SoC Lab Lab3 Report

R10522526 羅崇榮

1. System Architecture

在本實驗中我設計的 FIR 系統，其架構遵循講義的規定，並未使用額外的 register file 儲存 tap 與 data 的值。所有乘法計算所需的 data 與 coefficient 皆由 bram 中取得，每個 cycle 取一次值進行計算。

其系統架構圖如圖 1-1 所示，bram 則是選擇 bram11。在此架構中唯一用來儲存資料的 register 為 acc，而對於 AXI-lite 與 AXI-stream 以及 tap_bram 與 data_bram 的 I/O 訊號分別設計相應的邏輯。

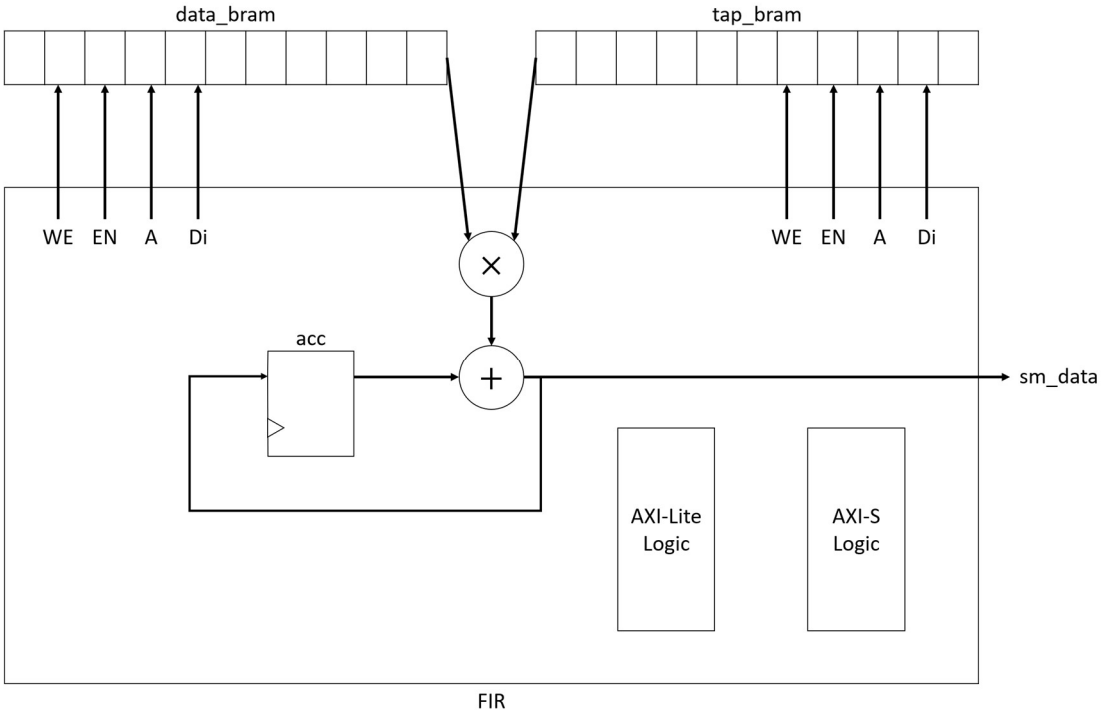

圖 1-1

2. Dataflow

FIR 計算乘法與累加的步驟如下，示意圖如圖 2-1。

A. tap_A、data_ptr、ptr 皆從位址 0 初始化。

B. 每過一個 cycle，tap_A 和 ptr 的值各自增加 1。

C. 當 tap_A 與 ptr 皆指到位址 10 時，準備輸出 acc 的結果。

D. 隨後的 cycle，tap_A 回到位址 0，data_ptr 的值增加 1，ptr 則指向前一個 data_ptr(即目前要算的 11 筆 data 的起始點)。

整體的計算流程圖則如圖 2-2 所示，每次計算一筆輸出值須 11 個 cycle。

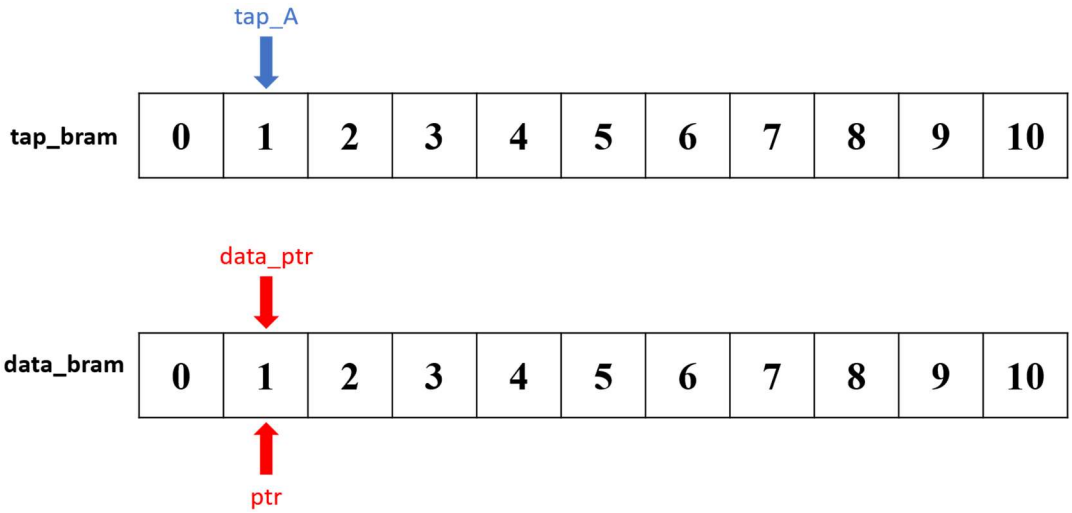

圖 2-1-A

圖 2-1-B

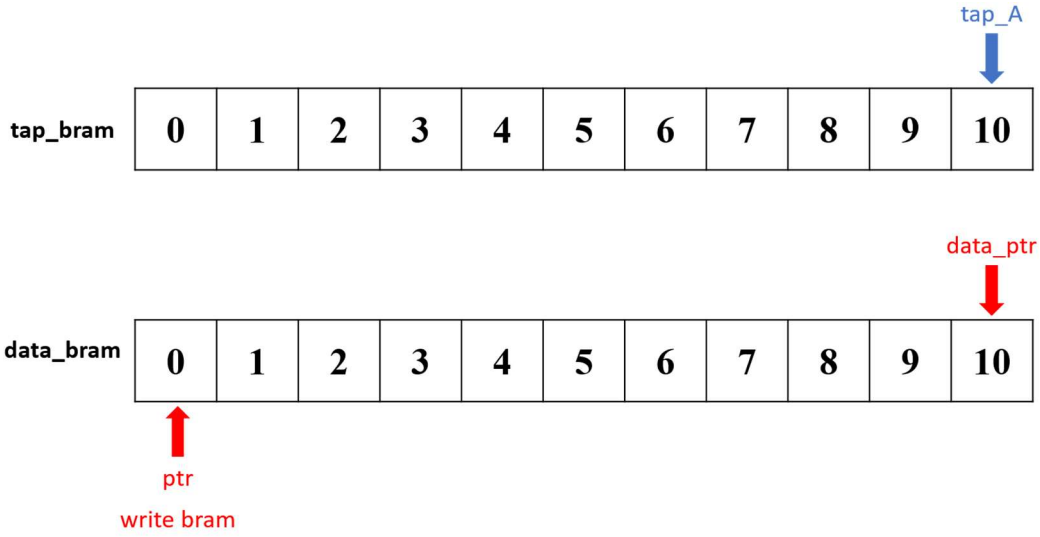

圖 2-1-C

圖 2-1-D

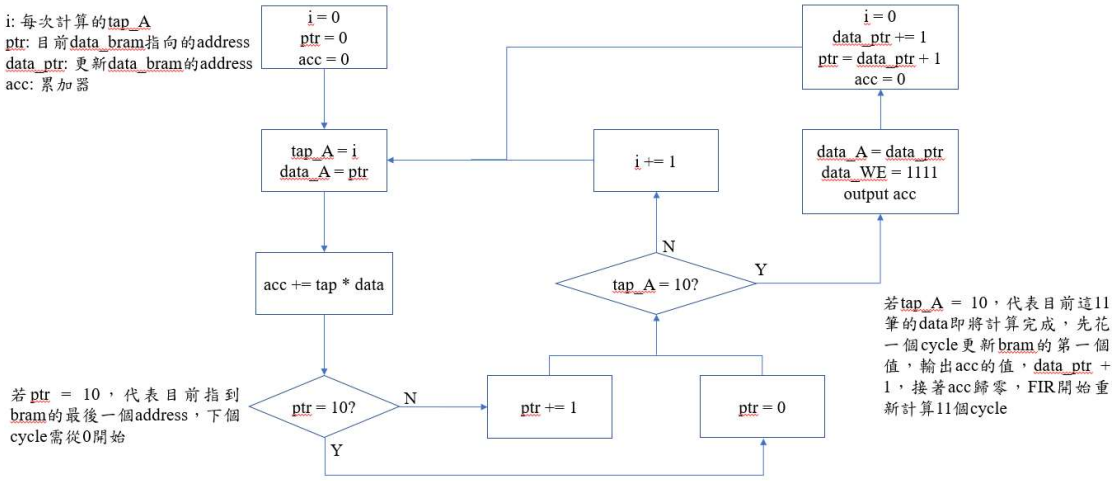

圖 2-2

3. I/O Protocol

•AXI-lite

在 tap 的 Write 操作中，當 awready 與 awvalid 同時為 High 時，tap_bram 會收到預計寫入的地址，而當 wready 與 wvalid 同時為 High 時，data 會在下一個 cycle 寫入 bram。由於我在 FIR 系統中為 output 設計了 output register，因此資料的寫入會延遲一個 cycle，如圖 3-1。

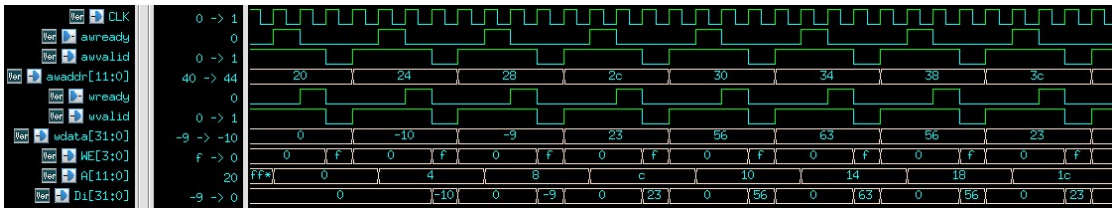

圖 3-1

另一方面，Read 操作也是遵循相同的 protocol，當 ready 與 valid 同時為 High 時傳地址/讀資料，如圖 3-2。

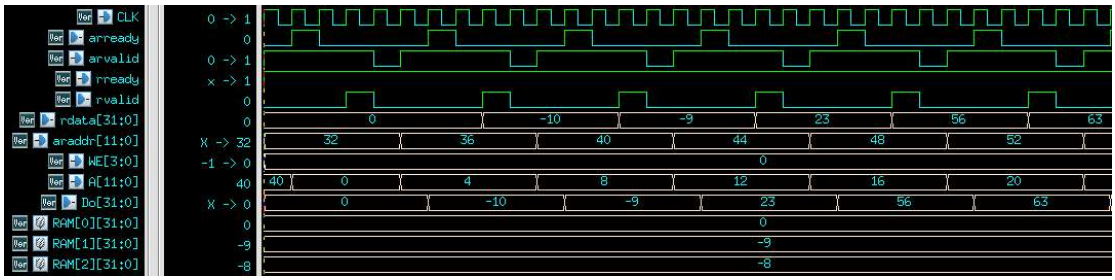

圖 3-2

•AXI-stream

data 的 I/O 是採用串流的形式，即當 ss_tready 與 ss_tvalid 同時為 High，則 data 就會持續傳至 data_bram 中，而當 sm_tready 與 sm_tvalid 同時為 High，累加器的值會持續輸出，如圖 3-3。

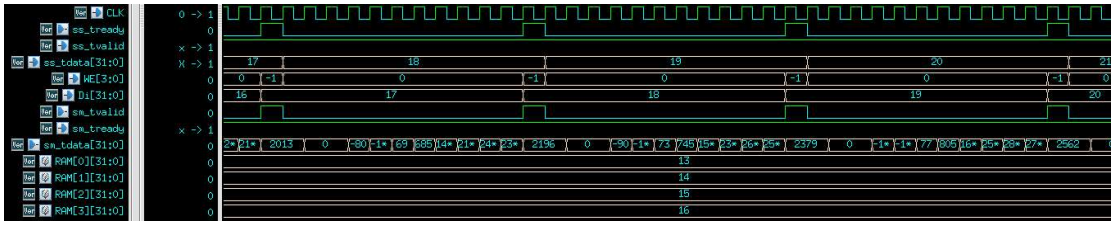

圖 3-3

4. Resource Usage

本次 lab 設計的 FIR 所消耗的資源如圖 4-1。整個系統的控制和計算邏輯是透過 lookup table 執行。Register 一共消耗了 195 個，我認為這個數字相對較大的原因是，我在 FIR 中的每一個 output 都有設 output register，一方面是為了減少 timing violation 的發生，另一方面則讓我的設計更加模組化。

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	203	0	0	53200	0.38
LUT as Logic	203	0	0	53200	0.38
LUT as Memory	0	0	0	17400	0.00
Slice Registers	195	0	0	106400	0.18
Register as Flip Flop	195	0	0	106400	0.18
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

圖 4-1

在合成中有使用到 DSP 的 IP，如圖 4-2，我推測這些 IP 是用在乘法與累加的運算上。

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	3	0	0	220	1.36
DSP48E1 only	3				

圖 4-2

5. Timing Summary

本次 FIR 的工作頻率設定為 4.5ns 進行合成，其 timing summary 如圖 5-1，可看出沒有出現 violation 的情形。

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.007 ns	Worst Hold Slack (WHS):	0.137 ns	Worst Pulse Width Slack (WPWS):	1.750 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:	260	Total Number of Endpoints:	260	Total Number of Endpoints:	196

All user specified timing constraints are met.

圖 5-1

最長的 timing path 如圖 5-2 所示，為 tap_A 的更新，我推測 critical path 並非乘法運算的原因有二:

1. tap_A 本身更新的邏輯就較為複雜，涵蓋了 AXI-lite 與乘法運算，特定情況也需要 stall，使合成時需要使用更多的邏輯閘。
2. 在 Synthesis report 中可觀察到合成時有使用到 DSP 的 IP。查閱 XILINX 的官方文件時，我看到 FPGA 板中 DSP 的 IP 是有經過高度優化的，因此其乘法的速度應該會很快。

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.007ns  (required time - arrival time)
  Source:                 tap_A_r_reg[2]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@2.250ns period=4.500ns})
  Destination:            tap_A_r_reg[10]/D
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@2.250ns period=4.500ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            4.500ns  (axis_clk rise@4.500ns - axis_clk rise@0.000ns)
  Data Path Delay:        4.357ns  (logic 1.293ns (29.676%)  route 3.064ns (70.324%))
  Logic Levels:           5  (LUT4=1 LUT5=2 LUT6=2)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 6.628 - 4.500 )
    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
```

圖 5-2

clock cycle: 4.5ns

總花費 cycle:從 ap_start 開始算到 ap_done: 7219