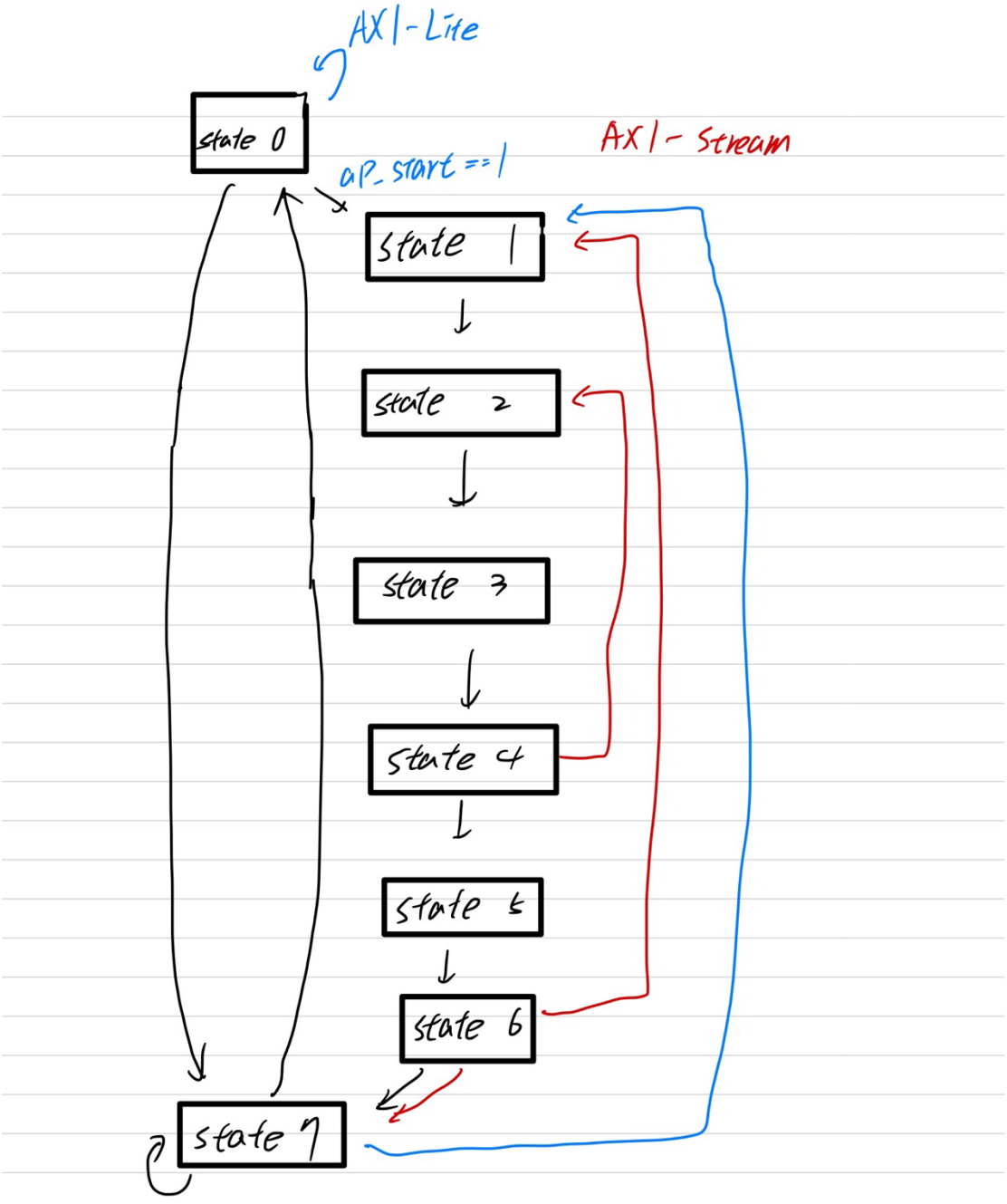
111061589 紀皓洋 LAB3

Block diagram

我全部使用ＦＳＭ的設計，



Stage 0 會將所有暫存器的值歸零，同時利用 AXI Lite 協議將所需資料傳送進來，並直接存到 BRAM 去。

因為這裡是傳 11 個 tap 值進來，會剛剛好存滿 11。而其他像是 data_length 或是 ap_start，我使用 reg 暫存他們代表的意義。

每一次完成 stage 0 會有三種選擇，可以選擇繼續接收 tap num，或是當 AP_start 升起時，開始 AXI Stream 的傳輸，或是進行 check 。

先說 Stage 7 ，這個階段會做一些地址判定，

如果是 ap 的地址，那麼根據是否做完 600 個 data 會有不同的值，對應的意義分別是 ap_done , ap_valid。如果做完 ap_done 會升起，如果ＦＩＲ停止運算，ap_valid 升起。

如果不是 ap 的地址，那便會從ＢＲＡＭ中提取 tap，並透過 AXI lite 來傳送值回去給 testbench，目的是做 check 的動作。

Stage 1 代表 Data 透過 AXI Stream 傳送到ＦＩＲ中，再將其存到ＢＲＡＭ中。與此同時會將計數計加一，方便我觀察目前進行到第幾個資料。

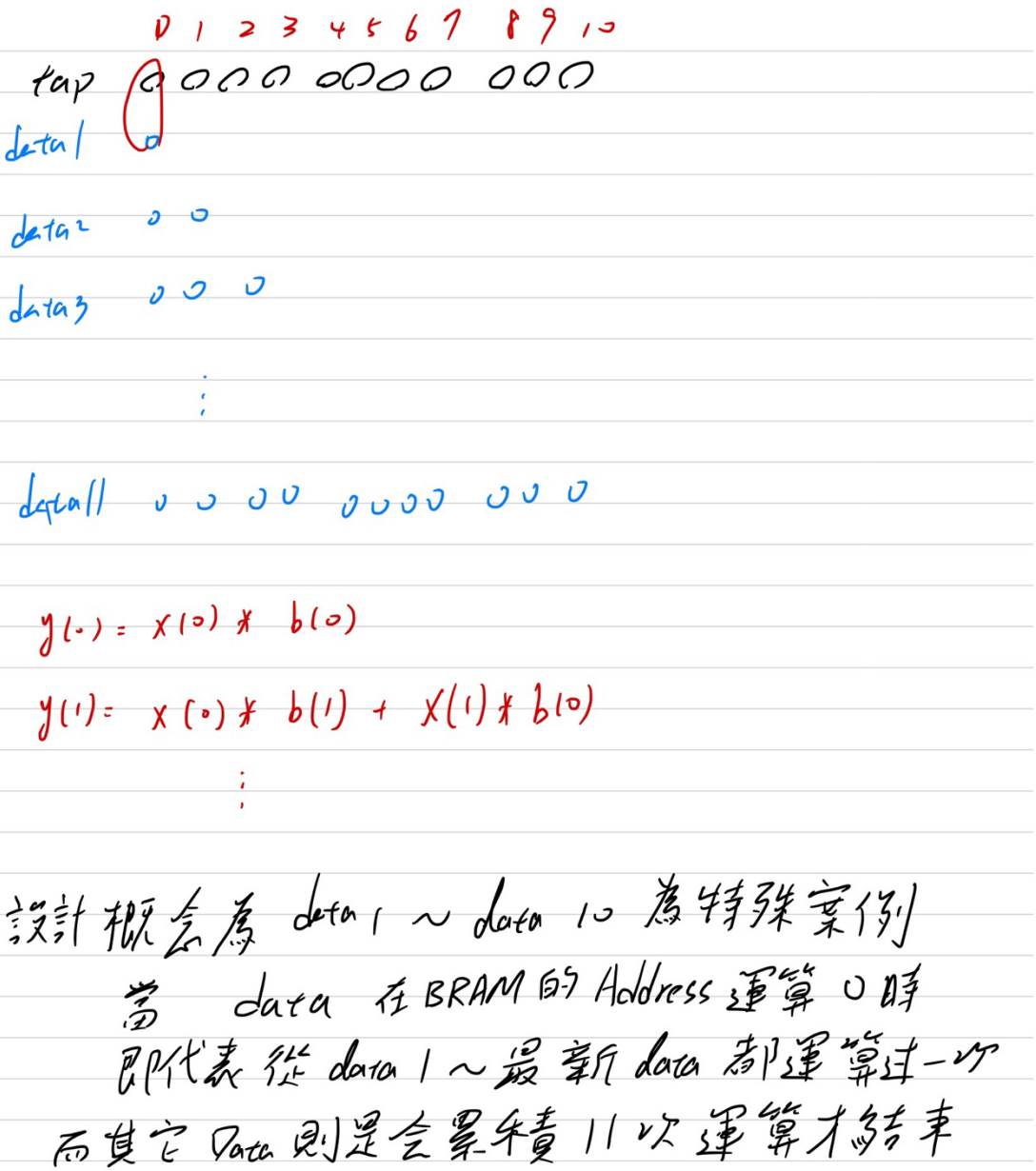
另外這裡也代表新的一筆資料進來，所以累加器等需要歸零的暫存器都會在這粒進行歸零與初始化。

同時也對下一次新 DATA 的ＢＲＡＭ地址作更新。

Stage 2 就是很簡單地從ＢＲＡＭ中提取值出來

Stage 3 就是將提取出來的值做運算。

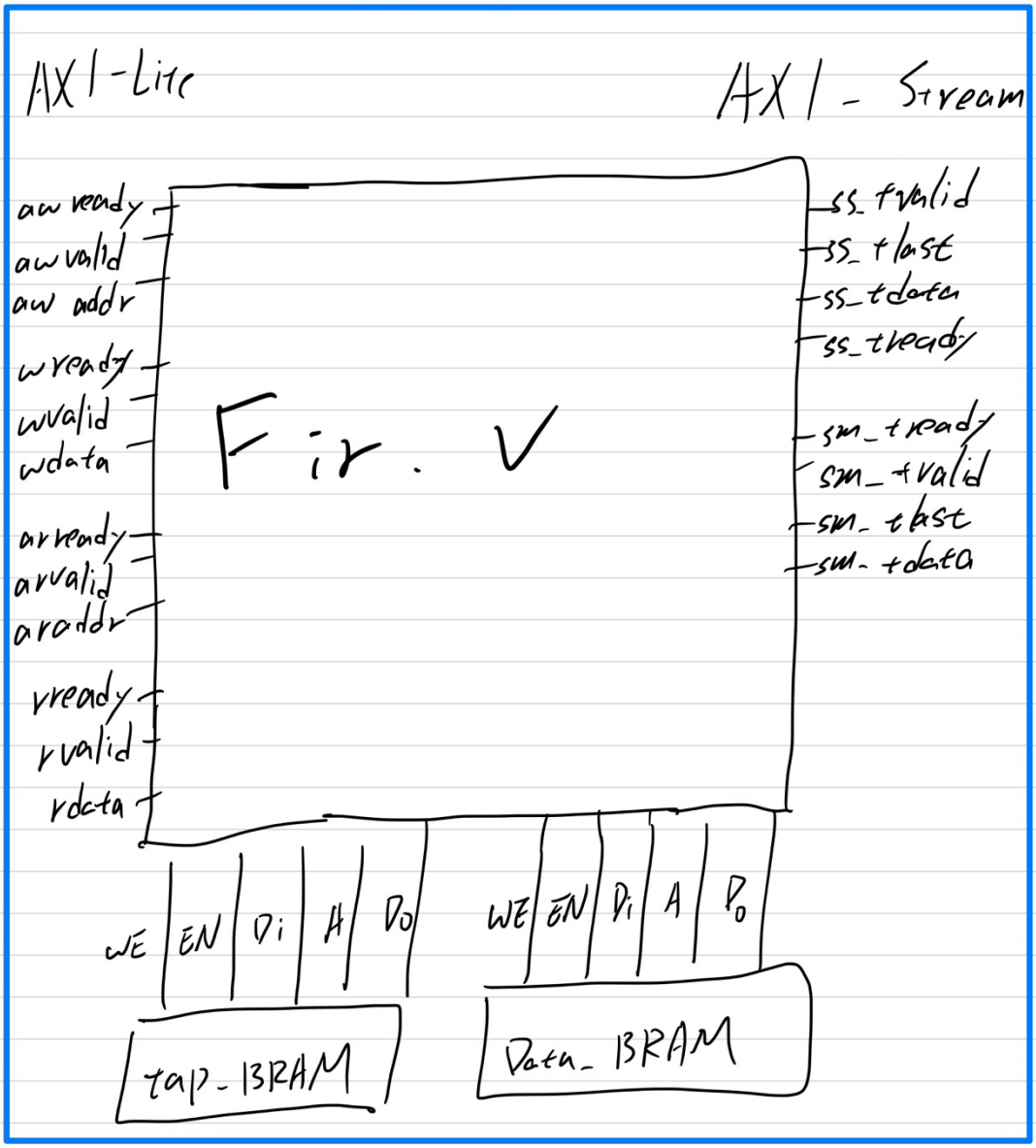
Stage 4 會判斷是屬於前十筆資料還是其他需要做 11 次資料存取，會做這個判斷跟我的設計概念有關。另外他會再是否計算已結束，還是要繼續運算。



Stage 5 就是當運算結束後，將 Data 送回 testbench。

Stage 6 就是判斷是否還有資料需要運算，如果有，那就回 Stage1，若沒有就 Stage 7 等待測試。另外一種情況是因為 testbench 想要測試 AP_valid 的狀態，因此當還沒運算結束但卻想測試時，我有再拉一條線去 Stage 7 等待測試，阿如果測完，發現還沒運算完，就會再回去 Stage 1 繼續新一輪的運算。

Testbench.

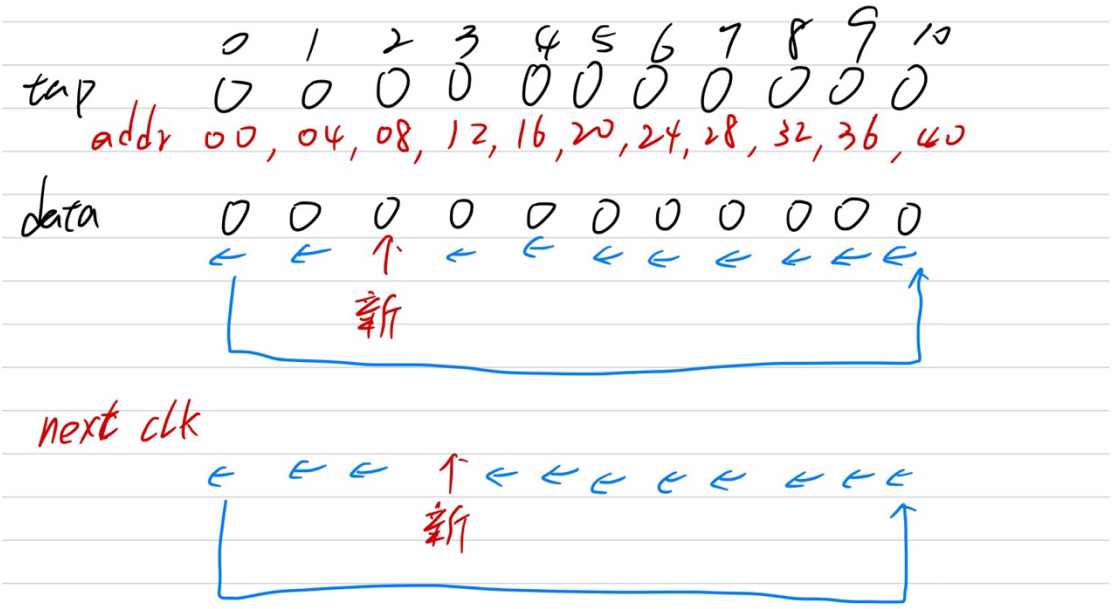

AXI-Lite
AXI-Stream
awready
awvalid
awaddr
wready
wvalid
wdata
arready
arvalid
araddr
rready
rvalid
rdata
Fir.v
ss_tvalid
ss_tlast
ss_tdata
ss_tready
sm_tready
sm_tvalid
sm_tlast
sm_tdata
WE
EN
Di
A
Do
WE
EN
Di
A
Do
tap_BRAM
Data_BRAM

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

AXI-Lite 我會做 handshake，當 valid 為 1，將 ready 拉起，此時已完成 handshake，接著接受 data 的同時，直接將其存入 BRAM。但如果不是存 tap，那我就不會存入 BRAM，因為也用不到。

AXI-Stream，其概念應該是完成一次 handshake 就可以不停地送值，達到類似 Burst 的效果，但因為我的一次計算就需要大量的時間，根本用不到他的優點，因此我還是會對每一次 data 進來時做一次 handshake。送值出去也會再做一次 handshake。

How to access shiftram and tapRAM to do computation



當有新 Data 送進 BRAM，最舊將被取代
同時它的地址將被記住.
由它開始運行1圈

而 tap 保持位置不變，因此每次地址.
都從同一位置開始

但正如前面所提到的特殊前十筆資料，我會做另外的處理，以 data 的地址作為判斷依據。

How ap_done is generated

在 stage 7 都會判斷是否做完，若做完就會將 ap_done 拉起，若沒做完就不會。

Slack

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.879 ns	Worst Hold Slack (WHS):	0.144 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	42	Total Number of Endpoints:	42	Total Number of Endpoints:	82

All user specified timing constraints are met.

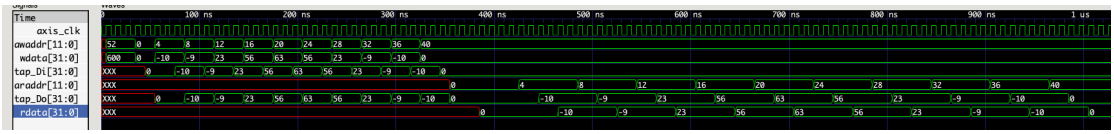
Timing Summery

```
Design Timing Summary
---------------------

    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints      WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints     WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
    -------      -------  ---------------------  -------------------      -------      -------  ---------------------  -------------------     --------     --------  ---------------------  --------------------
      6.879        0.000                      0                   42        0.144        0.000                      0                   42        4.500        0.000                      0                    82


All user specified timing constraints are met.
```

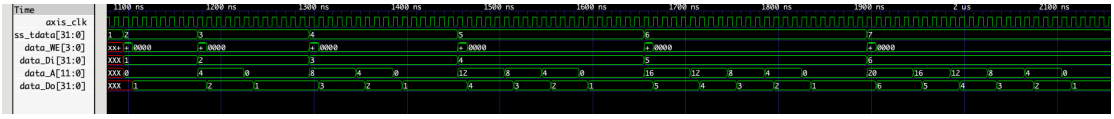
Max Delay Path

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             6.879ns  (required time - arrival time)
  Source:                 cstate_reg[0]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            data_EN_reg/S
                            (rising edge-triggered cell FDSE clocked by axis_clk  {rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            10.000ns  (axis_clk rise@10.000ns - axis_clk rise@0.000ns)
  Data Path Delay:        2.384ns  (logic 0.773ns (32.425%)  route 1.611ns (67.576%))
  Logic Levels:           1  (LUT3=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 12.128 - 10.000 )
    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
```

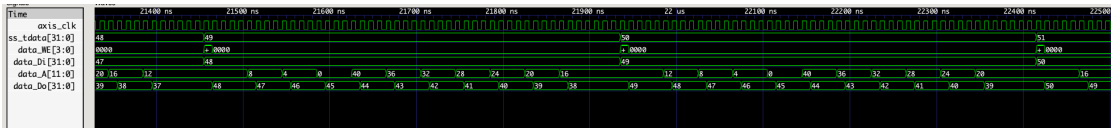
Read the tap from testbench and write it into BRAM



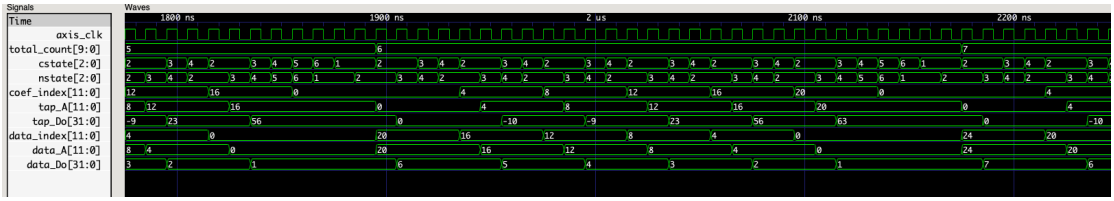
Read from BRAM and send back to testbench



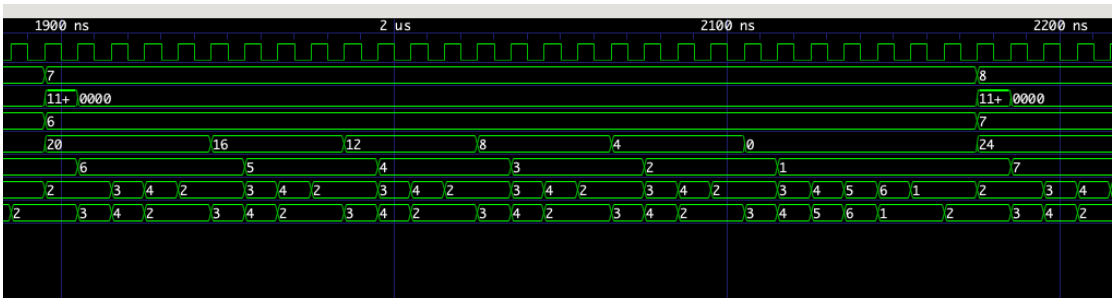
Write the new data from testbench into BRAM



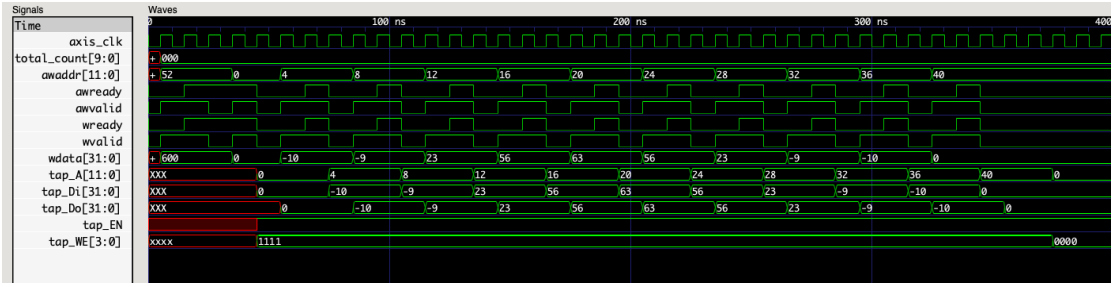
The data and tap read from BRAM



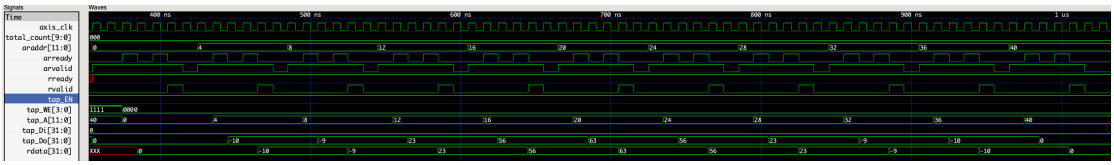
The data and tap read from BRAM but for the first 10 data



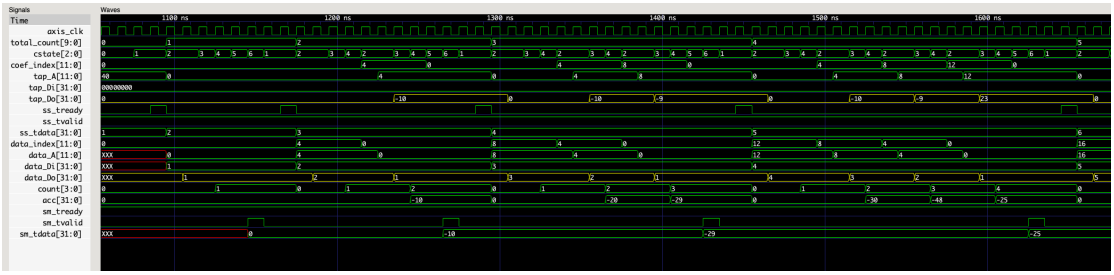
AXI-Lite write Handshake



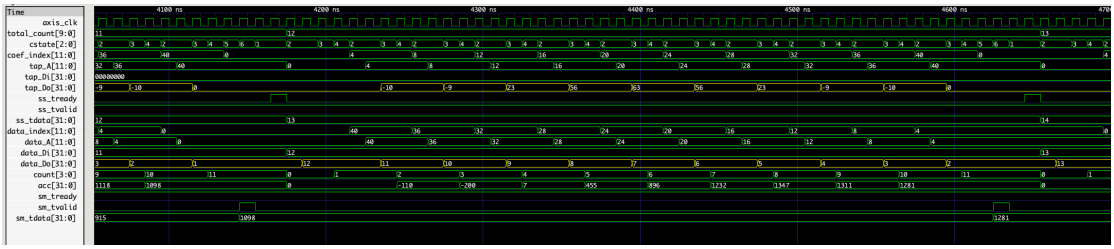
AXI-Lite read Handshake



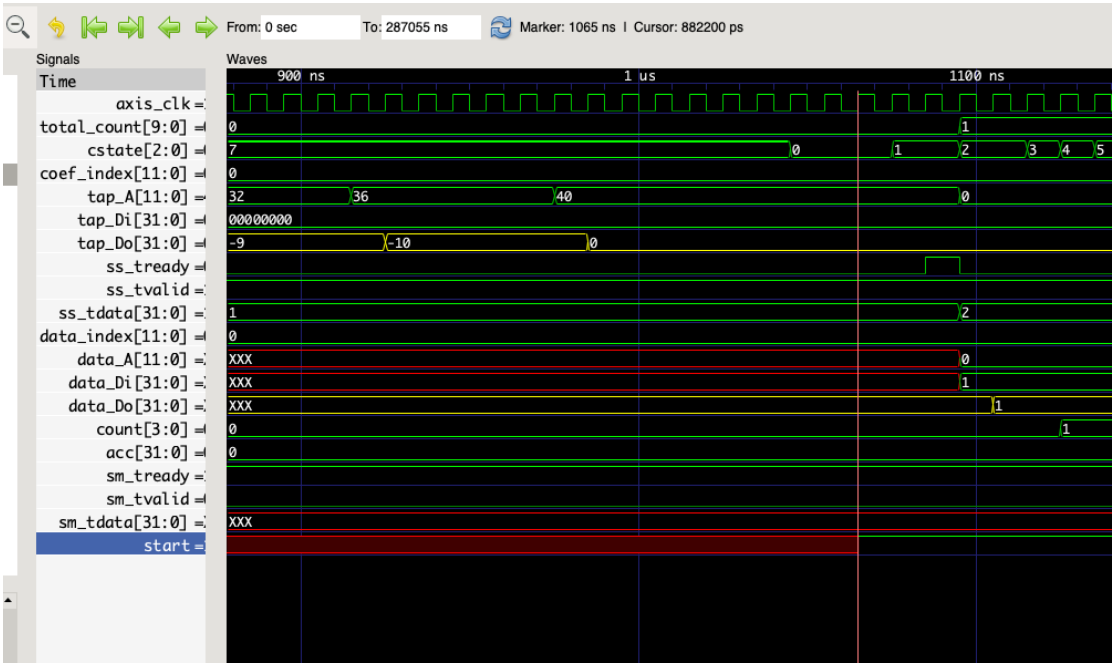
Calculate for the acc with first 10 data



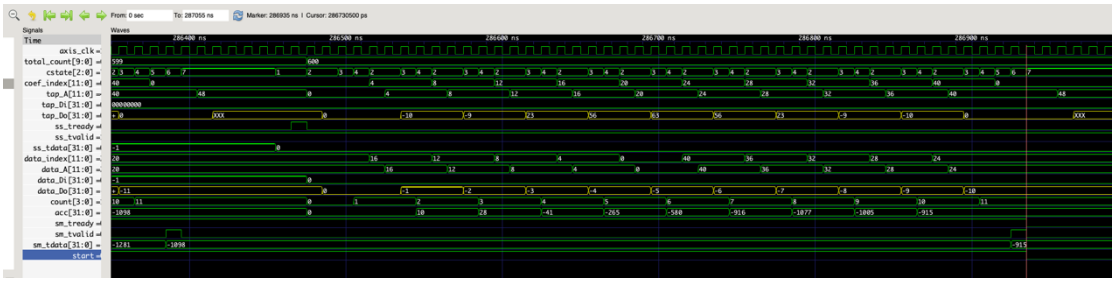
Calculate for the acc with other data



Ap_start



AP_done



心得：

此作業的第一步是先了解 AXI-Lite 與 AXI-Stream 的協議，對於 handshake 的概念也是一個新的學習。

第二步是了解 BRAM 的概念，並如何去運用與接線。是一個能在同一個 CLK 給他 input 又輸出 output 的記憶體，很酷。

第三步是了解如何運用這僅有的 11 words 記憶體來儲存 data 與 tap，取值並運算，對於 address 的存取與轉換，我認為關鍵在於我用了三個 reg 來存取地址，第一個是存去新 data 的地址，另外兩個分別儲存 data 運算時的地址與 tap 運算時的地址。

第四步就是對於 ap 這個東西該如何去接收與判定。

我一開始並沒有使用ＦＳＭ的設計，因此狀態寫得很亂，但還是有寫出來，但因為繳交期限有延後，因此我有再寫一個 FSM 的版本，我自己也比較滿意 FSM 的版本，寫出來比較清楚，對於整個流程也更加清晰。

遲交原因：

我是因為看到繳交期限有修改，所以才延遲繳交，如果事先有說明清楚，我會在昨天繳交。當然，學校停電也有影響到一部分，因為我都是在學校的電腦寫作業，但我是因為看到繳交期限有延期，所以才沒有及時繳交，如果只要在繳交時間前完成作業繳交，我想我都會不停地去嘗試更好的設計，即便最後並沒有獲得更好的成績。

希望助教能通融並理解我們，停電與公告都是我遲交的原因，謝謝。