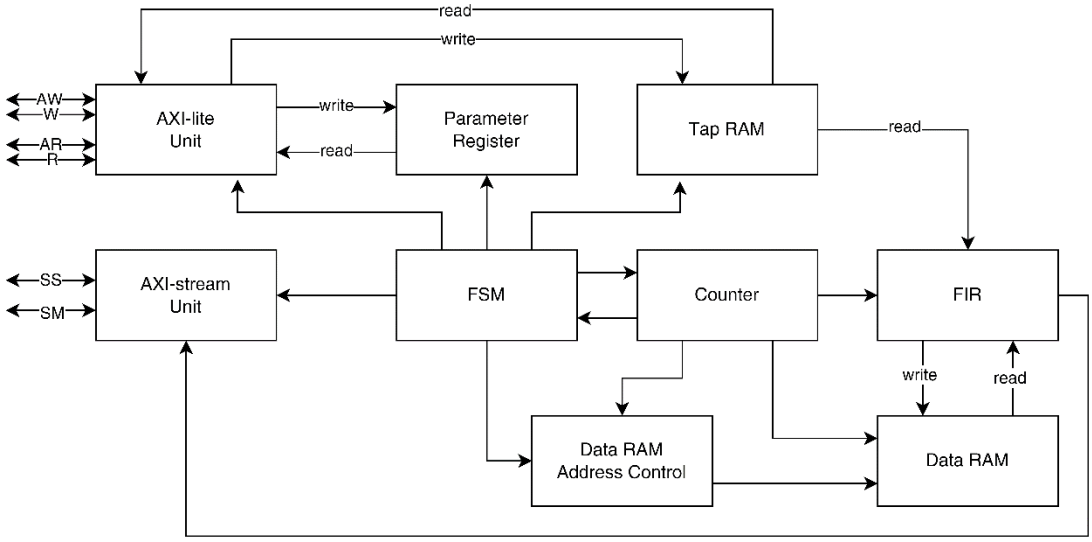
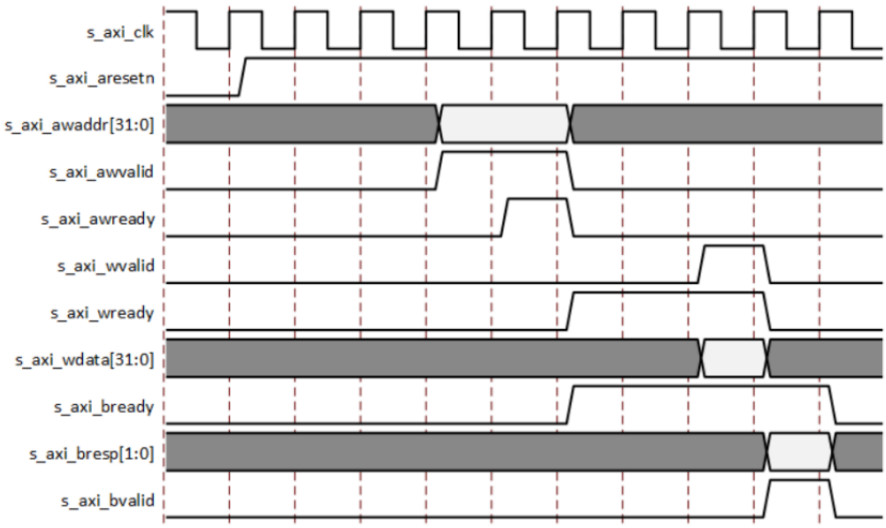
SoC Lab3_report R12921006 黃芯柔

- **Block Diagram:**

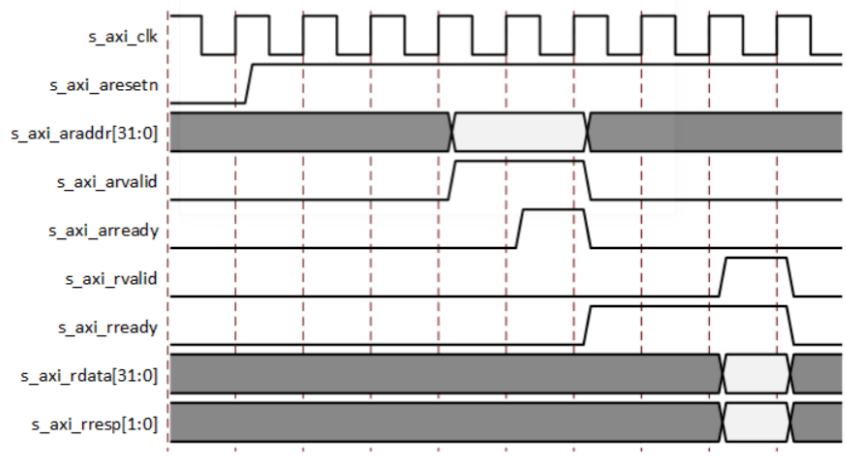


- **Describe Operation:**
  - ✓ **AXI-Lite**
    - **a. Write Channel**



我進行了一些邏輯上的調整，將寫入就緒狀態（write ready）和地址寫入就緒狀態（awrite ready）的控制分開。這兩個就緒狀態的邏輯相似，只要尚未發生 hand shake，例如，寫入請求通道的 hand shake 還未發生，那麼就會允許就緒信號升高。這種設計有助於實現請求通道和數據通道的獨立性。直到這兩個通道的 hand shake 都發生後，才會將地址和數據保存在 IP（內部處理單元）中。然後根據需要，可以將數據寫入 ap register，或者輸入數據長度和 Tap RAM 的數值。

b. Read Channel



Read Channel 的邏輯與 Write Channel 相似，都是根據 hand shake 是否成立來決定就緒信號的升高。然而，由於 AXI Lite Read 通道的數據和有效（valid）信號在 Read 請求通道的 hand shake 發生後才輸出，因此如果 Read 請求通道的 hand shake 未發生，Read 通道的有效信號就不會升高。這樣的設計有助於確保控制的正確性。此外，讀取的數據選擇也是根據讀取地址的值來選擇相應的結果輸出。

一旦 Read 數據通道的 hand shake 發生後，再重置兩個通道的 hand shake 狀態，以準備接收下一個指令輸入。換句話說，這種設計確保了在一次操作完成之前，不會發生下一次操作的開始。

✓ **AXI-stream**

AXI-Stream 在 Lab3 中用於接收計算資料以及輸出計算結果。因此我使用了一個 Counter 來確定何時提高接收端的就緒（ready）訊號。舉例來說，在 Counter 為 0 時，就會令 ready=1 以準備接收資料。如果 hand shake 發生，允許 Counter 遞增，以控制後續的 BRAM 和 FIR 進行計算。直到 Counter 達到 22，將計算結果存儲在輸出緩衝區中。因此，如果輸出緩衝區中有值，就會設定 sm channel 的有效（valid）訊號並輸出資料，直到 hand shake 成立後，將輸出緩衝區標記為空，從而允許下一組資料等待 sm channel 傳輸。

✓ **FSM**

我把狀態機設定成 4 種狀態，分別是 idle、operation、complete、done

1. idle：在此狀態下，可以輸入 TAP 數據和一些數據長度參數進行設定，直到 ap_start=1，然後進入 Operation 狀態。
2. operation：一旦進入此狀態，將首先清空 DataRAM 中的所有數值為 0，然後開始與 Counter 進行計算。例如，當 Counter 為 0 時，會接收 ss channel 的數據進入 FIR 運算，同時會不斷與 TAP RAM

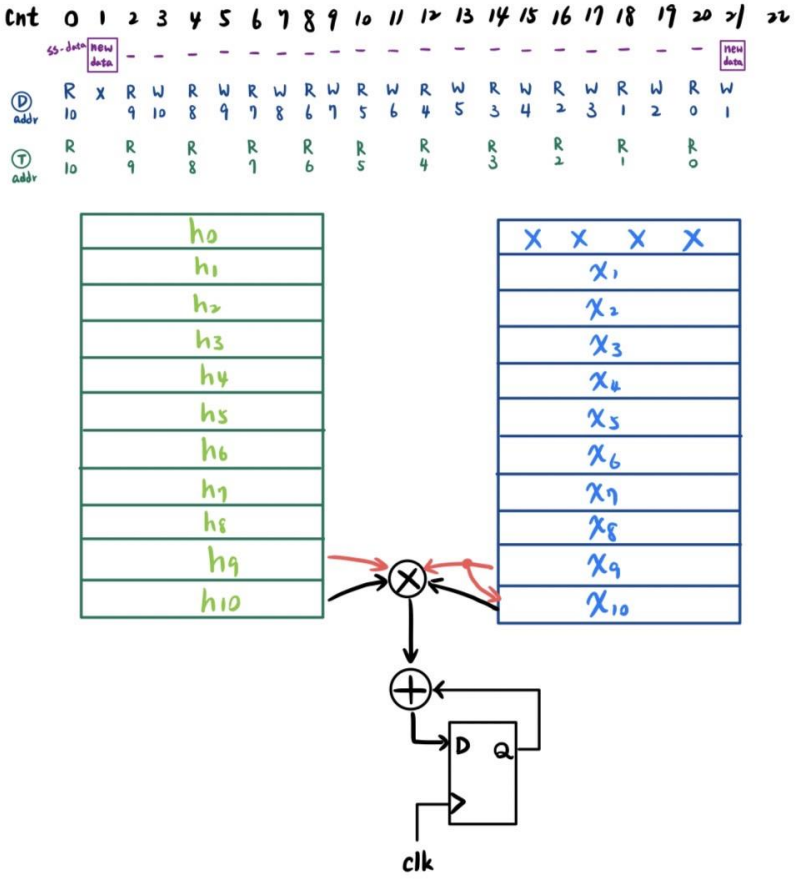
進行讀取以及 DataRAM 進行讀寫。最終當 Counter 為 22 時，如果 sm channel 沒有讀取 FIR 的計算結果，Counter 會保持在 22，直到數據成功被讀取。之後，Counter 會增加 1，直到最後一組數據計算完成，然後進入 complete 狀態。

3. complete：進入這個狀態表示 FIR 已經完成了所有計算，但是最後一組數據尚未被讀取。因此，如果 hand shake 發生後，就會離開這個狀態並進入 done 狀態。
4. done：這個狀態表示 FIR 已經完成了所有計算，且所有數據都已經被讀取。因此，下一個周期將返回到 idle 狀態，等待外部控制命令的輸入。

✓ **FIR**

根據作業的要求，使用了一個乘法器以及一個加法器，透過 counter 控制 FIR 的輸入進行計算，直到 counter 為 22 時就能完成一筆 FIR 運算。

✓ **Address Control**



我利用 counter 來控制 DataRAM 和 TapRAM 的存取，首先計算順序從 h10*x(t-10)開始，在 counter=0 時設定 address，等到 counter=1 時

就可以讀取這兩筆數據進行運算，接著在 counter=2 時處理 h9*x(t-9)，counter=3 時會將計算後的 y(9)與先前的 y(10)相加，同時把 x(t-9)存入下面一格的 entry，產生 shifter register 的效果，以此類推，直到 counter=20 讀取 address 為 0 的兩筆 data，但在計算上 x(t)是直接使用新輸入的 ss data，並在 counter=21 時將 ss data 寫入第一格，結束。

- ✓ **ap_coefficient Control**

ap_ coefficient 由於只需要實現 3 個 port，故我只開三個 bit 去儲存，並且由於只有 ap_start 是可以被寫入的，其他的 bit 數皆沒有辦法藉由外部去寫入。

我利用上述提到的 FSM 對於不同 ap_coefficient 進行控制：

1. ap_start：在外部透過 axi_Lite 拉至 1 之後，FSM 進入 start 的狀態，並將 ap_start 拉回 0。
2. ap_idle：初始狀態為 1，並且在 FSM 狀態為 idle 時設定為 1，當 ap_start 為 1 的時後將其設定為 0。
3. ap_done：初始狀態為 0，並且在 FIR 處理最後一筆 data 且 Yn 發生 handshake 的下一個 cycle 將其拉為 1。

- ✓ **Testbench**

我執行兩次測試。首先檢查 ap_idle 是否為 1。然後開始使用 AXI-Lite 寫通道來傳輸 TAP coefficient，傳輸完 11 組數據後，開始使用 AXI-Lite 讀通道來檢查已經寫入 TAP RAM 的數據。檢查完成後，開始將 ss_ready 設為 1，以接收 AXI-stream slave 通道的數據。在傳輸數據的同時，testbench 還通過 AXI-stream master 通道來接收和比對 FIR 的計算結果。直到 ss channel 發送倒數第二組輸入數據後，將使用 AXI-Lite 來檢查 ap_idle 和 ap_done 是否為 0。然後才發送最後一組輸入數據。接下來在 sm channel 接收了 599 組計算輸出後，也會立即通過 AXI-Lite 檢查 ap_idle 和 ap_done 同時為 0，因為此時 FIR 仍在計算最後一組數據。最後在 sm channel 接收完最後一組 FIR 輸出後，再次使用 AXI-Lite 檢查 ap_idle 和 ap_done 都為 1，完成第一次模擬。

然後進行第二次模擬，幾乎與第一次相同，不同在於當 sm channel 讀取倒數第二組數據時，會延遲 15 個周期後啟動 AXI-Lite 檢查 ap_idle 是否為 1。因為理論上在經過這麼多延遲後，FIR 應該已經完成最後一組數據的計算，應將 ap_idle 設定為 1，並等待最後一組結果被 sm channel 讀取。因此，當 sm channel 讀取最後一組數據時，將使用 AXI-Lite 檢查 ap_idle 是否為 0 且 ap_done 為 1，因為 ap_idle 已經被讀取，應設定為 0，而且因為數據已經全部發送完畢，ap_done 應設定為 1。

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          4] Golden answer:          35, Your answer:          35
[PASS] [Pattern          5] Golden answer:         158, Your answer:         158
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 =       13914
```

2. 第二次模擬

```
-----------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 =       14006
[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
```

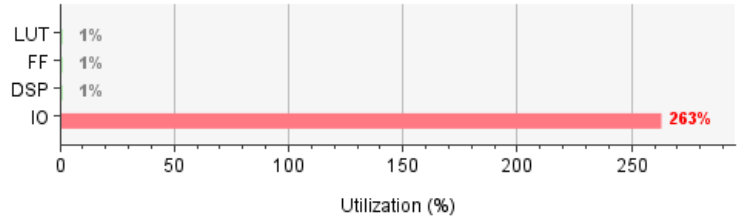
```
[PASS] [Pattern        592] Golden answer:       -2196, Your answer:       -2196
[PASS] [Pattern        593] Golden answer:       -2013, Your answer:       -2013
[PASS] [Pattern        594] Golden answer:       -1830, Your answer:       -1830
[PASS] [Pattern        595] Golden answer:       -1647, 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 =       27826


$finish called at time : 278265 ns : File "D:/soc/Lab3/lab3-2/fir_tb.v" Line 275
INFO: [USF-XSim-96] XSim completed. Design snapshot 'fir_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:06 . Memory (MB): peak = 2695.297 ; gain = 19.309
```

第一次的模擬總共花費的時間為：13914-96=13818 個 clock cycle，而第二次的模擬為 ：13820 個 clock cycle，會稍多的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。

- **Resource usage**

Resource	Utilization	Available	Utilization %
LUT	330	53200	0.62
FF	162	106400	0.15
DSP	3	220	1.36
IO	329	125	263.20



- **Timing Report**

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.849 ns	Worst Hold Slack (WHS):	0.142 ns	Worst Pulse Width Slack (WPWS):	6.500 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:	246	Total Number of Endpoints:	246	Total Number of Endpoints:	163

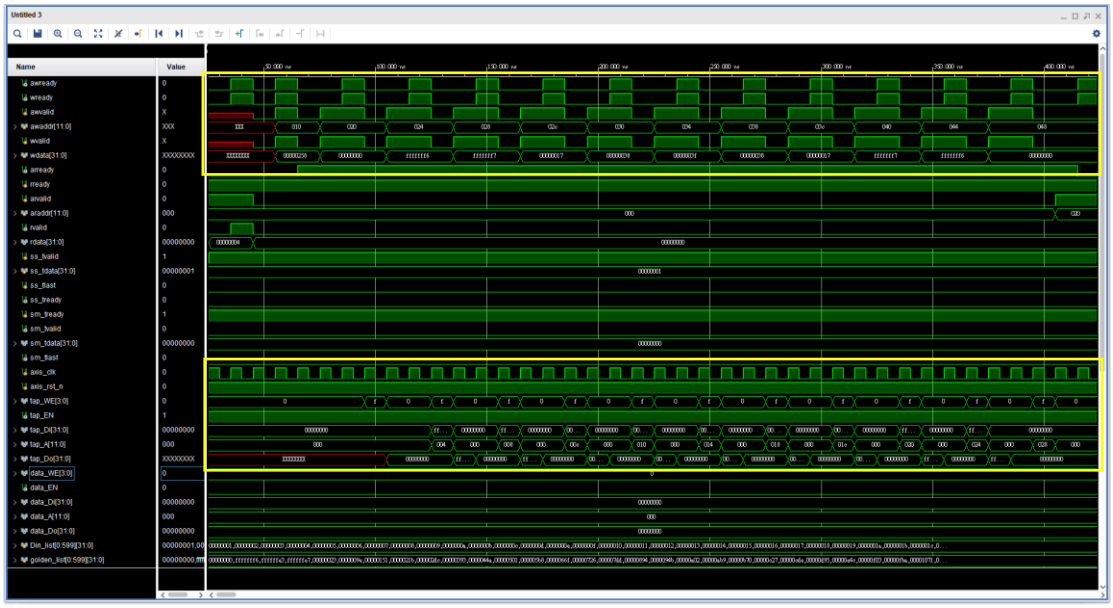
All user specified timing constraints are met.

Name	Path 1
Slack	0.849ns
Source	cnt_cal_reg[3]/C (rising edge-triggered cell FDCE clocked by axis_clk {rise@0.000ns fall@7.000ns period=14.000ns})
Destination	fir_out_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	13.014ns (logic 8.692ns (66.787%) route 4.322ns (33.213%))
Logic Levels	12 (CARRY4=5 DSP48E1=2 LUT2=2 LUT3=1 LUT5=1 LUT6=1)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.035ns

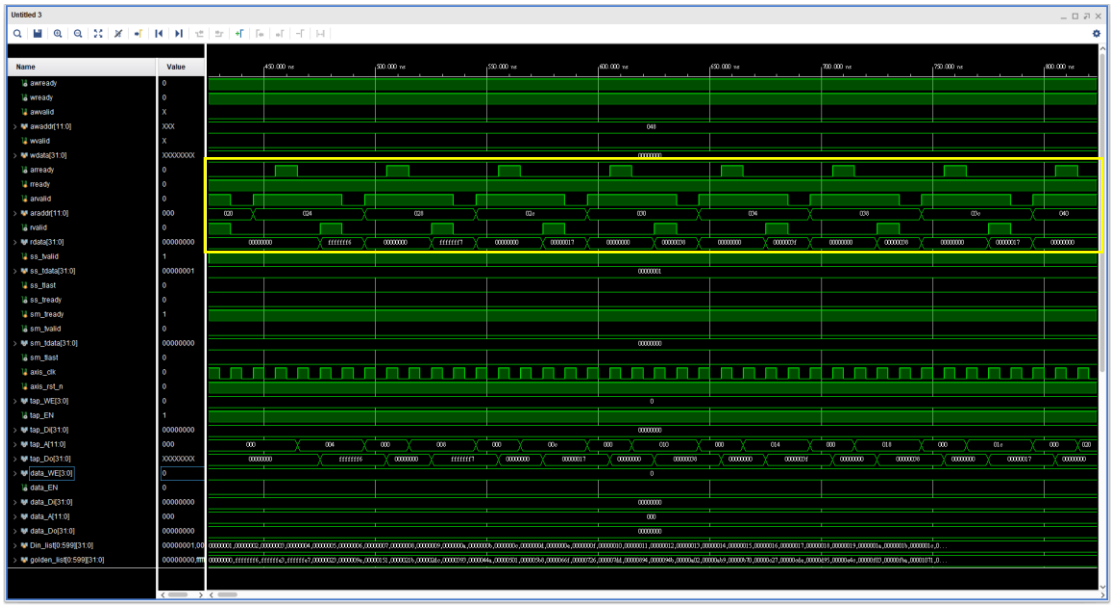
- **Simulation Waveform**

1. Coefficient program, and read back

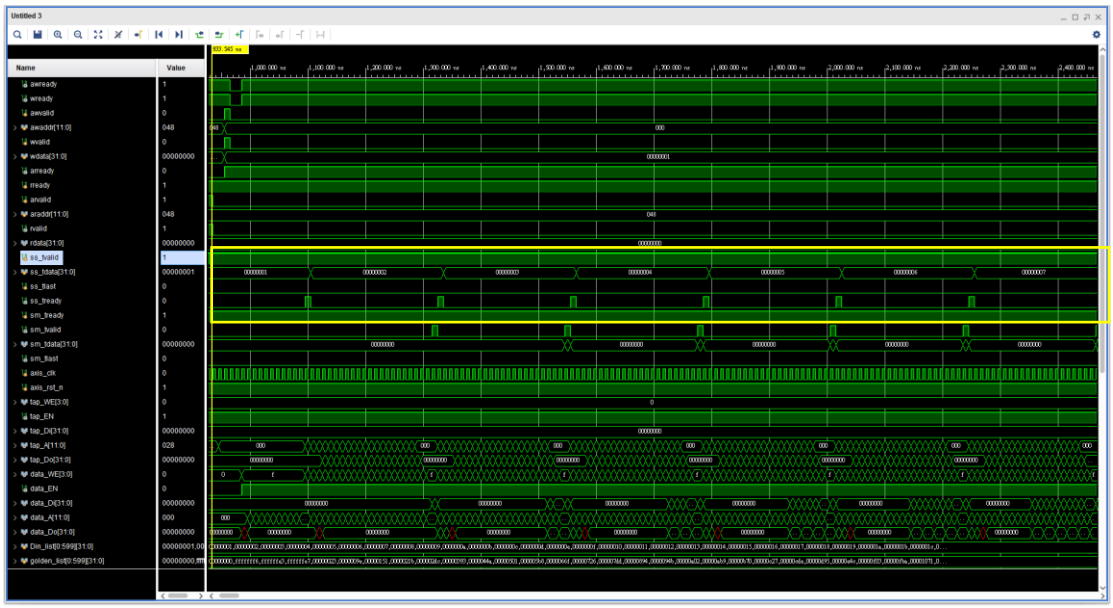
Tap coefficient wirte:



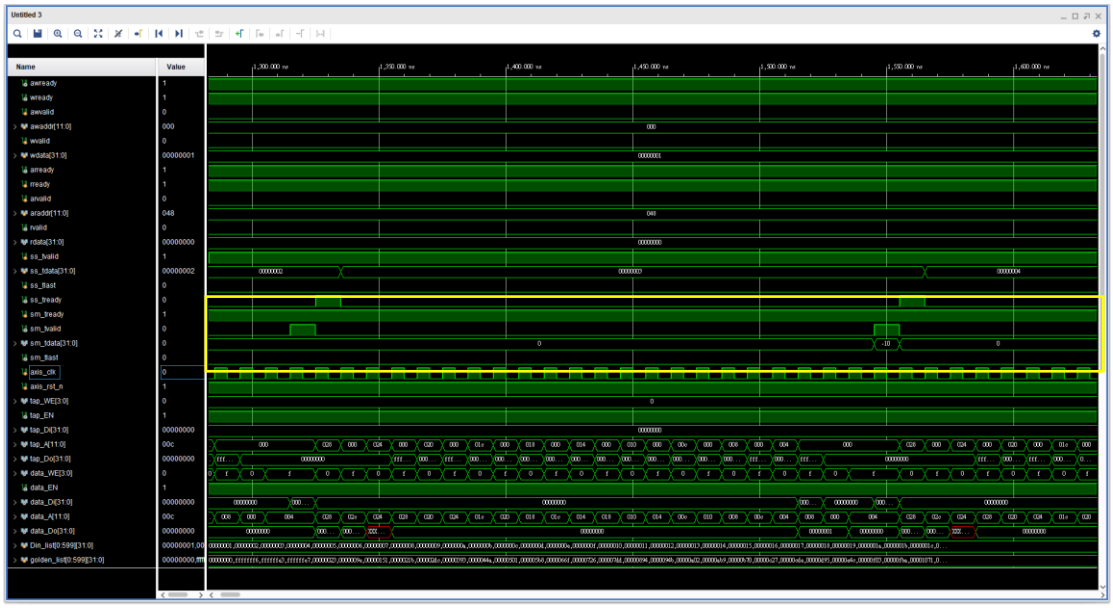
Tap coefficient read:



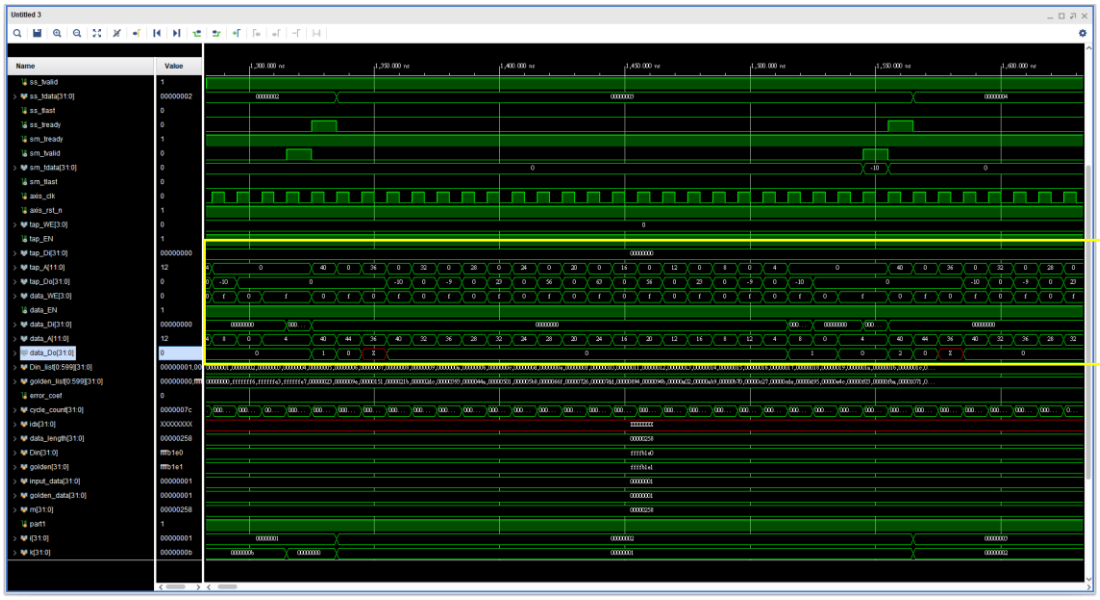
2. Data-in stream-in



3. Data-out stream-out



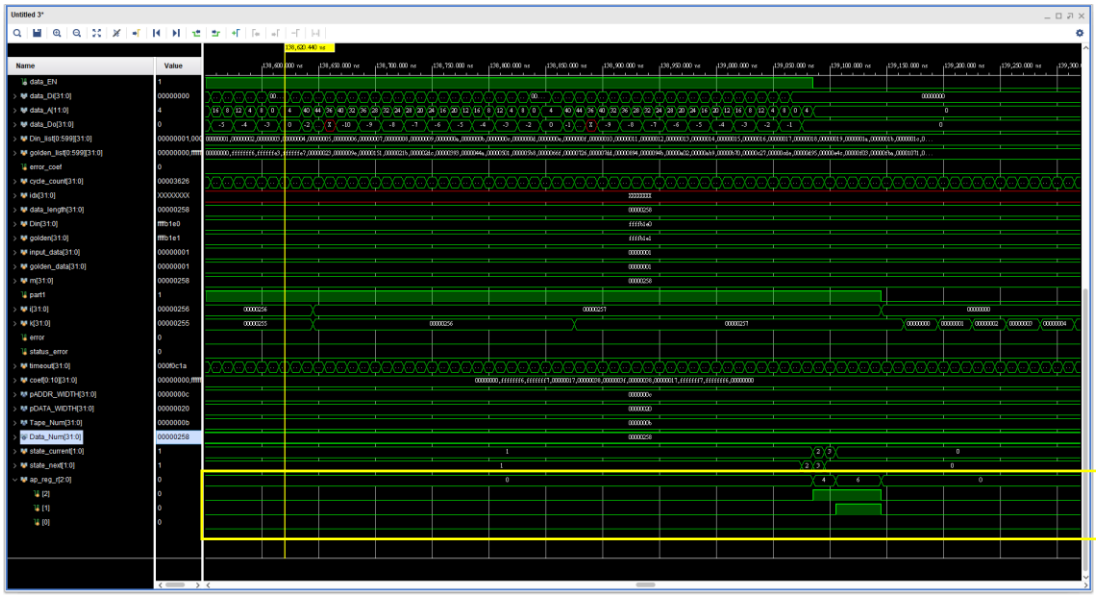
4. RAM access control



5. FSM

ap_start = 1

ap_done = 1, ap_idle = 1


Untitled 3*
Name
Value
data_EN
data_D[31:0]
data_A[11:0]
data_Do[31:0]
Din_list[0:599][31:0]
golden_list[0:599][31:0]
error_coef
cycle_count[31:0]
idx[31:0]
data_length[31:0]
Din[31:0]
golden[31:0]
input_data[31:0]
golden_data[31:0]
m[31:0]
part1
i[31:0]
k[31:0]
error
status_error
timeout[31:0]
coef[0:10][31:0]
pADDR_WIDTH[31:0]
pDATA_WIDTH[31:0]
Taps_Num[31:0]
Data_Num[31:0]
state_current[1:0]
state_next[1:0]
ap_reg_ig[2:0]
[2]
[1]
[0]