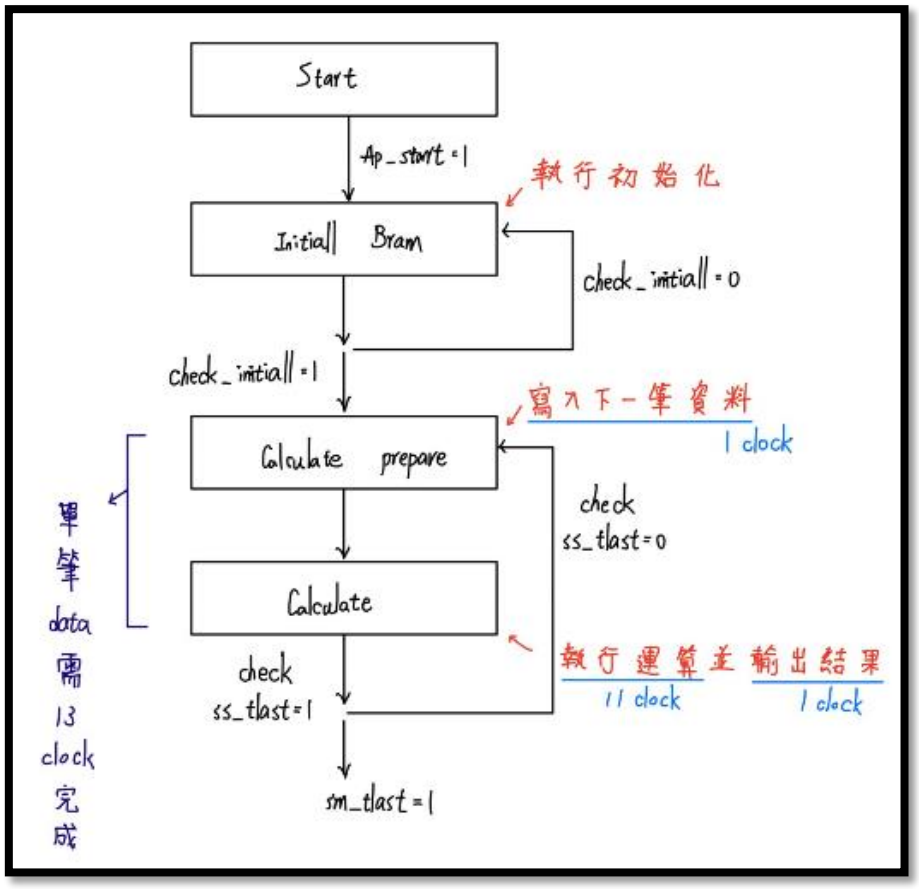
SoC Design Laboratory Lab_3

台灣科技大學 M11202207 呂彥霖

1. Block Diagram

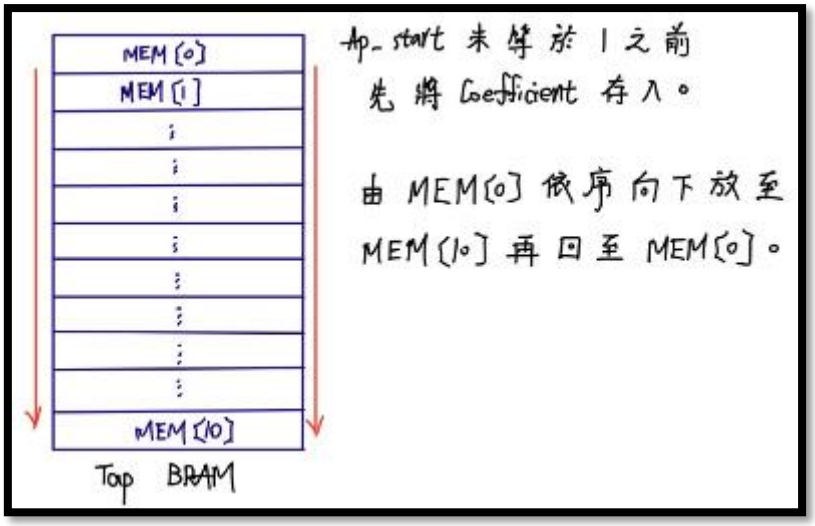


資料計算流程，當收到 Ap_start = 1 時 Ap_idle 會向下拉至 0，並開始執行 FIR Module。

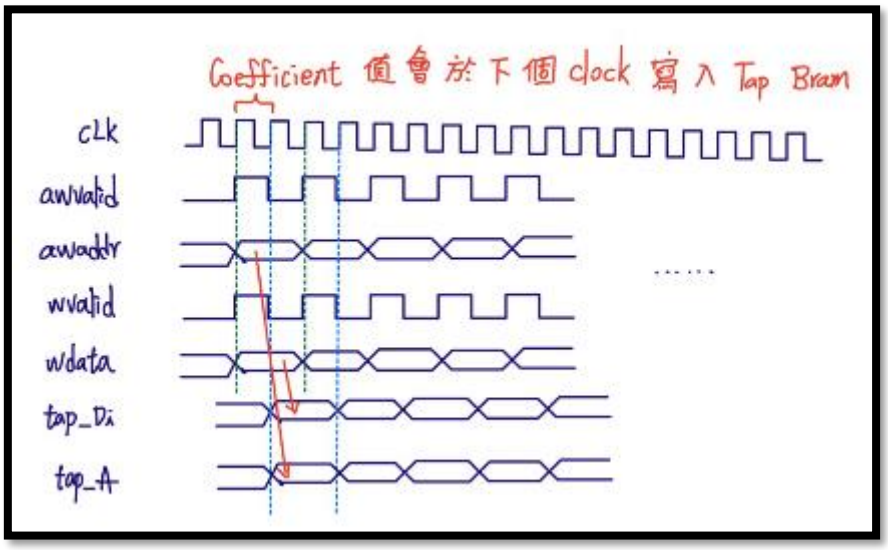
我的設計是先將 BRAM 每個 Block 先初始化至 0，省去前 10 筆資料還要做 Special Case 去特別處理他們的多餘設計，並於初始化完成後開始進入 Convolution 的運算。

進入運算後(報告後方呈現)，如上圖每筆 Data 會消耗 13 個 clock，含寫入資料 1 clock、運算 11 clock、輸出結果 1 clock，於最後收到 ss_tlast = 1 時判斷為最後一筆資料，並於該筆資料完成運算後輸出 sm_tlast = 1、Ap_done = 1，而 Ap_idle 因為 Ap_done = 1 而回至 1，此時 Ap_start 也回至 0 的位子，來結束整個 FIR module。

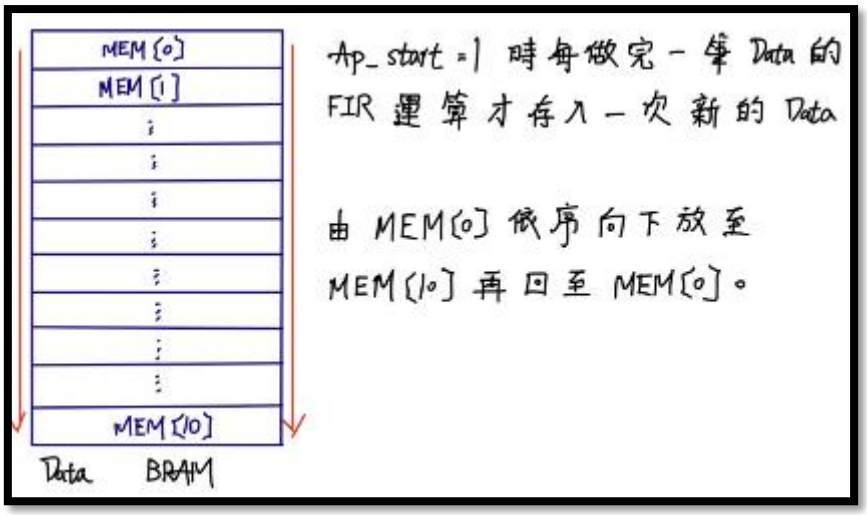
2. Tap Bram & Data Bram Storage Design



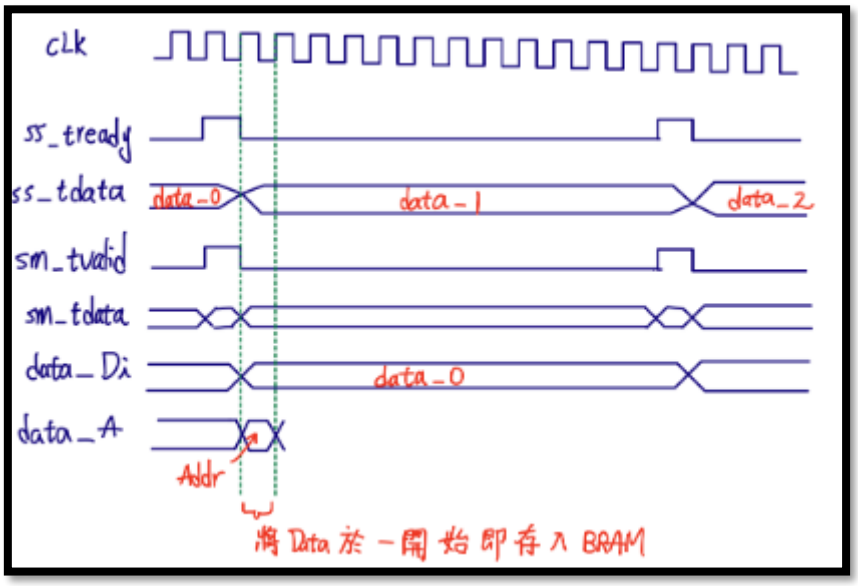
Tap Bram 儲存方式為從第一個 Block 放至最後一個 Block 依序存放，其值在 Ap_start = 1 開始之前，即會將資料存入，存放時預期波形圖如下。



因為該部分我切 2 個 States 來做，分出準備&儲存兩個狀態，因此會延遲 1 個 Clock ，箭頭為該筆資料存入的時間，由於資料送過來時 AXI4_lite 會將 awvalid 與 wvalid 升高來表示有傳遞資料的情況發生，因此可以透過偵測這兩個訊號，來決定何時收取資料。

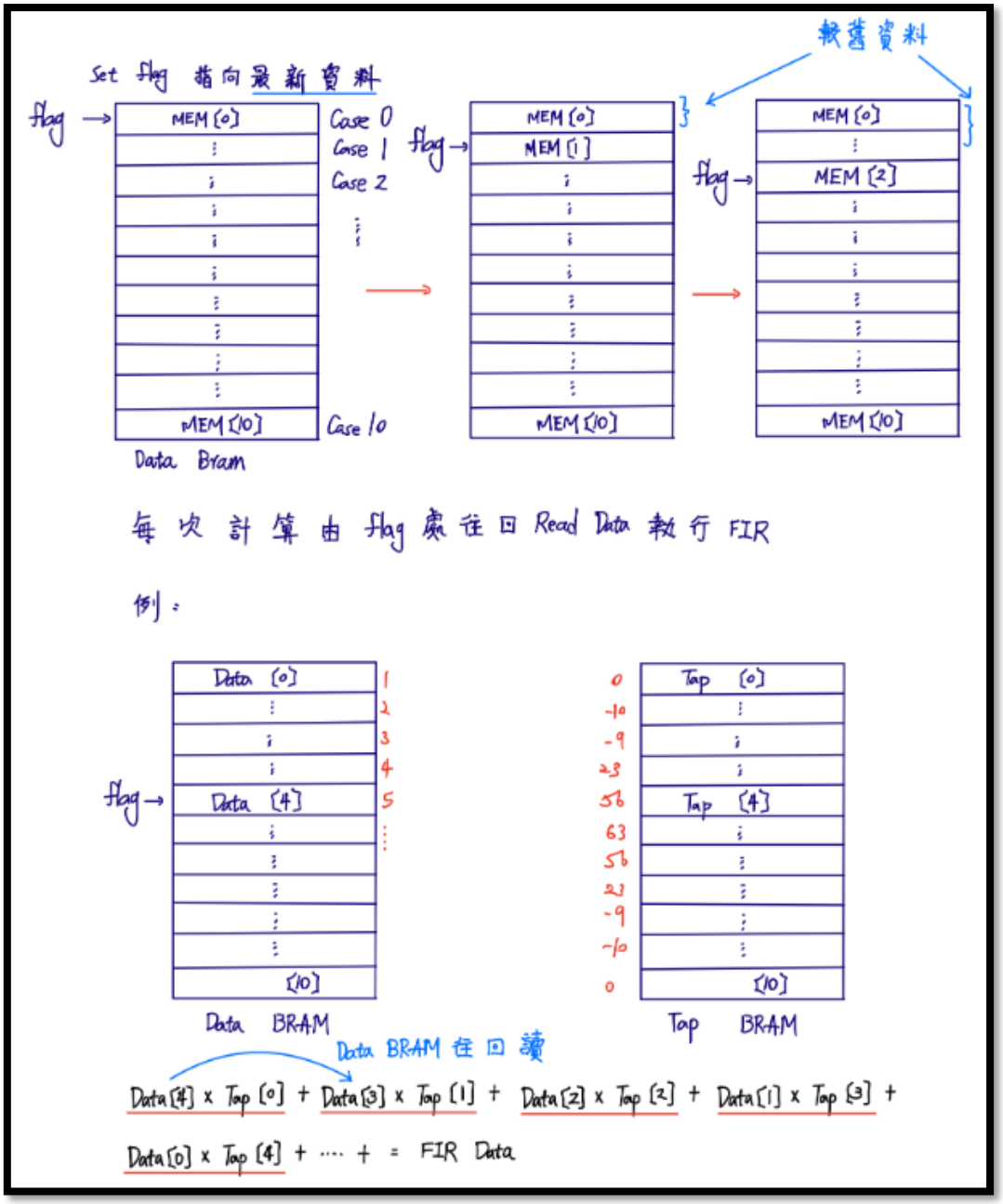


Data Bram 與 Tap Bram 儲存順序一樣，皆由最上層 Block 往下存取，直到 11 個 Block 皆存滿後再回至最上層取代掉先前 Data，預期存取波形如下。



Data Bram 存取資料的時間為 1 個 Clock，接收來源為採用 AXI4-Stream 的 ss_tdata 上的資料，並於上筆資料完成運算並輸出後，存取一筆新的資料進入，開始執行運算(報告後方詳細說明)。

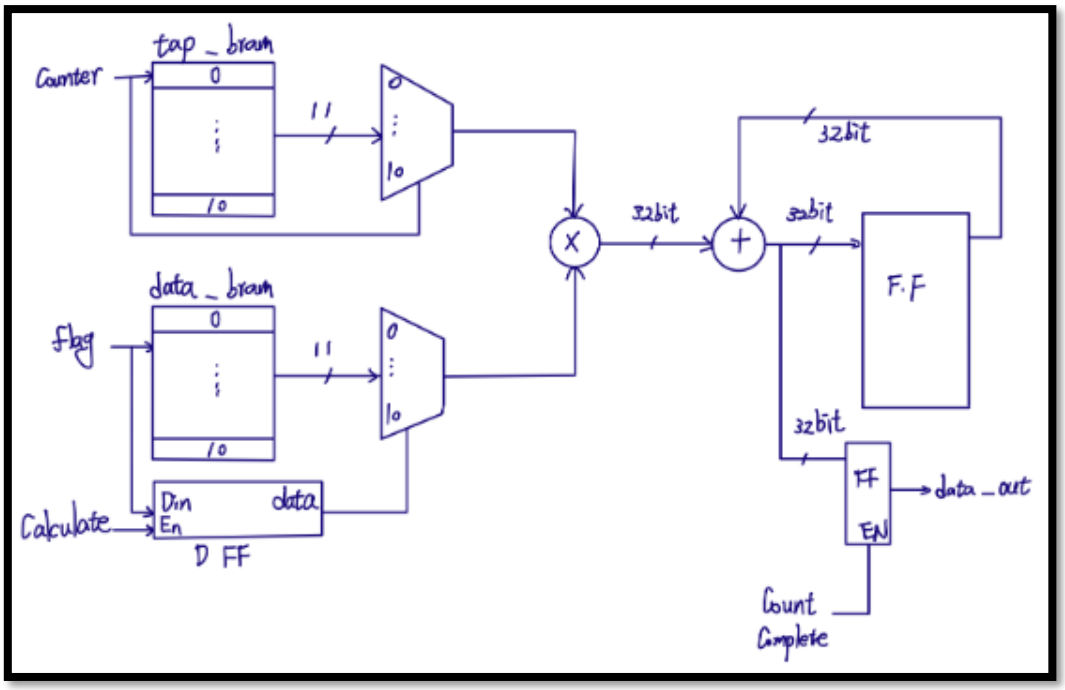
3. Tap Bram & Data Bram Data Processing



因為每次運算需要讀取的 Data Bram Block 不相同，在這裡我設計一個 Flag 指向最新一筆的資料，該 Flag 會隨著存入的資料新舊程度而改變，並由 addr = 0 依序指到 addr = 10。

透過 Flag 知道最新一筆資料後，由該筆資料往前將資料讀出，與 Tap Bram 資料進行 Convolution，例如 Flag = 4，則將 Data Bram 資料由 Block 4 -> 3 -> 2 -> 1 -> 0 -> 10 -> … -> 5 依此類推至當前 Flag 的下一筆，共 11 筆資料的輸出，藉此完成運算。

4. Data Path

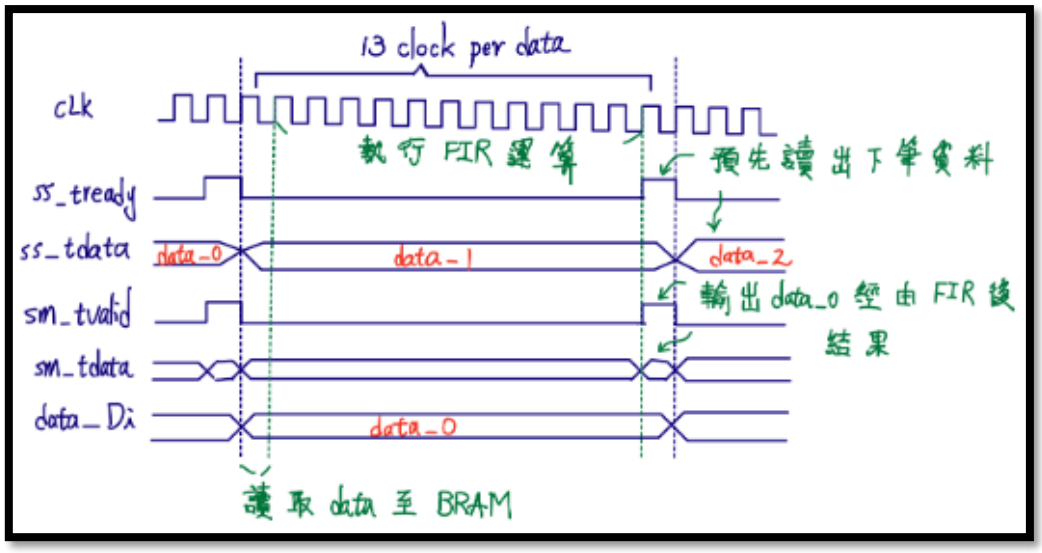


資料由左至右進行處理，Tap Bram 沒有 Flag，每次皆以順序向下輸出資料 11 筆，並由多工器來選擇需要輸出的資料，Data Bram 則有一 Flag，將 Flag 值透過暫存器存起後輸出到多工器，因為選取的資料為 Flag 當前值依序往前，因此使用多工器紀錄 Flag 數值。

資料藉由多工器選擇出後，經由一個乘法器與一個加法器進行運算，並透過後方一暫存器來存取當前資料至下一個 Clock，依此類推做 Convolution 運算，符合本作業規範的乘法器與加法器使用規劃。

整個 FIR Module 大量使用多工器與移位功能，因此只在 Convolution 階段使用一個乘法器與一個加法器，未在其餘任何處(後方報告呈現)使用加法器與其它運算符號。

5. Data Path Waveform



該圖為 FIR Data Path 執行 Convolution 運算時，預計產生之波形。Module 會在整個處理過程第 1 個 Clock 讀取 Data 至 Bram，並經由 11 個 Clock 的運算，最後於第 13 個 Clock 將 ss_tready 與 sm_tvalid 同時拉起。

ss_tready = 1 會將下一筆欲執行資料於 Testbench 由 ss_tdata 送出，而 sm_tvalid = 1 則是因為前述 11 個 Clock 算好的值需要進行輸出，sm_tdata 會立即輸出該值，因此整個過程拆分為 1 個 Clock 讀取、11 個 Clock 運算、1 個 Clock 輸出結果。

6. Resource Usage

DSP Open

Utilization | **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	219	53200	0.41
FF	100	106400	0.09
DSP	3	220	1.36
IO	322	125	257.60
BUFG	2	32	6.25

```
Part Resources:
DSPs: 220 (col length:60)
BRAMs: 280 (col length: RAMB18 60 RAMB36 30)
```

```
Start RTL Component Statistics
---------------------------------------------------
Detailed RTL Component Info :
+---Adders :
          3 Input   32 Bit       Adders := 1
+---Registers :
                       32 Bit    Registers := 2
                       12 Bit    Registers := 1
                        4 Bit    Registers := 6
                        3 Bit    Registers := 1
                        1 Bit    Registers := 12
+---Muxes :
          2 Input   32 Bit        Muxes := 4
          4 Input   32 Bit        Muxes := 2
         12 Input   32 Bit        Muxes := 1
          2 Input   12 Bit        Muxes := 5
          4 Input    4 Bit        Muxes := 3
         15 Input    4 Bit        Muxes := 2
          2 Input    4 Bit        Muxes := 1
          2 Input    2 Bit        Muxes := 1
          3 Input    2 Bit        Muxes := 1
          2 Input    1 Bit        Muxes := 23
          4 Input    1 Bit        Muxes := 12
         12 Input    1 Bit        Muxes := 2
```

DSP Close (本次報告繳交)

Utilization | **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	767	53200	1.44
FF	100	106400	0.09
IO	322	125	257.60
BUFG	2	32	6.25

```
Part Resources:
DSPs: 220 (col length:60)
BRAMs: 280 (col length: RAMB18 60 RAMB36 30)
```

```
Start RTL Component Statistics
---------------------------------------------------
Detailed RTL Component Info :
+---Adders :
          2 Input   32 Bit       Adders := 1
+---Registers :
                       32 Bit    Registers := 2
                       12 Bit    Registers := 1
                        4 Bit    Registers := 6
                        3 Bit    Registers := 1
                        1 Bit    Registers := 12
+---Multipliers :
                     32x32  Multipliers := 1
+---Muxes :
          2 Input   32 Bit        Muxes := 4
          4 Input   32 Bit        Muxes := 2
         12 Input   32 Bit        Muxes := 1
          2 Input   12 Bit        Muxes := 5
          4 Input    4 Bit        Muxes := 3
         15 Input    4 Bit        Muxes := 2
          2 Input    4 Bit        Muxes := 1
          2 Input    2 Bit        Muxes := 1
          3 Input    2 Bit        Muxes := 1
          2 Input    1 Bit        Muxes := 23
          4 Input    1 Bit        Muxes := 12
         12 Input    1 Bit        Muxes := 2
```

一開始有發現DSP似乎是預設開啟的狀態，因此乘法器與加法器皆透過DSP進行合成了，導致報告中出現 Adder 有 3 Input，與 LUT 使用量非常低，不同於以往使用乘法器後應該出現的資源使用量。

後面則是研究 Vivado 後發現關閉 DSP 的方式，關起後報告即出現乘法器，與 LUT 使用量大幅增加，並且可以清楚看到整個 FIR Module 僅使用一個加法器，因該作業是後來才說明只有 Convolution 時規範加法器數量，故我設計之初就已經規劃完全僅使用一個加法器。

7. Timing Report

```
------------------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------------------

Clock      Waveform(ns)          Period(ns)      Frequency(MHz)
-----      ------------          ----------      --------------
axis_clk   {0.000 2.100}         4.200           238.095


------------------------------------------------------------------------
```

Max Frequency

```
------------------------------------------------------------------------------------------------------------------------
From Clock:  axis_clk
  To Clock:  axis_clk

Setup :            0  Failing Endpoints,  Worst Slack        0.012ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.140ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        1.600ns,  Total Violation        0.000ns
------------------------------------------------------------------------------------------------------------------------
```

Slack 為正

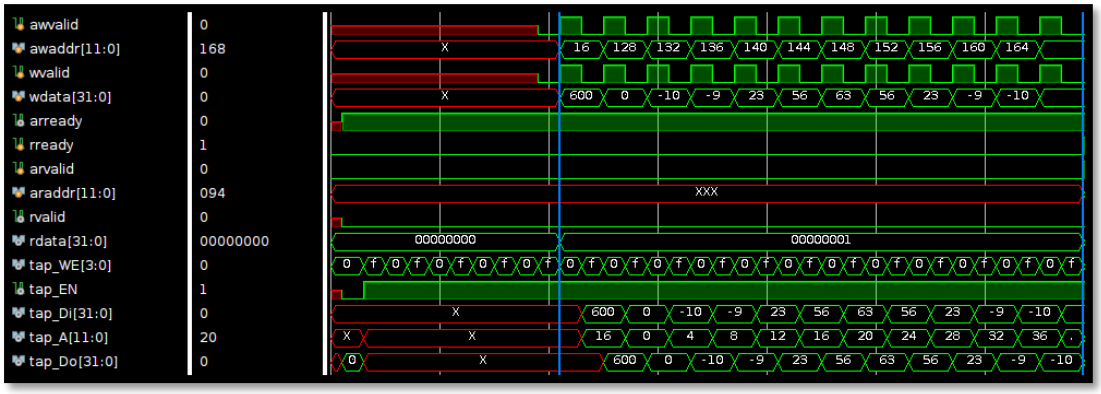
```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.012ns  (required time - arrival time)
  Source:                 cal_block_reg[0]/C
                            (rising edge-triggered cell FDRE clocked by axis_clk  {rise@0.000ns fall@2.100ns period=4.200ns})
  Destination:            cal_count_reg[0]/CE
                            (rising edge-triggered cell FDRE clocked by axis_clk  {rise@0.000ns fall@2.100ns period=4.200ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            4.200ns  (axis_clk rise@4.200ns - axis_clk rise@0.000ns)
  Data Path Delay:        3.806ns  (logic 1.145ns (30.084%)  route 2.661ns (69.916%))
  Logic Levels:           4  (LUT3=1 LUT4=1 LUT5=2)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 6.328 - 4.200 )
    Source Clock Delay      (SCD):    2.456ns
    Clock Pessimism Removal (CPR):    0.184ns
  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)
                                                    0.000     0.000 r
                                                    0.000     0.000 r  axis_clk (IN)
                         net (fo=0)                 0.000     0.000    axis_clk
                                                                   r  axis_clk_IBUF_inst/I
                         IBUF (Prop_ibuf_I_O)       0.972     0.972 r  axis_clk_IBUF_inst/O
                         net (fo=1, unplaced)       0.800     1.771    axis_clk_IBUF
                                                                   r  axis_clk_IBUF_BUFG_inst/I
                         BUFG (Prop_bufg_I_O)       0.101     1.872 r  axis_clk_IBUF_BUFG_inst/O
                         net (fo=100, unplaced)     0.584     2.456    axis_clk_IBUF_BUFG
                         FDRE                                       r  cal_block_reg[0]/C
-------------------------------------------------------------------    -------------------
                         FDRE (Prop_fdre_C_Q)       0.478     2.934 r  cal_block_reg[0]/Q
                         net (fo=9, unplaced)       0.782     3.716    cal_block_reg_n_0_[0]
                                                                   r  count_tap[3]_i_5/I1
                         LUT4 (Prop_lut4_I1_O)      0.295     4.011 r  count_tap[3]_i_5/O
                         net (fo=1, unplaced)       0.449     4.460    count_tap[3]_i_5_n_0
                                                                   r  count_tap[3]_i_4/I4
                         LUT5 (Prop_lut5_I4_O)      0.124     4.584 r  count_tap[3]_i_4/O
                         net (fo=4, unplaced)       0.473     5.057    count_tap[3]_i_4_n_0
                                                                   r  cal_count[3]_i_3/I4
                         LUT5 (Prop_lut5_I4_O)      0.124     5.181 r  cal_count[3]_i_3/O
                         net (fo=2, unplaced)       0.460     5.641    cal_count
                                                                   r  cal_count[3]_i_1/I1
                         LUT3 (Prop_lut3_I1_O)      0.124     5.765 r  cal_count[3]_i_1/O
                         net (fo=4, unplaced)       0.497     6.262    cal_count[3]_i_1_n_0
                         FDRE                                       r  cal_count_reg[0]/CE
-------------------------------------------------------------------    -------------------
                         (clock axis_clk rise edge)
                                                    4.200     4.200 r
                                                    0.000     4.200 r  axis_clk (IN)
                         net (fo=0)                 0.000     4.200    axis_clk
                                                                   r  axis_clk_IBUF_inst/I
                         IBUF (Prop_ibuf_I_O)       0.838     5.038 r  axis_clk_IBUF_inst/O
                         net (fo=1, unplaced)       0.760     5.798    axis_clk_IBUF
                                                                   r  axis_clk_IBUF_BUFG_inst/I
                         BUFG (Prop_bufg_I_O)       0.091     5.889 r  axis_clk_IBUF_BUFG_inst/O
                         net (fo=100, unplaced)     0.439     6.328    axis_clk_IBUF_BUFG
                         FDRE                                       r  cal_count_reg[0]/C
                         clock pessimism            0.184     6.511
                         clock uncertainty         -0.035     6.476
                         FDRE (Setup_fdre_C_CE)    -0.202     6.274    cal_count_reg[0]
-------------------------------------------------------------------
                         required time                        6.274
                         arrival time                        -6.262
-------------------------------------------------------------------
                         slack                                0.012
```

Max Data Path Delay

Max Frequency 是依據 Synthesis 測出 Max path delay 後根據該 Delay 所設定的值，最終選定 Period = 4.2 並跑出 Slack 於 Setup time & Hold time 皆為正值，實驗期間有設定過 Period = 4.1 即無法通過，因此最大頻率約在 238MHz。

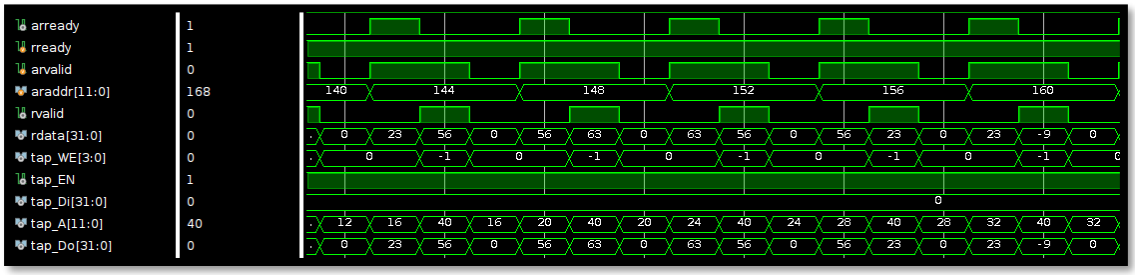
8. Simulation Waveform

Coefficient and Read Back



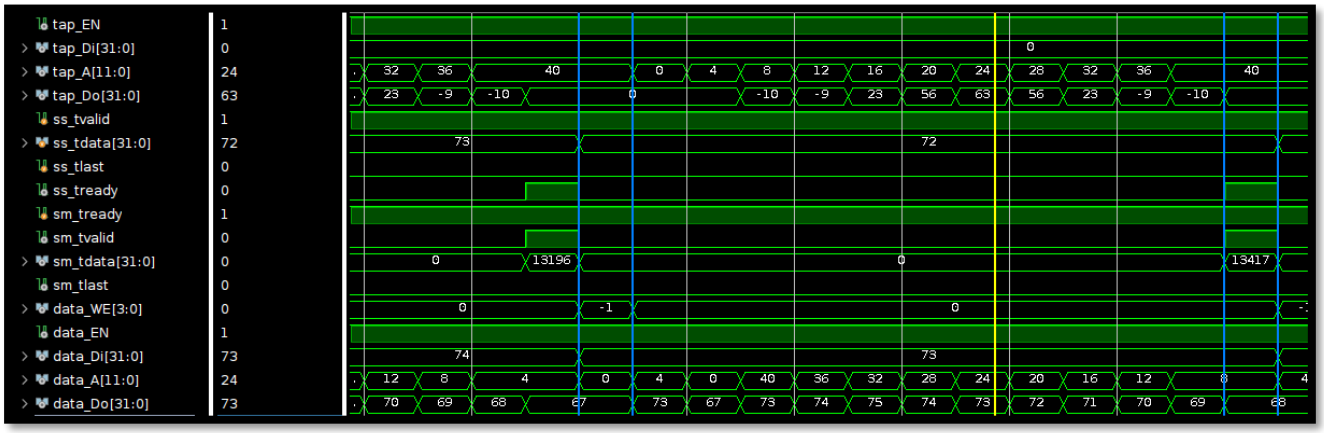
在 Ap_start = 1 之前，先將 Coefficient 值存入，因為有 awready & wready 與發送端進行 Handshake，當 awvalid & wvalid = 1 時，即會有 Addr & Data 透過 awaddr & wdata 傳輸過來。

此時在前方設計有提到 Tap Bram 採用的設計，讀取時間會延後一個 Clock，考量該部分只是簡易的將 Coefficient 傳入 Bram，因此我是切 2 個 States(報告後方 FSM 顯示)不斷重覆動作來接收完 11 筆資料。



該部分為將 Coefficient 值重新寫回 Testbench，並驗證其是否存在於正確的 Bram Block 中，亦採用 AXI4-lite 方式，資料會在 rvalid = 1 時由 rdata 同時送出，因此該部分只需在 Testbench 傳送對應的位址過來時，傳輸正確值回去即可。(因於 Slack 上有看到可將位址改成 0x80，故位址是從 0x80 開始+4)

Data In & Stream In & RAM Access Control



在左邊藍色線中間的 1 clock 為讀取該值進入 Block 而中間則是 11 clock 運算，與最後的 1 clock 輸出該值。

(1) Data Stream In

我採用的設計是 non-blocking 設計，因此 ss_tdata 值會在下一個 clock 才進入 data_Di，該部分詳細在 Data Stream Out 說明。

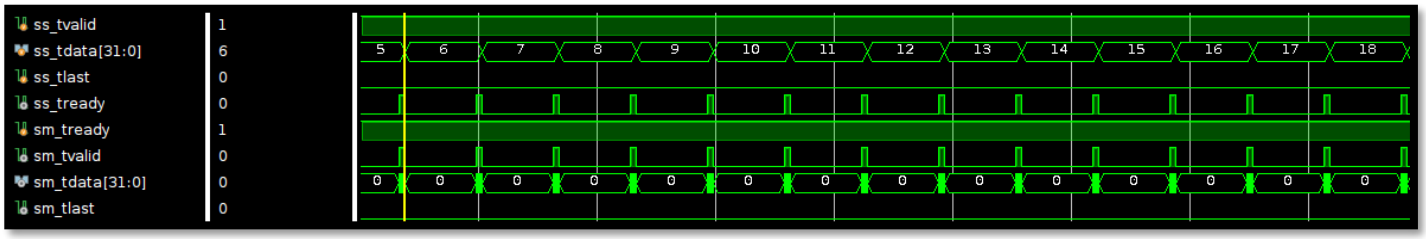
讀取時的 1 clock 可以看到 data_WE＝4'b1111、data_EN＝1，即可將目前在 data_Di 上的值存入目前所打開的暫存器位址(data_A)中，而當該 clock 過後因為沒有寫入資料的需求，故將 data_WE＝4'b0000。

(2) RAM Access Control

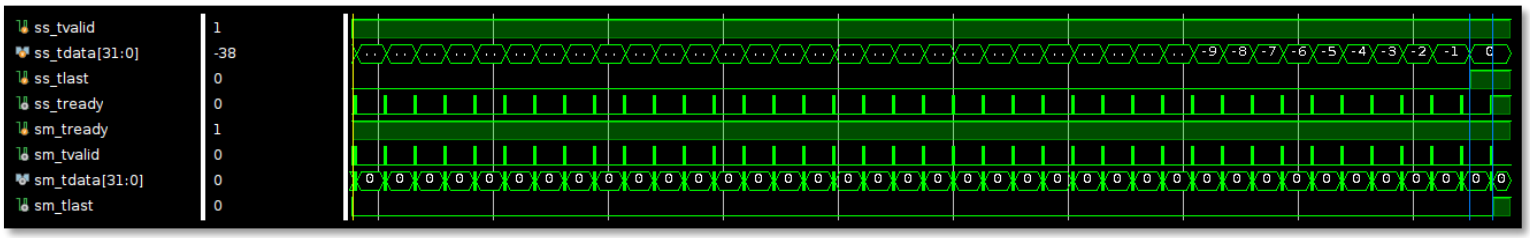
中間可以看到 tap_Do 輸出的值與 data_Do 的值在同一個 clock 上對應，其對應的值會互相相乘並輸出到暫存器中，在下一個 clock 執行乘法運算時，會加上一筆的值，而該值即是剛剛存入暫存器的返還值，依此類推執行 11 clock 後，即可以輸出該運算值。

(3) Data Stream Out

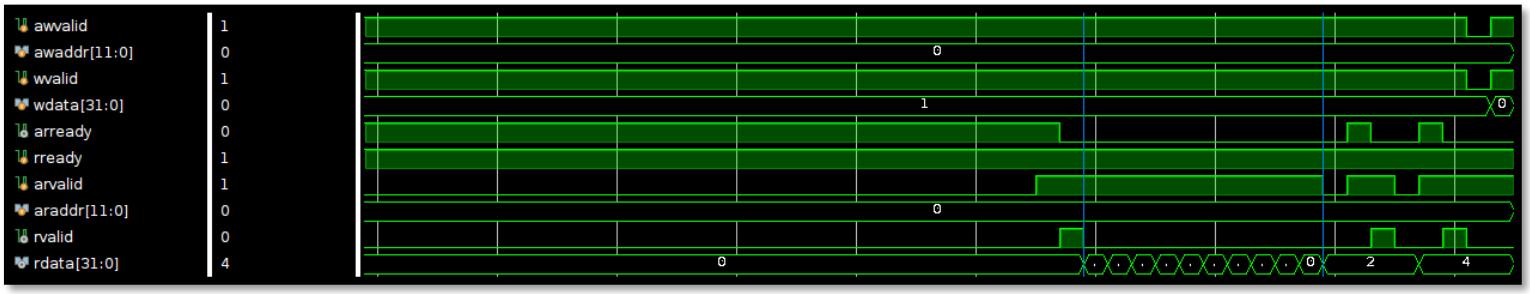
最後上方圖中藍色線框起來的 1 clock 為輸出資料，此時 sm_tvalid 也要隨即設 1 搭配 sm_tdata 於同一個 clock 進行輸出完成 AXI4-Stream 的規範。此時補充 Data Stream In 說明，在同時為了節省 1 clock 的浪費，因此我也將 ss_tready 在同時也等於 1，使其 ss_tdata 輸出下一筆資料，達到在 1 個 clock 中做兩件事情。(下方圖為顯示 AXI4-Stream port)



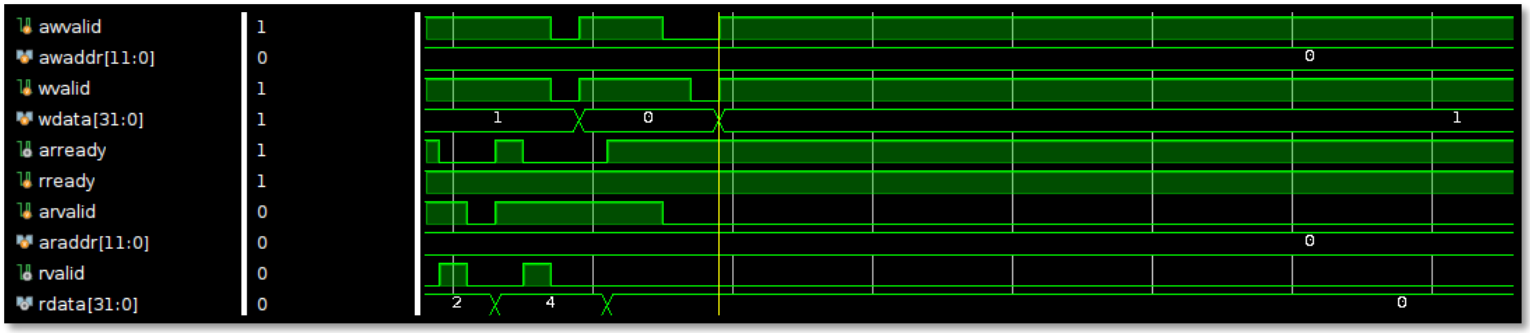
AXI4_Stream & AXI4_Lite Last Signal



此處可以看到 ss_tlast 會優先跳起表示進入的為最後一筆 Data，而等待該筆 Data 運算結束之後 sm_tlast 也會伴隨著結果的輸出而一起跳起。

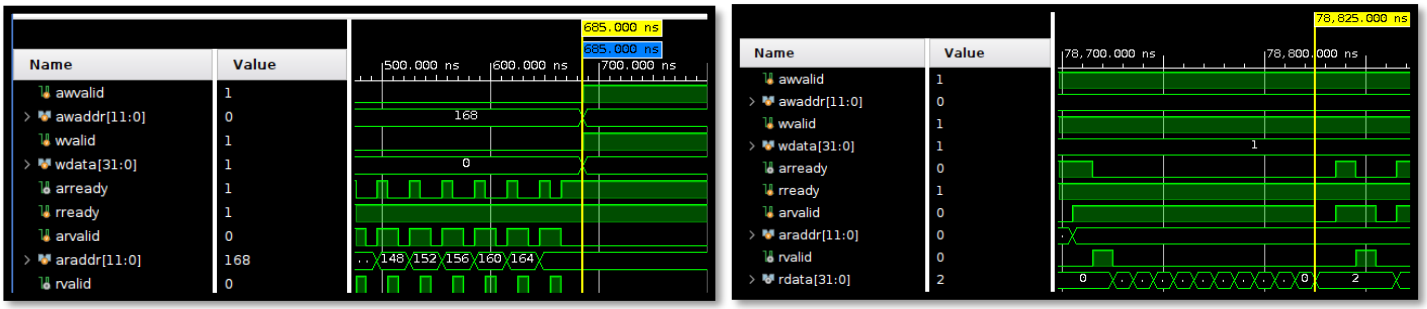


等待兩個 tlast 都跳起來後，Ap_done 就要被開啟，以及 Ap_idle 因為 Ap_done 的跳起也跟著被拉高，在圖中可以看到該值呈現，也可確認 Ap 訊號是以 AXI4-Lite 的方式來傳輸，最終 Testbench 端會因為 Ap_idle = 1，輸出一個 Ap_start = 0 訊號來關閉整個 FIR Module。



直到下個 600 筆資料欲執行，Ap_Start 才會再次被開啟，本次 Testbench 共傳輸「三次」600 筆資料進入處理，以確保該 FIR Module 在開始、執行、結束的 AXI4-Lite、AXI4-Stream 皆無問題。

9. Clock Cycle from Ap_start to Ap_done



時間由 685ns 至 78825ns 總共耗費了 78140ns，本次設定一個 clock 為 10ns，故處理一筆完整資料所需「7814 個 clock」，其中包含 600 筆 Data 每筆 13 clock，其餘為進行初始化所消耗。

10. FSM