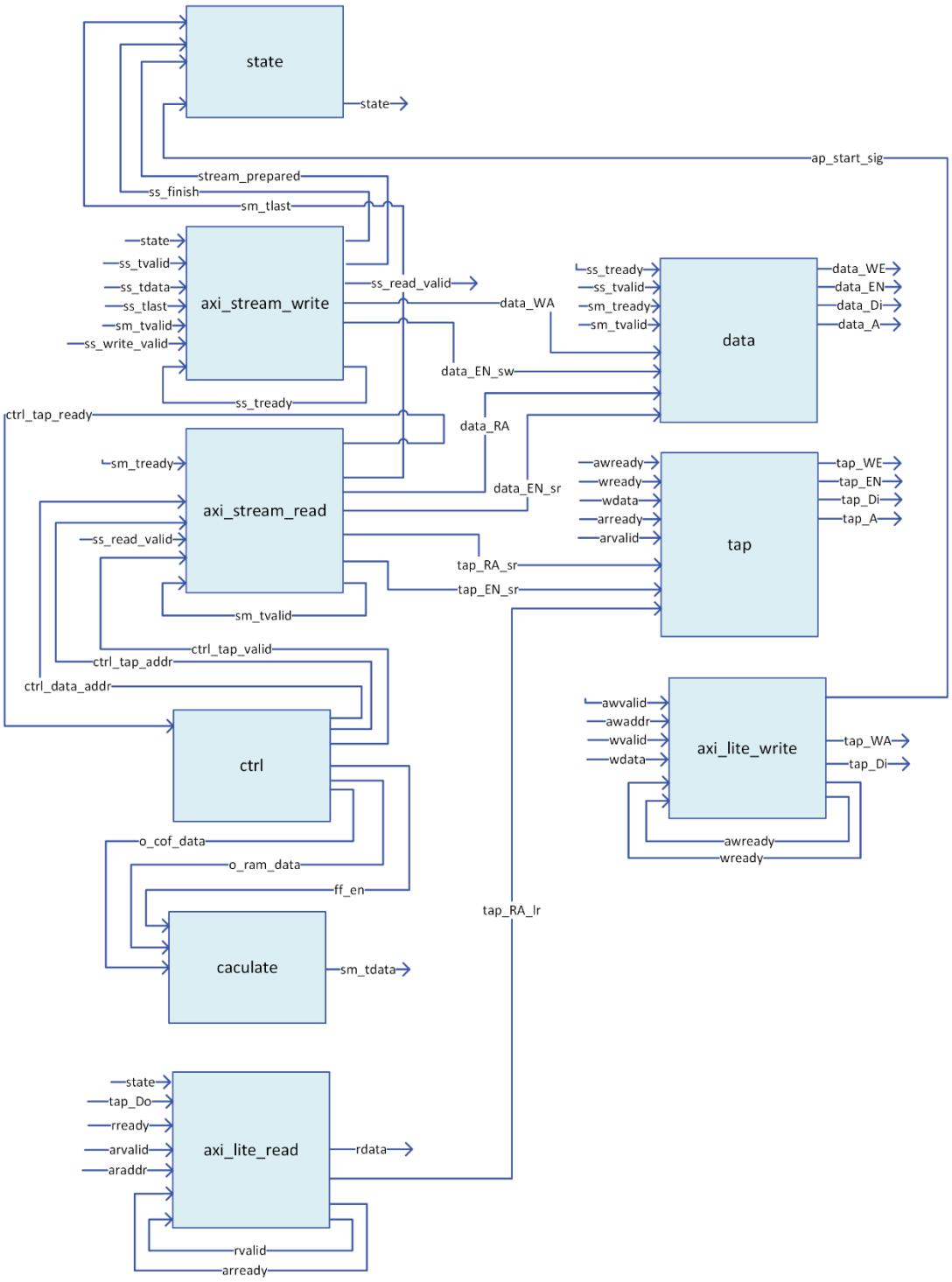
SOC LAB#3 Report

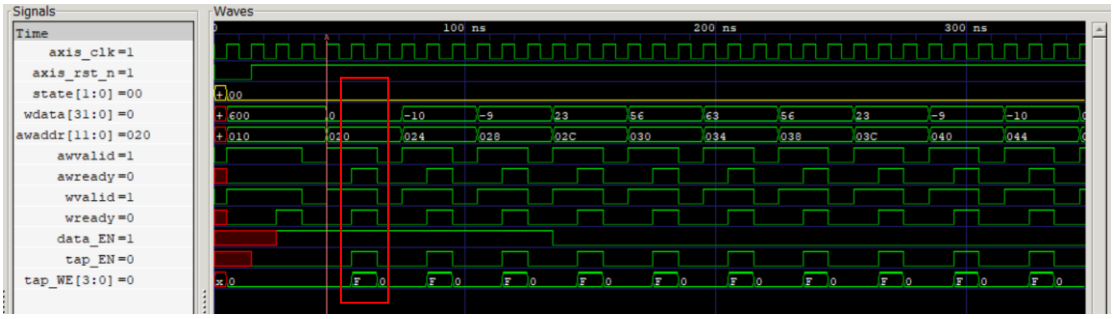
1. Block Diagram



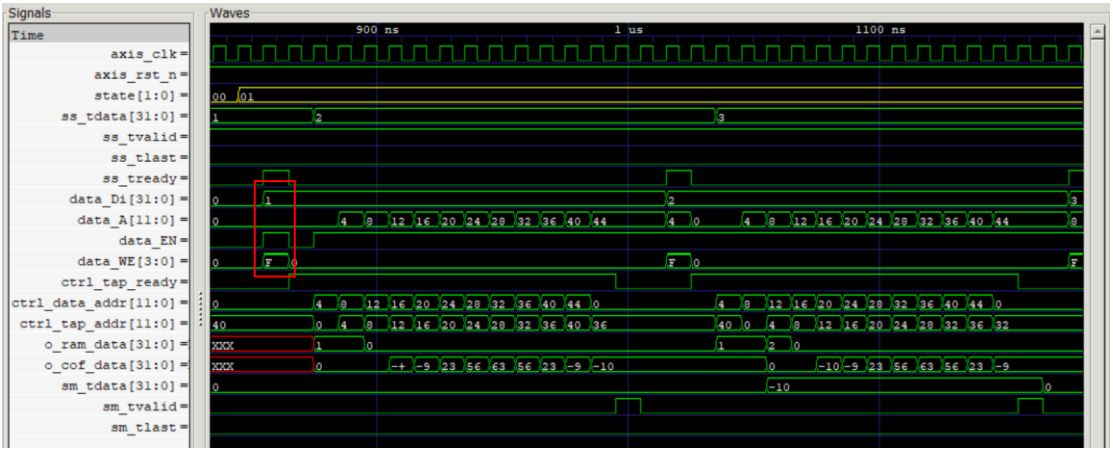
2. Describe operation

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

當 tap_EN = 1 與 tap_WE = 4'hF 時，會將 wdata 寫進去 awaddr(會做 shift)的位置。



當 data_EN = 1 與 data_WE = 4'hF 時，會將 ss_tdata 寫進去 data_A 的位置。



2.2 How to access shiftram and tapRAM to do computation

ctrl 模組會輸出對應的 tap_addr 與 data_addr，之後從 RAM 讀出數值。

e.g. RAM_size = 3 * data_size
data：1, 2, 3, 4
tap：-1, 3, -1

1st

data_ram_addr	A0	A1	A2
tap_ram_addr	A2	A0	A1
data	1	0	0
tap	-1	-1	3

2nd

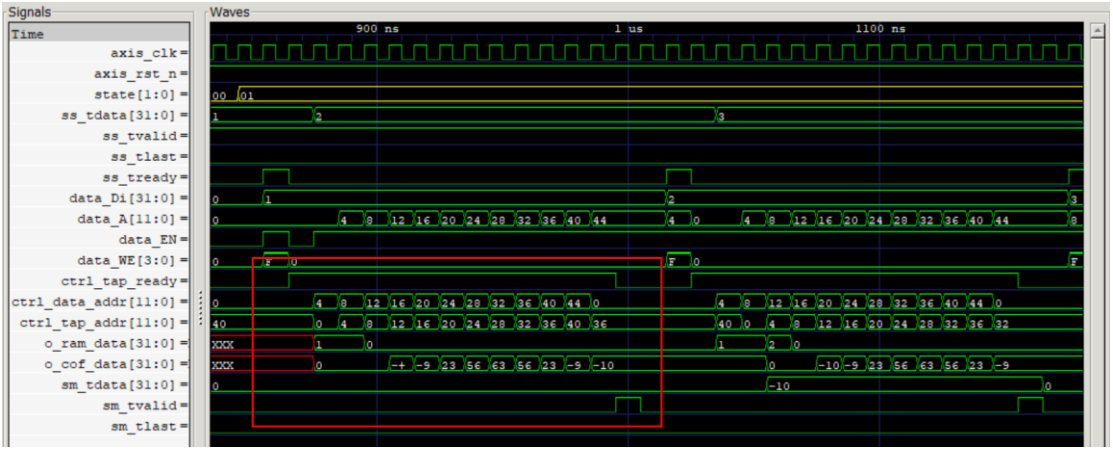
data_ram_addr	A0	A1	A2
tap_ram_addr	A1	A2	A0
data	1	2	0
tap	3	-1	-1

3rd

data_ram_addr	A0	A1	A2
tap_ram_addr	A0	A1	A2
data	1	2	3
tap	-1	3	-1

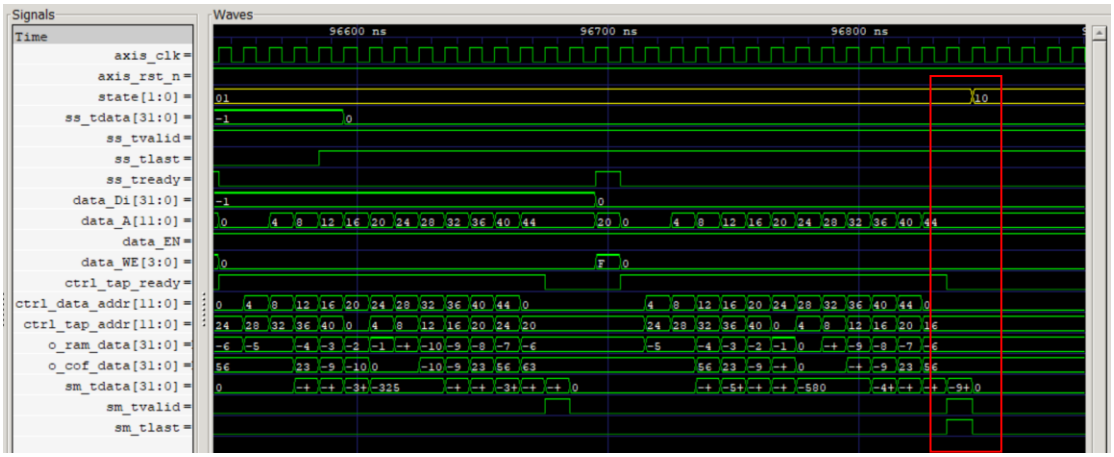
4th

data_ram_addr	A0	A1	A2
tap_ram_addr	A2	A0	A1
data	4	2	3
tap	-1	-1	3



2.3 How ap_done is generated.

當 ss_tlast = 1 與 sm_tlast = 1 後，state = ap_done(2'b10)。



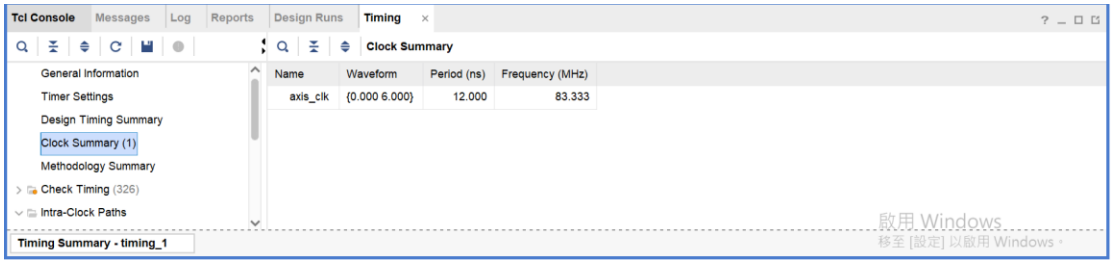
3. Resource usage

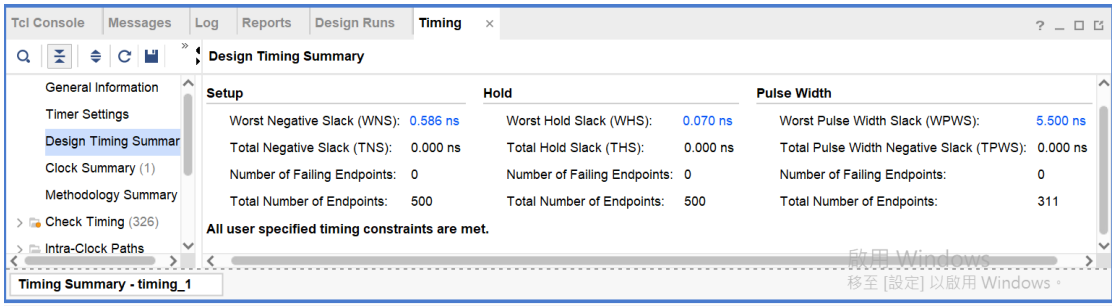
```
1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |  297 |     0 |          0 |     53200 |  0.56 |
|   LUT as Logic          |  297 |     0 |          0 |     53200 |  0.56 |
|   LUT as Memory         |    0 |     0 |          0 |     17400 |  0.00 |
| Slice Registers         |  200 |     0 |          0 |    106400 |  0.19 |
|   Register as Flip Flop |  200 |     0 |          0 |    106400 |  0.19 |
|   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 |
+-------------------------+------+-------+------------+-----------+-------+
* Warning! The Final LUT count, after physical optimizations and full implementation, i
```

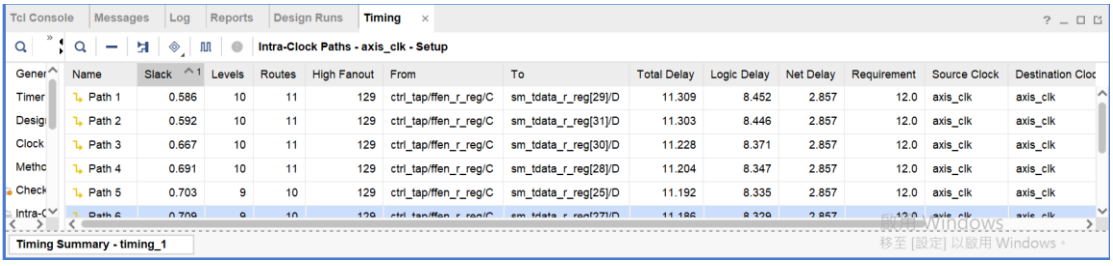
```
---------------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
          2 Input   12 Bit       Adders := 5
          2 Input    4 Bit       Adders := 2
+---Registers :
                    32 Bit    Registers := 3
                    12 Bit    Registers := 3
                     4 Bit    Registers := 3
                     1 Bit    Registers := 18
+---Multipliers :
                   32x32  Multipliers := 1
+---Muxes :
          2 Input   32 Bit        Muxes := 8
          3 Input   32 Bit        Muxes := 1
          2 Input   12 Bit        Muxes := 11
          2 Input    4 Bit        Muxes := 7
          2 Input    1 Bit        Muxes := 37
          3 Input    1 Bit        Muxes := 3
---------------------------------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------------------------------
```

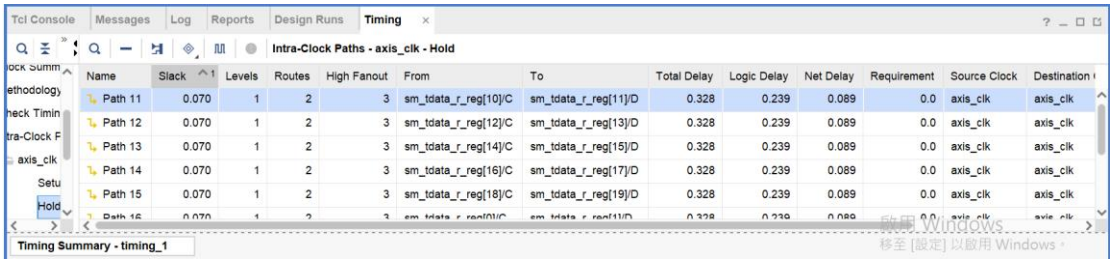
因為使用很多 reg 來儲存，所以可能 flip flop 使用比較多，是可以減少的。

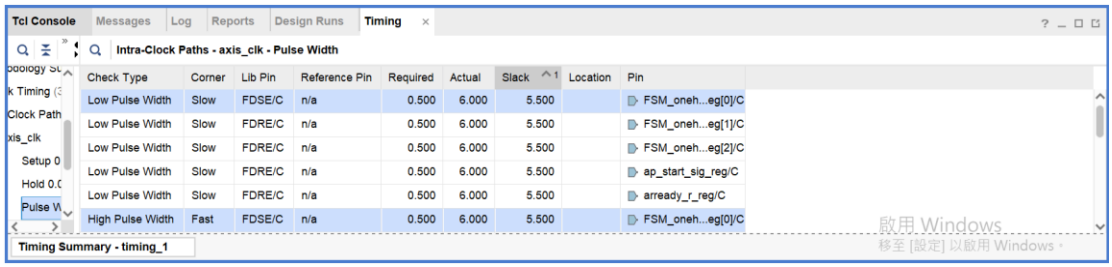
4. Timing Report





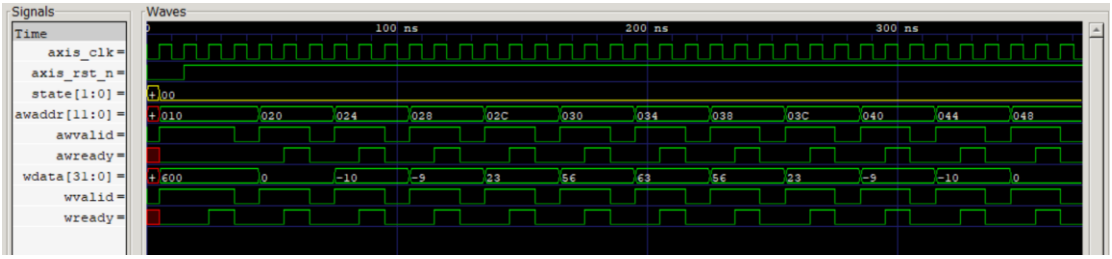




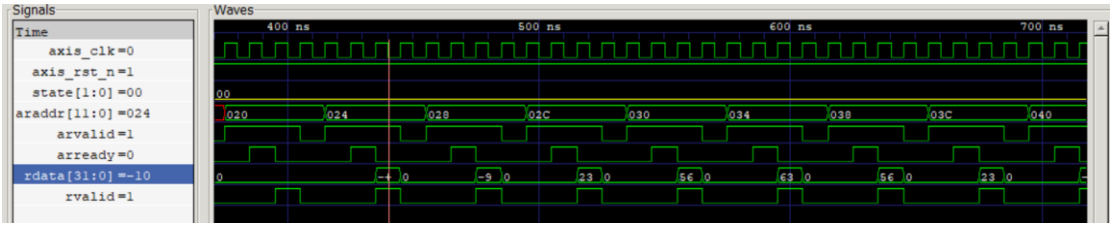


5. Simulation Waveform, show

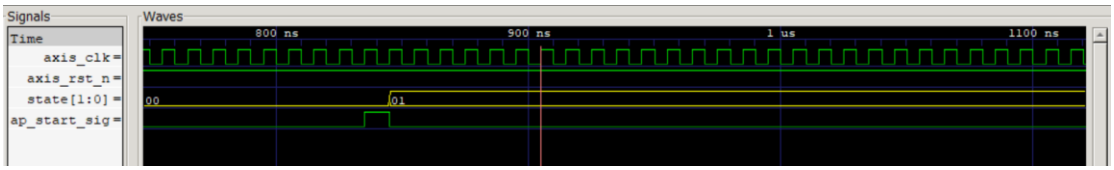
5.1 首先 state = ap_idle(2'b00)，開始的輸入 taps，並存在 RAM。



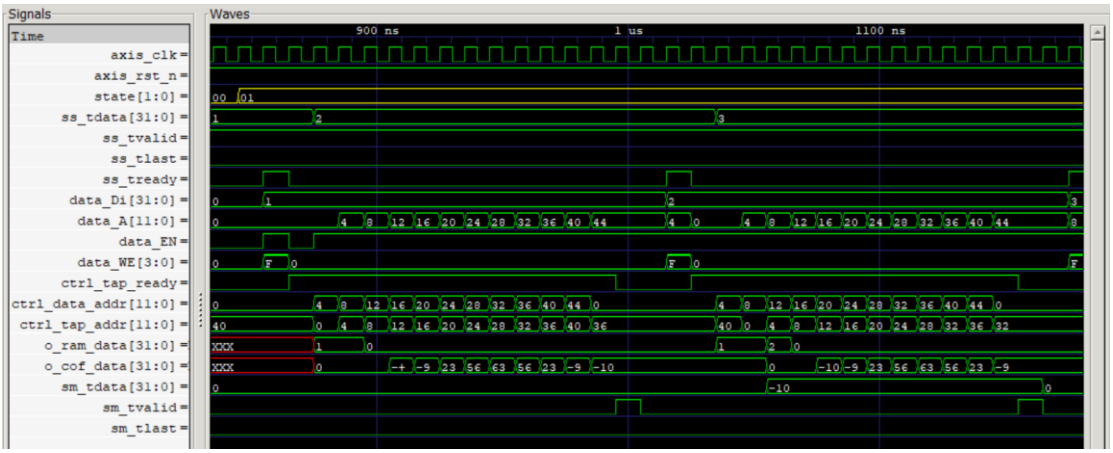
5.2 testbench 開始的檢查輸出的 taps。



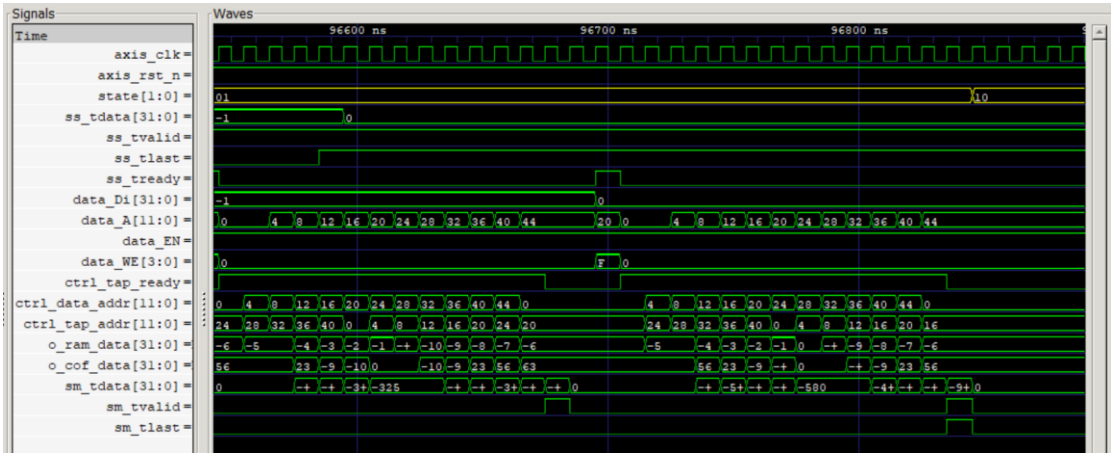
5.3 當接收到 testbench 的 ap_start 訊號，ap_start_sig = 1，state = ap_start(2'b01)。



5.3 state = ap_start(2' b01)後，會將 ss_tdata 寫入至 RAM，並且 ctrl_tap_ready = 1，ctrl 模組開始輸出對應的 tap_addr 與 data_addr，之後從 RAM 讀出數值做相乘累加，輸出結果，且 sm_tvalid = 1。



5.4 當 ss_tlast = 1 與 sm_tlast = 1 後，state = ap_done(2'b10)。


Signals
Waves
Time
axis_clk
axis_rst_n
state[1:0]
ss_tdata[31:0]
ss_tvalid
ss_tlast
ss_tready
data_Di[31:0]
data_A[11:0]
data_EN
data_WE[3:0]
ctrl_tap_ready
ctrl_data_addr[11:0]
ctrl_tap_addr[11:0]
o_ram_data[31:0]
o_cof_data[31:0]
sm_tdata[31:0]
sm_tvalid
sm_tlast
96600 ns
96700 ns
96800 ns