1098, Your answer:       -1098
------End the data input(AXI-Stream)------
OK: exp =           0, rdata =           0
[PASS] [Pattern       599] Golden answer:        -915, Your answer:        -915
OK: exp =           6, rdata =           6
-------------------------------------------
----------Congratulations! Pass------------
End of first Process, total cycle =        7315
```

第二次模擬 ：

```
----------Congratulations! Pass------------
End of first Process, total cycle =       7315


-----------Start next simulation----------
----Start the data input(AXI-Stream)----
----Start the coefficient input(AXI-lite)----
 Check Coefficient ...
OK: exp =          0, rdata =          0
OK: exp =        -10, rdata =        -10
OK: exp =         -9, rdata =         -9
OK: exp =         23, rdata =         23
OK: exp =         56, rdata =         56
OK: exp =         63, rdata =         63
OK: exp =         56, rdata =         56
OK: exp =         23, rdata =         23
OK: exp =         -9, rdata =         -9
OK: exp =        -10, rdata =        -10
OK: exp =          0, rdata =          0
 Tape programming done ...
 Start FIR
----End the coefficient input(AXI-lite)----
Set ap_start, total cycle =       7407
[PASS] [Pattern          0] Golden answer:          0, Your answer:          0
[PASS] [Pattern          1] Golden answer:        -10, Your answer:        -10
```

```
[PASS] [Pattern        587] Golden answer:      -3111, Your answer:      -3111
[PASS] [Pattern        588] Golden answer:      -2928, Your answer:      -2928
[PASS] [Pattern        589] Golden answer:      -2745, Your answer:      -2745
[PASS] [Pattern        590] Golden answer:      -2562, Your answer:      -2562
[PASS] [Pattern        591] Golden answer:      -2379, Your answer:      -2379
[PASS] [Pattern        592] Golden answer:      -2196, Your answer:      -2196
[PASS] [Pattern        593] Golden answer:      -2013, Your answer:      -2013
[PASS] [Pattern        594] Golden answer:      -1830, Your answer:      -1830
[PASS] [Pattern        595] Golden answer:      -1647, Your answer:      -1647
[PASS] [Pattern        596] Golden answer:      -1464, Your answer:      -1464
[PASS] [Pattern        597] Golden answer:      -1281, Your answer:      -1281
OK: exp =          0, rdata =          0
[PASS] [Pattern        598] Golden answer:      -1098, Your answer:      -1098
------End the data input(AXI-Stream)------
-------------------------------------------
OK: exp =          0, rdata =          0
OK: exp =          4, rdata =          4
[PASS] [Pattern        599] Golden answer:       -915, Your answer:       -915
OK: exp =          2, rdata =          2
-------------------------------------------
----------Congratulations! Pass------------
End of  total Process, total cycle =      14639
```

第一次的模擬，fir 消化 600 筆數據所需要花費的時間為：7315-96=7219 個 clock cycle，而第二次的模擬，fir 運算時長總共為 ：7232 個 clock cycle，原因為第二次的模擬添加了一些 clock delay 來協助測試 ap_idle、ap_done。

3. Resource usage

Resource	Utilization	Available	Utilization %
LUT	350	53200	0.66
FF	204	106400	0.19
DSP	3	220	1.36
IO	329	125	263.20

4. Timing Report

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.420 ns	Worst Hold Slack (WHS):	0.142 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:	298	Total Number of Endpoints:	298	Total Number of Endpoints:	205

All user specified timing constraints are met.

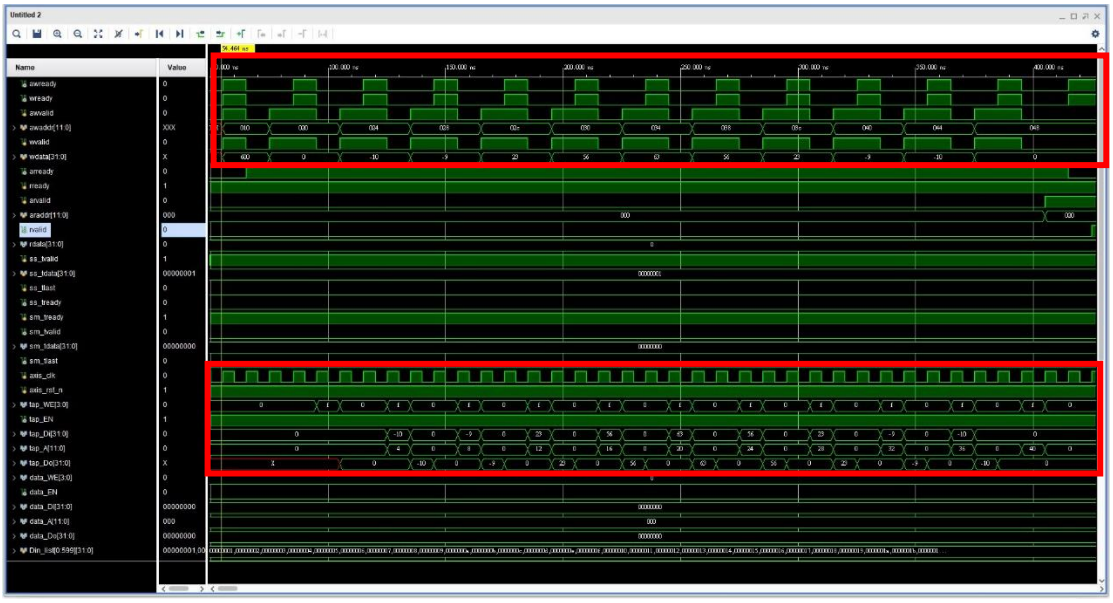
Project Summary | Device | fir_tb.v | constraints.xdc | **Path 1 - timing_1**

Summary

Name	Path 1
Slack	0.420ns
Source	SS_data_reg[16]/C (rising edge-triggered cell FDCE clocked by axis_clk {rise@0.000ns fall@6.500ns period=13.000ns})
Destination	cal_out_reg[31]/D (rising edge-triggered cell FDCE clocked by axis_clk {rise@0.000ns fall@6.500ns period=13.000ns})
Path Group	axis_clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	13.000ns (axis_clk rise@13.000ns - axis_clk rise@0.000ns)
Data Path Delay	12.443ns (logic 8.568ns (68.856%) route 3.875ns (31.144%))
Logic Levels	11 (CARRY4=5 DSP48E1=2 LUT2=2 LUT5=1 LUT6=1)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.035ns

5. Simulation Waveform

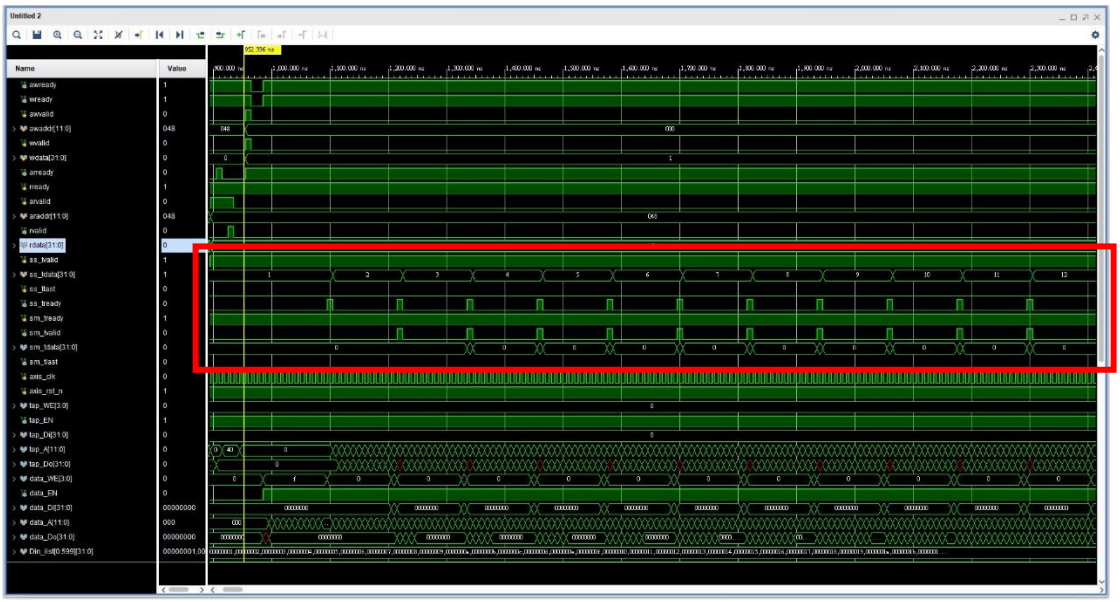
a. Coefficient program, and read back



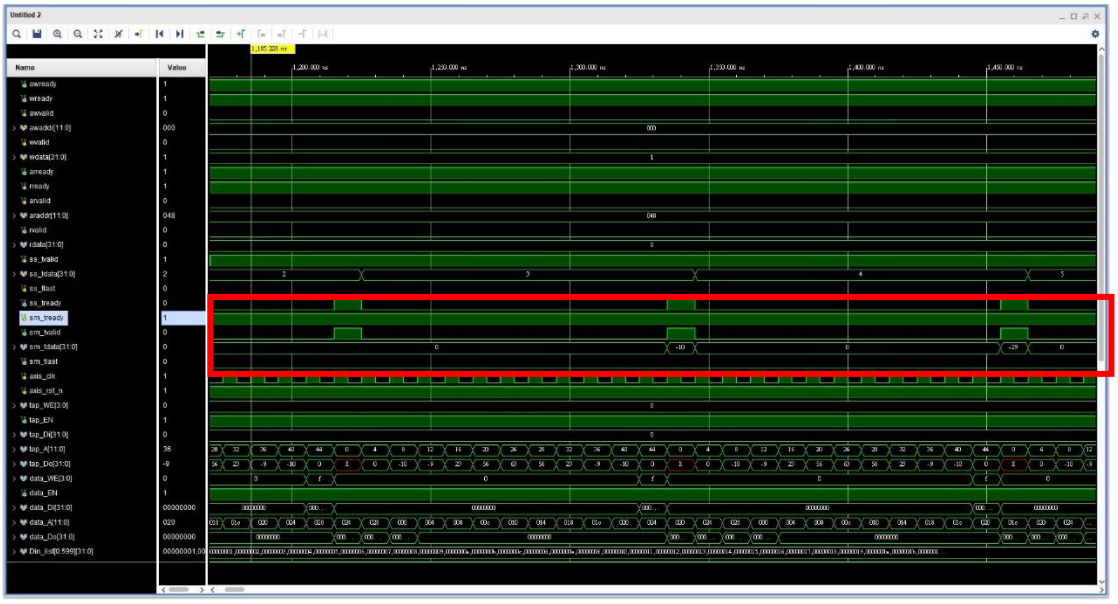
Tap coefficient wirte

Tap coefficient read back

b. Data-in stream-in

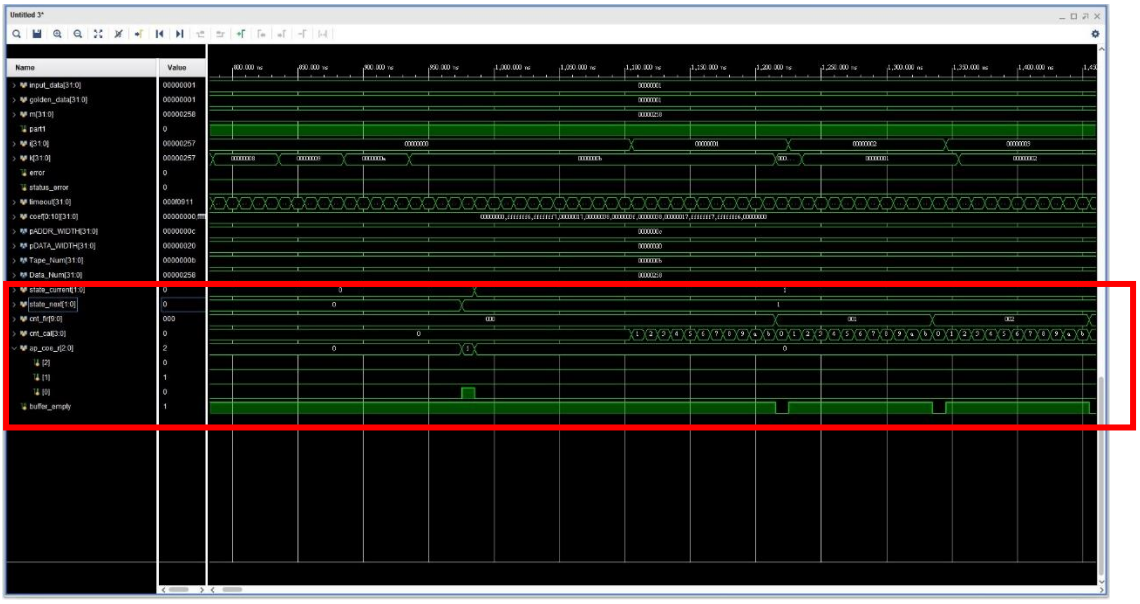


c. Data-out stream-out

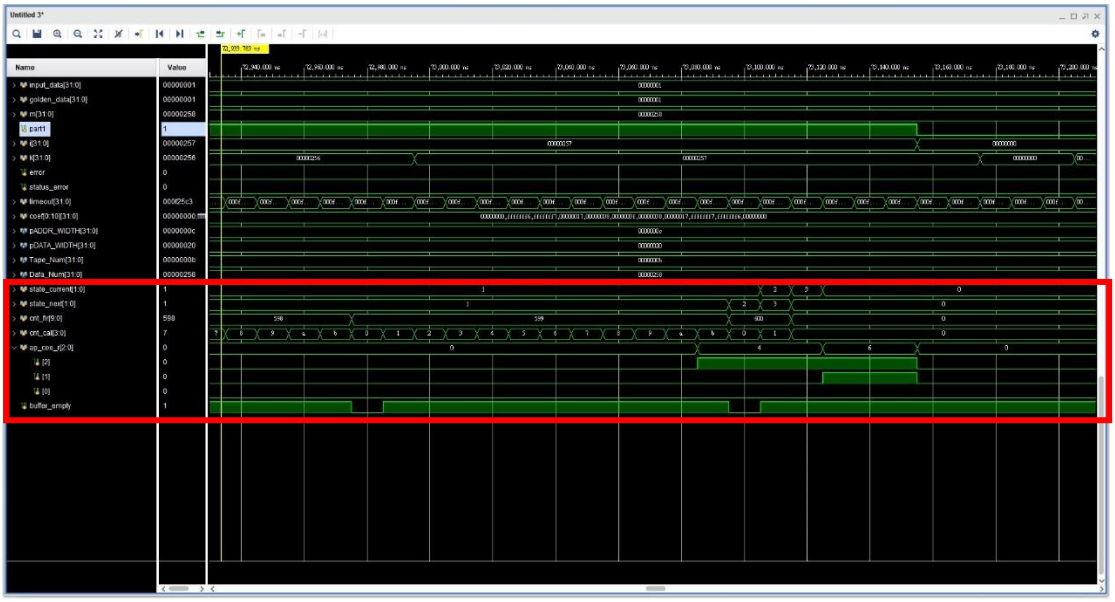


d. RAM access control

e. FSM



Ap_start = 1, fir start



Ap_done = 1, Ap_idle = 1, fir done