112-1 Soc Design Laboratory

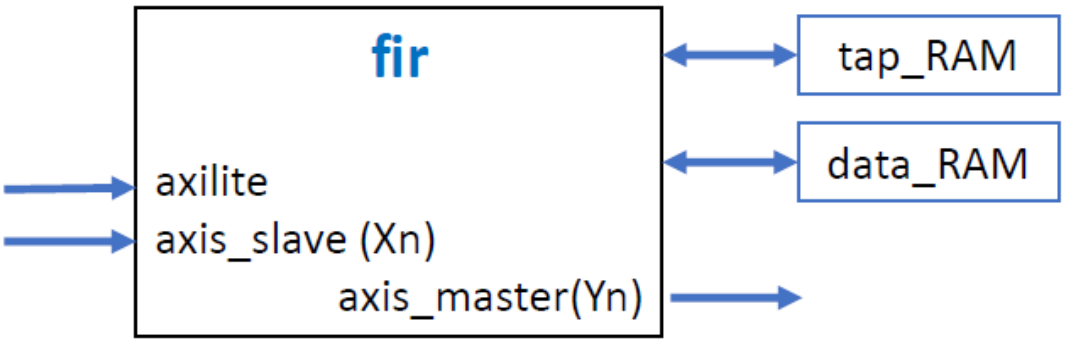
Lab3

學號:112061619

姓名:王證皓

1. Block Diagram

Function: $y[t] = \Sigma (h[i] * x[t - i])$



(1)AXI-Lite(h[i])

```
    output  wire                     awready,
    output  wire                     wready,

    input   wire                     awvalid,
    input   wire [(pADDR_WIDTH-1):0] awaddr,
    input   wire                     wvalid,
    input   wire [(pDATA_WIDTH-1):0] wdata,
//w
    output  wire                     arready,

    input   wire                     rready,
    input   wire                     arvalid,
    input   wire [(pADDR_WIDTH-1):0] araddr,

    output  wire                     rvalid,
    output  reg  [(pDATA_WIDTH-1):0] rdata,
//R
```

(2)AXI-Slave(X[n])

```
    input   wire                     ss_tvalid,
    input   wire [(pDATA_WIDTH-1):0] ss_tdata,
    input   wire                     ss_tlast,

    output  wire                     ss_tready,
//slave
```

(3)AXI-Master(Y[n])

```
    input   wire                     sm_tready,

    output  reg                      sm_tvalid,
    output  wire [(pDATA_WIDTH-1):0] sm_tdata,
    output  wire                     sm_tlast,
//master
```

(4)Tap_RAM & Data_RAM

```
// bram for tap RAM
output  wire [3:0]               tap_WE,
output  wire                     tap_EN,
output  wire  [(pDATA_WIDTH-1):0] tap_Di,
output  wire  [(pADDR_WIDTH-1):0] tap_A,
input   wire  [(pDATA_WIDTH-1):0] tap_Do,

// bram for data RAM
output  reg  [3:0]               data_WE,
output  wire                     data_EN,
output  wire [(pDATA_WIDTH-1):0] data_Di,
output  wire [(pADDR_WIDTH-1):0] data_A,
input   wire [(pDATA_WIDTH-1):0] data_Do,

input   wire                     axis_clk,
input   wire                     axis_rst_n
```

2. Operation

(1) Tap Parameter in

Tap parameter 會透過 AXI-Lite 來輸入，並存放到 Tap_RAM 裡面，AXI-Lite 由 wvalid 與 wready 的 handshake 來做 Data 的傳遞，address 也為相同原理，且須確定 Tap parameter 的 address 為 0x20-FF，此時 tap_We 與 tap_EN 皆須拉起，並從 RAM 的位址 0 從小到大依序存放到位址 44。

(2) Access Tap Parameter

要存取 Tap parameter 時，tap_WE 須關閉且 tap_EN 需打開，並依序輸入位址即可存取數值。

(3) Ap_ctrl

當存完 tap parameter 後，會輸入位址 0x00 並在資料的第 0 個 bit 拉起，也就是 ap_start 會啟動 FIR engine，此時 ap_idle 會變為 0，並

在運算結束後拉起 ap_done。

(4) AXI-Slave & AXI-Master

在 ap_start 啟動後，會透過 AXI-Slave 傳輸 X 的數值，即 ss_tready 會拉起以接收資料，並在運算結束後拉起 sm_tvalid 以輸出結果，並在最後一筆輸入輸出資料分別拉起 ss_tlast 與 sm_tlast。

(5) Data-RAM

此處以 bram 實現 shift ram 的效果，首先會將 ss_tready 拉起後的下一個 cycle 將 data_WE 打開，並存放到最底層(位址 44)，下一次輸入會存到位址 40，依此類推，直到存到位址 0 時，下一筆資料會從位址 44 繼續存放，當要存取時，會由 counter 控制位址從小到大存取。

3. Resource usage

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	223	0	0	53200	0.42
LUT as Logic	223	0	0	53200	0.42
LUT as Memory	0	0	0	17400	0.00
Slice Registers	276	0	0	106400	0.26
Register as Flip Flop	124	0	0	106400	0.12
Register as Latch	152	0	0	106400	0.14
F7 Muxes	0	0	0	26600	0.00
F8 Muxes	0	0	0	13300	0.00

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	140	0.00
RAMB36/FIFO*	0	0	0	140	0.00
RAMB18	0	0	0	280	0.00

FF:124
LUT:223
BRAM:0

4. Timing Report

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

Clock     Waveform(ns)       Period(ns)      Frequency(MHz)
-----     ------------       ----------      --------------
axis_clk  {0.000 7.425}      14.850          67.340


------------------------------------------------------------------------------------------------
| Intra Clock Table
| -----------------
------------------------------------------------------------------------------------------------

Clock             WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints     WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
-----             -------      -------  ---------------------  -------------------     --------     --------  ----------------------  --------------------
axis_clk            0.003        0.000                      0                   254        6.925        0.000                       0                   125
```

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.003ns  (required time - arrival time)
  Source:                 j_cnt_reg[4]/C
                            (rising edge-triggered cell FDRE clocked by axis_clk  {rise@0.000ns fall@7.425ns period=14.850ns})
  Destination:            temp_reg[29]/D
                            (rising edge-triggered cell FDRE clocked by axis_clk  {rise@0.000ns fall@7.425ns period=14.850ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            14.850ns  (axis_clk rise@14.850ns - axis_clk rise@0.000ns)
  Data Path Delay:        14.742ns  (logic 10.453ns (70.904%)  route 4.289ns (29.096%))
  Logic Levels:           15  (CARRY4=9 DSP48E1=2 LUT2=2 LUT4=1 LUT6=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 16.978 - 14.850 )
    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
```

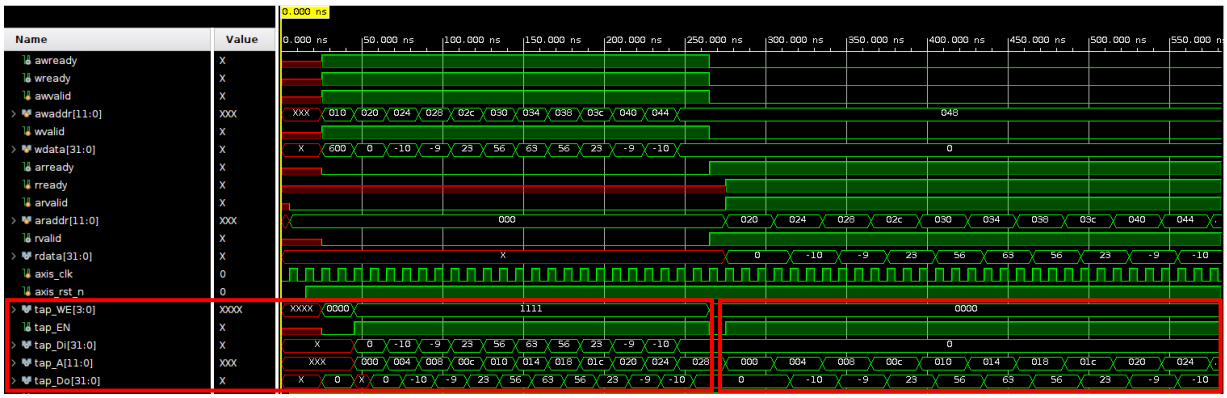
Maximum frequency 約為 67.34MHz

5. Simulation Waveform

(1)testbench 模擬結果

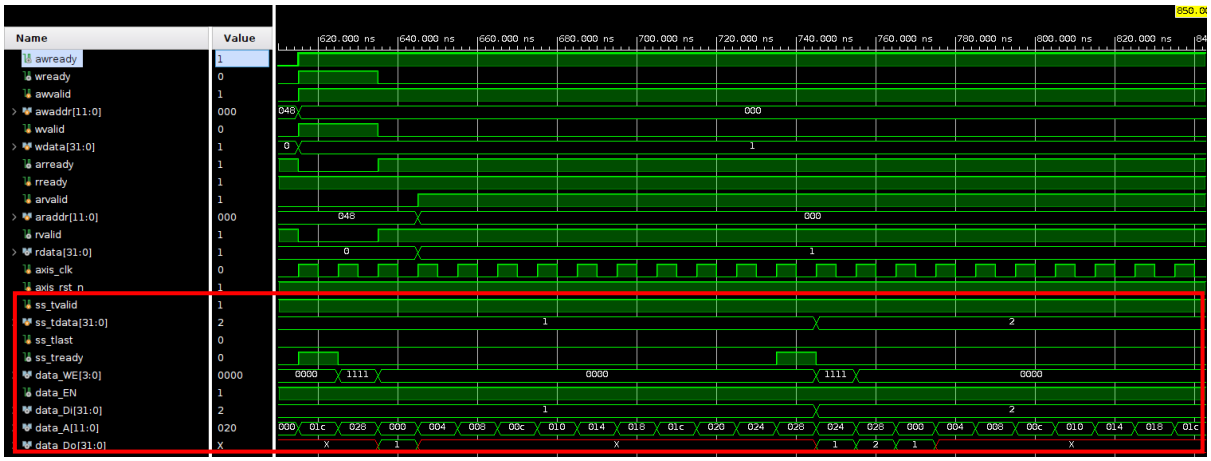
```
------------Start simulation-----------
----Start the data input(AXI-Stream)----
----Start the coefficient input(AXI-lite)----
 Check Coefficient ...
OK: exp =           0, rdata =           0
OK: exp =         -10, rdata =         -10
OK: exp =          -9, rdata =          -9
OK: exp =          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
OK: exp =           1, rdata =           1
----End the coefficient input(AXI-lite)----
...
[PASS] [Pattern       580] Golden answer:       -4392, Your answer:       -4392
[PASS] [Pattern       581] Golden answer:       -4209, Your answer:       -4209
[PASS] [Pattern       582] Golden answer:       -4026, Your answer:       -4026
[PASS] [Pattern       583] Golden answer:       -3843, Your answer:       -3843
[PASS] [Pattern       584] Golden answer:       -3660, Your answer:       -3660
[PASS] [Pattern       585] Golden answer:       -3477, Your answer:       -3477
[PASS] [Pattern       586] Golden answer:       -3294, Your answer:       -3294
[PASS] [Pattern       587] Golden answer:       -3111, Your answer:       -3111
[PASS] [Pattern       588] Golden answer:       -2928, Your answer:       -2928
[PASS] [Pattern       589] Golden answer:       -2745, Your answer:       -2745
[PASS] [Pattern       590] Golden answer:       -2562, Your answer:       -2562
[PASS] [Pattern       591] Golden answer:       -2379, Your answer:       -2379
[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
[PASS] [Pattern       598] Golden answer:       -1098, Your answer:       -1098
[PASS] [Pattern       599] Golden answer:        -915, Your answer:        -915
------End the data input(AXI-Stream)------
OK: exp =           2, rdata =           7
OK: exp =           4, rdata =           7
-----------------------------------------------
-----------Congratulations! Pass------------
$finish called at time : 72685 ns : File "/home/ubuntu/caravel-soc_fpga-lab-main/lab-fir/fir/tb/fir_tb.v" Line 201
relaunch_sim: Time (s): cpu = 00:00:09 ; elapsed = 00:00:10 . Memory (MB): peak = 8223.867 ; gain = 0.000 ; free physical = 3719 ; free virtual = 7962
save_wave_config {/home/ubuntu/caravel-soc_fpga-lab-main/lab-fir/fir_tb_behav.wcfg}
```

(2)Coefficient Program



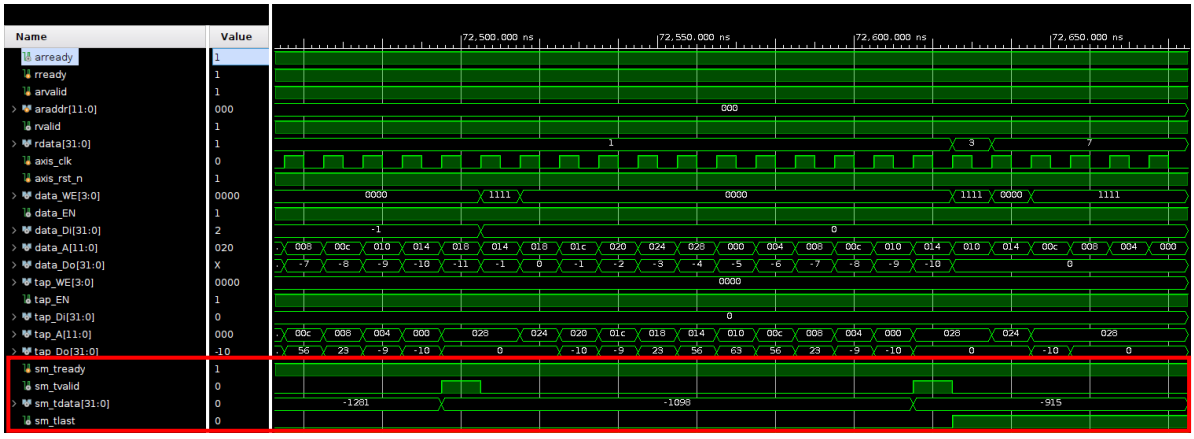
從圖中可看出左半邊為拉起 data_WE 輸入 tap parameter，右半邊則是關閉 data_WE 依序輸出參數。

(3)Data-in Stream-in



當 ss_tready 拉起時，下一個 cycle 時 data_WE 會打開並存入空位

(4)Data-out Stream-out



當運算完成時，拉起 sm_tready 並輸出 Y 值，並在最後拉起 sm_tlast

(5)RAM access control

Tap RAM 的位址由 counter 從 12’h28 依序減 4 往下數即可，Data RAM 則由兩組 counter 一起去控制輸出位址，一個 counter 在接收到 ss_tready 後將數值設為 12’h28 減去一個 offset，並從這個數開始往上數，而這個 offset 則是由另一個 counter，當接收到 ss_tready 時，會+4 並維持，像圖中所示，data_A 為紅框中 12’h28 時，offset 為 0，另一個 data_A 為 12’h24 時，offset 則為 4，以此類推即可得到 shift RAM 的效果。