[SoC Lab] Lab4-2

tags: `SoC Lab`, `SOC Design`

Team 13

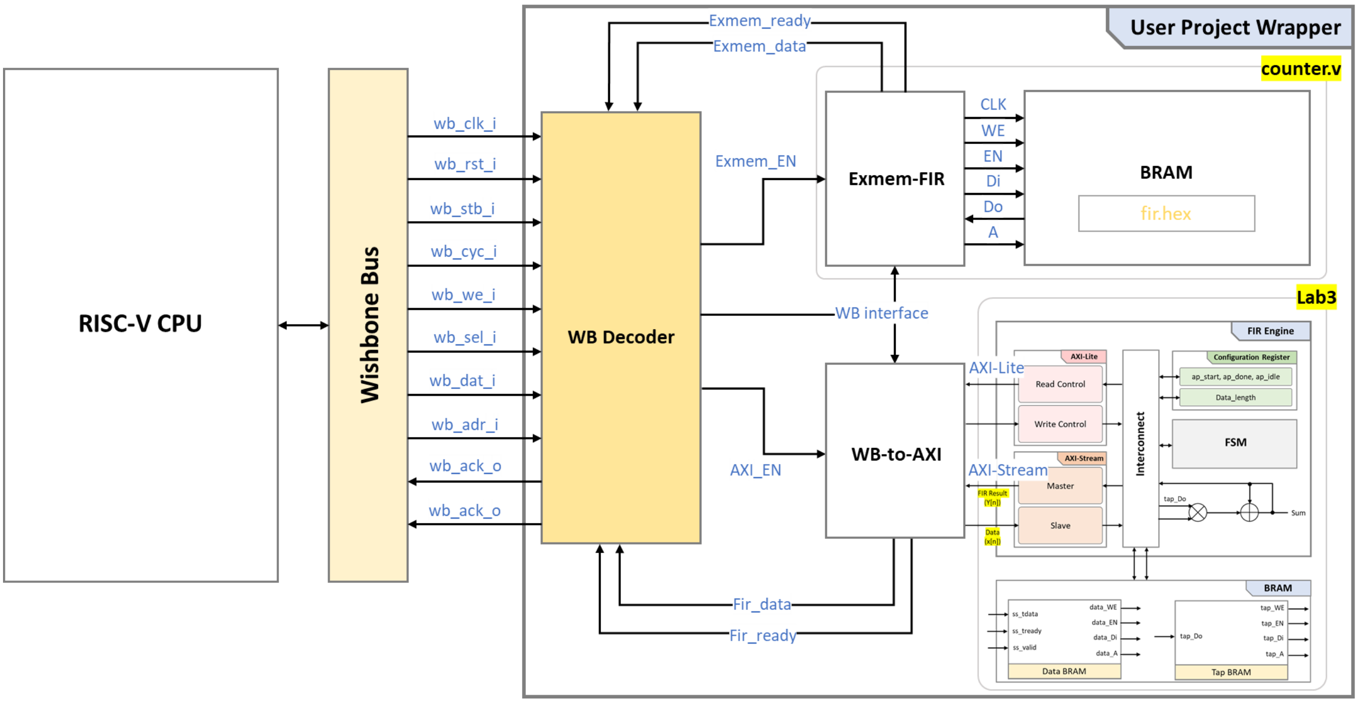
Student ID	Name
311551095	林聖博
312551174	張祐誠

附上此篇Hackmd Linkhttps://hackmd.io/@Sheng08/ByxwnUm4T

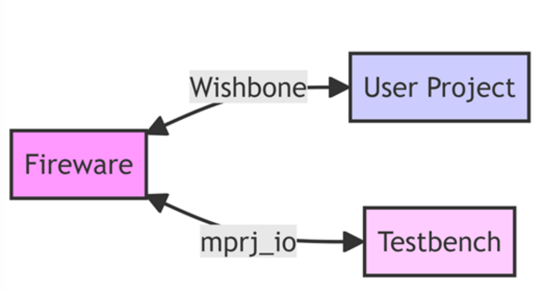
Lab 4 Spec

- Lab4-1 (lab-exmem_fir)
  - https://github.com/bol-edu/caravel-soc_fpga-lab/tree/main/lab-exmem_fir
- Lab4-2 (lab-caravel_fir)
  - https://github.com/bol-edu/caravel-soc_fpga-lab/tree/main/lab-caravel_fir

Design block diagram – datapath, control-path

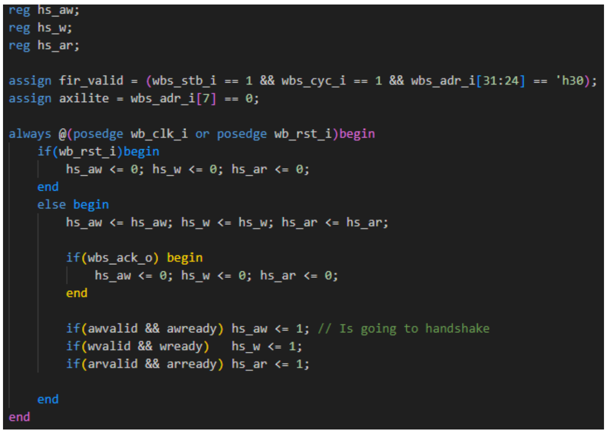


The interface protocol between firmware, user project and testbench



- **Firmware and User Project interface**：
    1. CPU根據 Firmware 的指令，透過 Wishbone（wbs_data_i）向 User Project（FIR）傳輸資料。
        - 例如: 將 FIR 所需的 tap value 存儲到 tap RAM
    2. CPU 也將輸入資料 `x[n]` 透過 Wishbone 傳送給 User Project，並將該資料存於 Data RAM 中。FIR 處理完成後，CPU會根據 Firmware 的指令接收輸出資料 `y[n]`
- **Firmware and Testbench interface**：
    1. Firmware and Testbench 之間的溝通方式是透過 `mprj`
        - 例如: Firmware 會在 `mprj[23:16]` 輸出一個開始訊號（`0xA5`），通知 Testbench 開始計時 delay
    2. 此過程最後，CPU 會將 FIR 的最終輸出透過`mprj` interface 傳送給 Testbench，Testbench 會檢查這些輸出`y[n]`來驗證答案的正確性
- **Firmware Code 執行與配置**：
    1. CPU 將編譯後的 Firmware Code（如`initfir`和`fir`）透過 Wishbone 存儲到 User Project 內的 `user-bram`
    2. CPU 執行再 `counter_la_fir.c` 中定義的主函數，編譯後的 assembly 透過 Wishbone 從 spiflash 中獲取
    3. CPU 配置 `mprj_io[31:16]`，使得 Testbench 可以觀察到這些 pin 的狀態

4. CPU 從 `user_bram` 讀取 fir 的 assembly，並透過 Wishbone 執行 Firmware Code

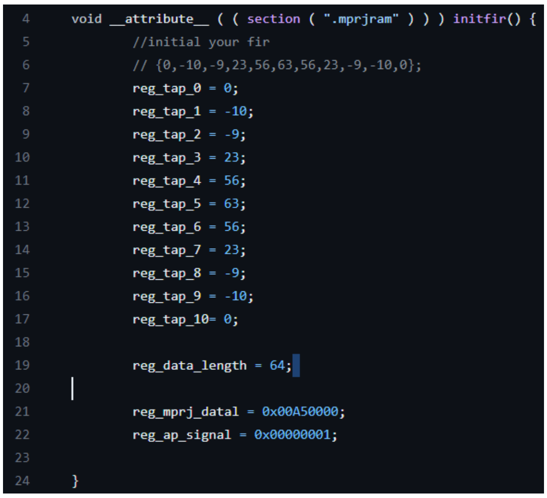
- **WB to AXI**
  - 

```
reg hs_aw;
reg hs_w;
reg hs_ar;

assign fir_valid = (wbs_stb_i == 1 && wbs_cyc_i == 1 && wbs_adr_i[31:24] == 'h30);
assign axilite = wbs_adr_i[7] == 0;

always @(posedge wb_clk_i or posedge wb_rst_i)begin
    if(wb_rst_i)begin
        hs_aw <= 0; hs_w <= 0; hs_ar <= 0;
    end
    else begin
        hs_aw <= hs_aw; hs_w <= hs_w; hs_ar <= hs_ar;

        if(wbs_ack_o) begin
            hs_aw <= 0; hs_w <= 0; hs_ar <= 0;
        end

        if(awvalid && awready) hs_aw <= 1; // Is going to handshake
        if(wvalid && wready)   hs_w <= 1;
        if(arvalid && arready) hs_ar <= 1;

    end
end
```

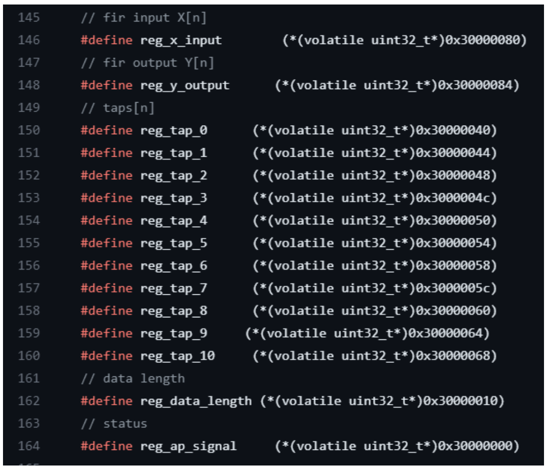
  - 當 hs (handshake) 為 1 時，代表 ack 訊號即將傳回給 WB

Waveform and analysis of the hardware/software behavior

- **Software**
  - 輸入 64 個 X 值(分別為 **1~64**) 到地址 `0X30000080`，並透過 Fireware 指令使 CPU 將 X 輸入給 FIR，而 `reg_tap_0` 則定義在 `0X30000040`

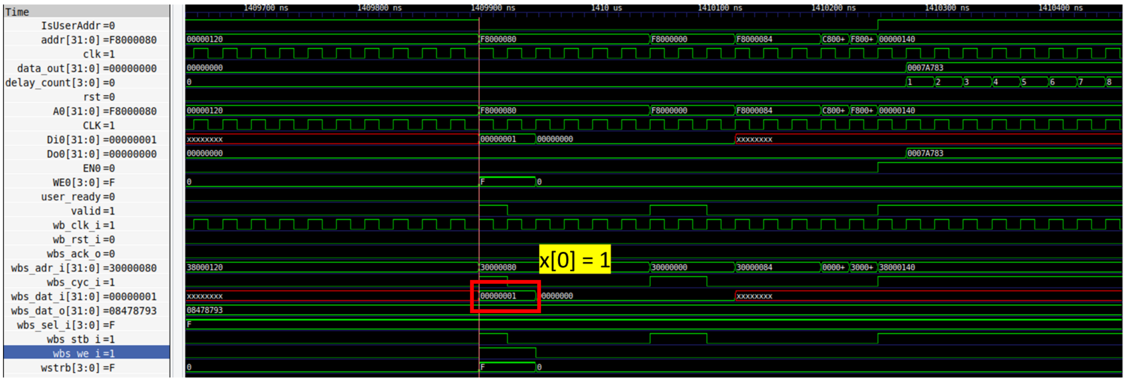


```
void __attribute__ ( ( section ( ".mprjram" ) ) ) initfir() {
        //initial your fir
        // {0,-10,-9,23,56,63,56,23,-9,-10,0};
        reg_tap_0 = 0;
        reg_tap_1 = -10;
        reg_tap_2 = -9;
        reg_tap_3 = 23;
        reg_tap_4 = 56;
        reg_tap_5 = 63;
        reg_tap_6 = 56;
        reg_tap_7 = 23;
        reg_tap_8 = -9;
        reg_tap_9 = -10;
        reg_tap_10= 0;

        reg_data_length = 64;


        reg_mprj_datal = 0x00A50000;
        reg_ap_signal = 0x00000001;

}
```

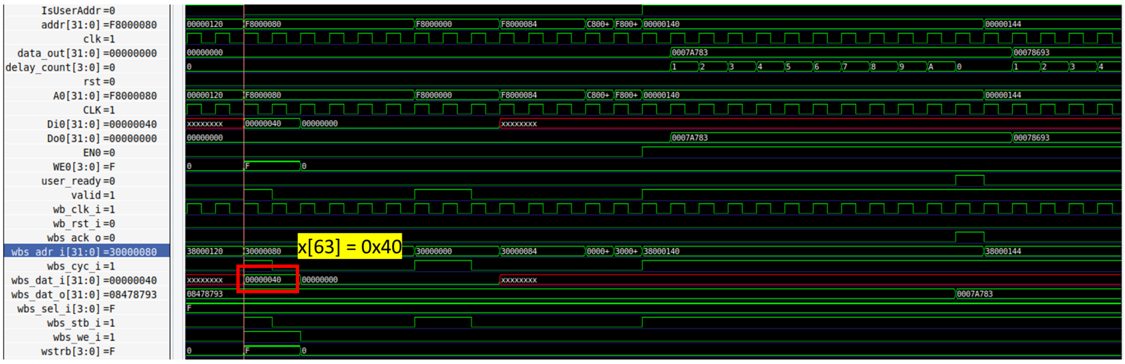


```
// fir input X[n]
#define reg_x_input         (*(volatile uint32_t*)0x30000080)
// fir output Y[n]
#define reg_y_output        (*(volatile uint32_t*)0x30000084)
// taps[n]
#define reg_tap_0       (*(volatile uint32_t*)0x30000040)
#define reg_tap_1       (*(volatile uint32_t*)0x30000044)
#define reg_tap_2       (*(volatile uint32_t*)0x30000048)
#define reg_tap_3       (*(volatile uint32_t*)0x3000004c)
#define reg_tap_4       (*(volatile uint32_t*)0x30000050)
#define reg_tap_5       (*(volatile uint32_t*)0x30000054)
#define reg_tap_6       (*(volatile uint32_t*)0x30000058)
#define reg_tap_7       (*(volatile uint32_t*)0x3000005c)
#define reg_tap_8       (*(volatile uint32_t*)0x30000060)
#define reg_tap_9      (*(volatile uint32_t*)0x30000064)
#define reg_tap_10      (*(volatile uint32_t*)0x30000068)
// data length
#define reg_data_length (*(volatile uint32_t*)0x30000010)
// status
#define reg_ap_signal       (*(volatile uint32_t*)0x30000000)
```

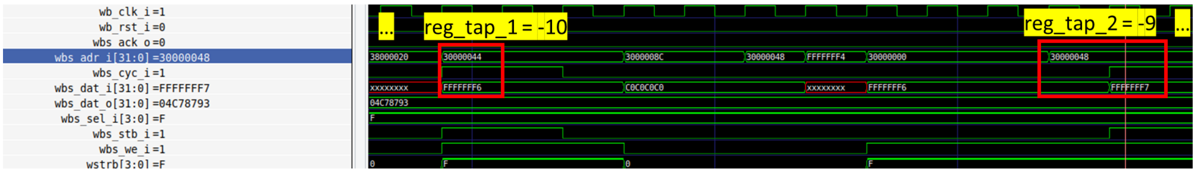
  - Input(`x[n]`):
    - First(x[0]): x[0]為1開始傳



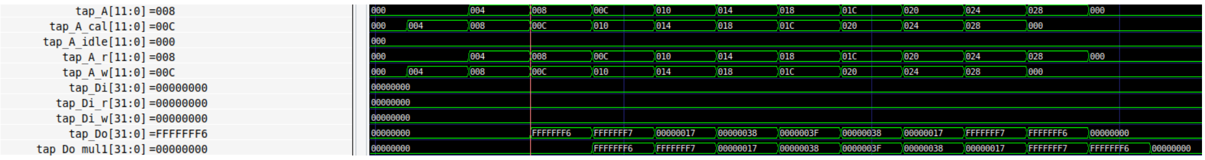
    - Last(x[63]): x[63]為64(`0x40`)



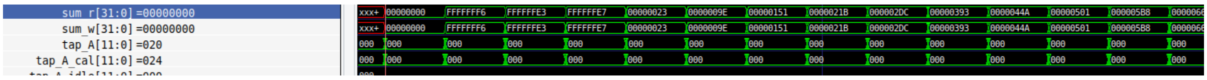
  - Tap:



- **Hardware**
  - 讀取Tap值



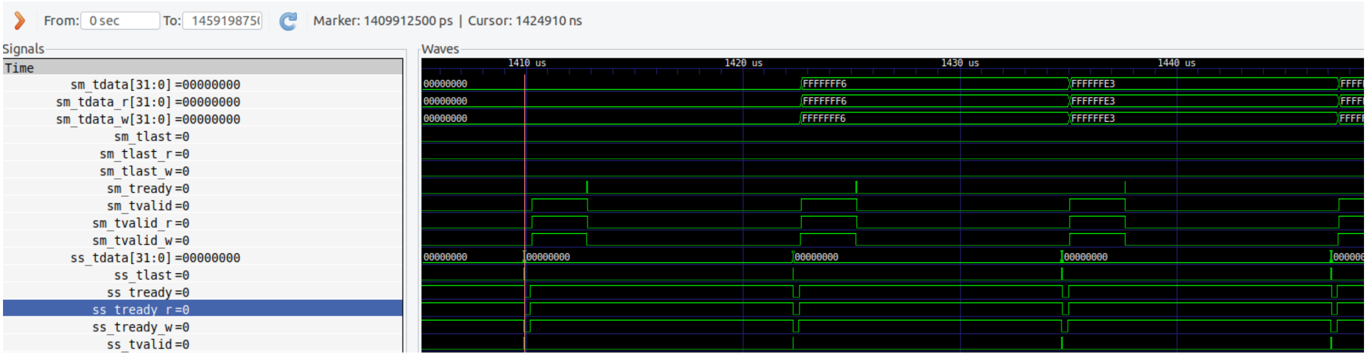
  - FIR 計算(乘加結果)



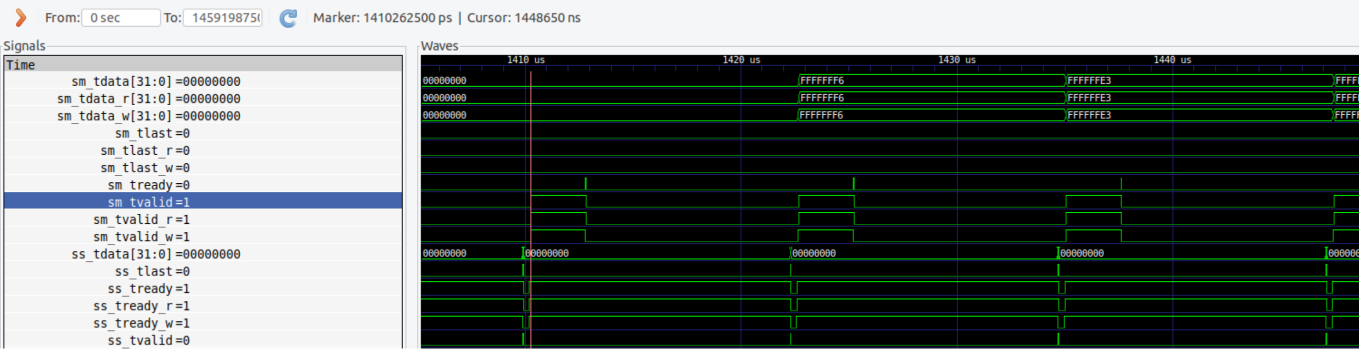
What is the FIR engine theoretical throughput, i.e. data rate? Actually measured throughput?

- **Theoretical troughput (cycle):** `11 + 1`; **Data rate(1/cycle):** `1 / (11 + 1)`
- **Actually measured throughput:** `14 cycle`

```
Latency: (1410262500 - 1409912500)ps / (25000)ps = 14 cycle
```



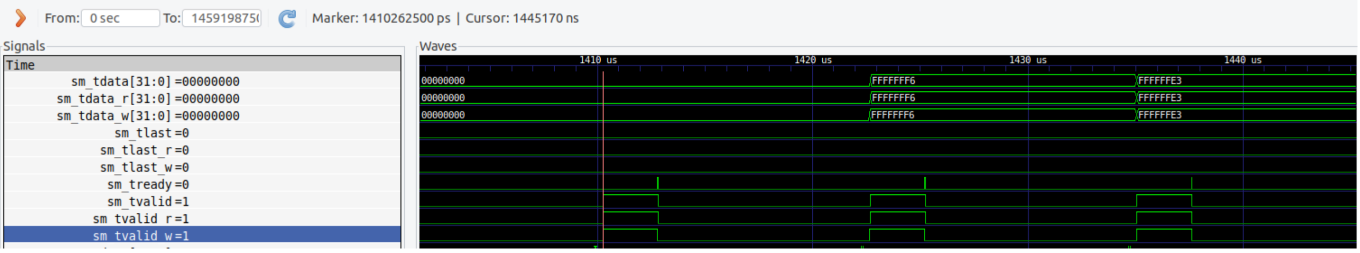
```
From: 0 sec  To: 1459198750  Marker: 1409912500 ps | Cursor: 1424910 ns
Signals
Time
     sm_tdata[31:0] =00000000
   sm_tdata_r[31:0] =00000000
   sm_tdata_w[31:0] =00000000
          sm_tlast =0
        sm_tlast_r =0
        sm_tlast_w =0
         sm_tready =0
         sm_tvalid =0
       sm_tvalid_r =0
       sm_tvalid_w =0
     ss_tdata[31:0] =00000000
          ss_tlast =0
         ss_tready =0
       ss_tready_r =0
       ss_tready_w =0
         ss_tvalid =0
```



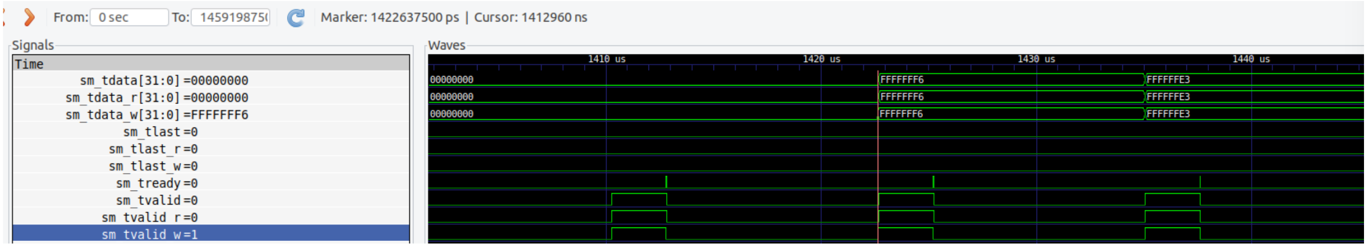
```
From: 0 sec  To: 1459198750  Marker: 1410262500 ps | Cursor: 1448650 ns
Signals
Time
     sm_tdata[31:0] =00000000
   sm_tdata_r[31:0] =00000000
   sm_tdata_w[31:0] =00000000
          sm_tlast =0
        sm_tlast_r =0
        sm_tlast_w =0
         sm_tready =0
         sm_tvalid =1
       sm_tvalid_r =1
       sm_tvalid_w =1
     ss_tdata[31:0] =00000000
          ss_tlast =0
         ss_tready =1
       ss_tready_r =1
       ss_tready_w =1
         ss_tvalid =0
```

What is latency for firmware to feed data?

```
Latency: (1422637500 - 1410262500)ps / (25000)ps = 495 cycle
```



```
From: 0 sec  To: 1459198750  Marker: 1410262500 ps | Cursor: 1445170 ns
Signals
Time
     sm_tdata[31:0] =00000000
   sm_tdata_r[31:0] =00000000
   sm_tdata_w[31:0] =00000000
          sm_tlast =0
        sm_tlast_r =0
        sm_tlast_w =0
         sm_tready =0
         sm_tvalid =1
       sm_tvalid_r =1
       sm_tvalid_w =1
```



```
From: 0 sec  To: 1459198750  Marker: 1422637500 ps | Cursor: 1412960 ns
Signals
Time
     sm_tdata[31:0] =00000000
   sm_tdata_r[31:0] =00000000
   sm_tdata_w[31:0] =FFFFFFF6
          sm_tlast =0
        sm_tlast_r =0
        sm_tlast_w =0
         sm_tready =0
         sm_tvalid =0
       sm_tvalid_r =0
       sm_tvalid_w =1
```

What techniques used to improve the throughput?

Does bram12 give better performance, in what way?

以目前的情況下使用 BRAM12 並不會帶來比 BRAM11 更好的效能，由於Address bits 變大，又在 Access register 的過程中會比較 Address，所以 compare 的時間會變長。 另外，BRAM11 已足夠存放本 Lab 存放的資料數量。綜上所述，**猜測** BRAM12 不會有更好的 performance。

Can you suggest other method to improve the performance?

1. 由於本 Lab 只有 one Adder 和 one multiplier，所以無法執行平行運算，若提供更多，可能可以稍微提升 performance

2. 若有 shifted register 用來存放 data 而非 bram 的話，應該可以減少資料搬運的時間
3. firmware 應該有更好的寫法，但目前沒有想到

Any other insights ?

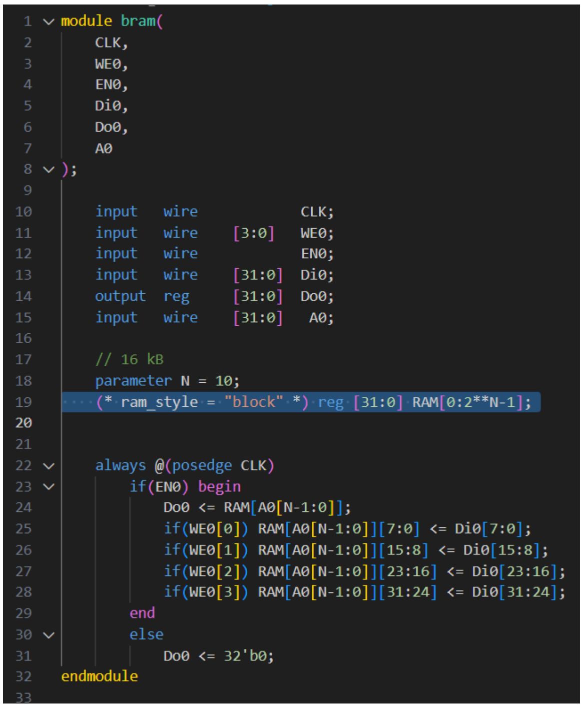
Lab4-1 v.s Lab4-2 設計

在實作 Lab4-1 與 Lab4-2 過程中可發現，在相同實驗條件架構下，Lab4-1 屬於利用軟體(韌體)方式計算 FIR；而 Lab4-2 則使用硬體方式計算，因此發現 Lab4-2 所需的 Cycle 少。

BRAM

- 此實驗有三個 BRAM —— User BRAM, Data RAM 和 Tap bram，但 USER BRAM 通過`(* ram_style = "block" *)` 被synthesis為一實體Block RAM。

```
2. Memory
---------

+-------------------+------+-------+------------+-----------+-------+
|     Site Type     | Used | Fixed | Prohibited | Available | Util% |
+-------------------+------+-------+------------+-----------+-------+
| Block RAM Tile    |    1 |     0 |          0 |       140 |  0.71 |
|   RAMB36/FIFO*    |    1 |     0 |          0 |       140 |  0.71 |
|     RAMB36E1 only |    1 |       |            |           |       |
|   RAMB18          |    0 |     0 |          0 |       280 |  0.00 |
+-------------------+------+-------+------------+-----------+-------+
```



```
module bram(
    CLK,
    WE0,
    EN0,
    Di0,
    Do0,
    A0
);

    input   wire            CLK;
    input   wire    [3:0]   WE0;
    input   wire            EN0;
    input   wire    [31:0]  Di0;
    output  reg     [31:0]  Do0;
    input   wire    [31:0]   A0;

    // 16 kB
    parameter N = 10;
    (* ram_style = "block" *) reg [31:0] RAM[0:2**N-1];


    always @(posedge CLK)
        if(EN0) begin
            Do0 <= RAM[A0[N-1:0]];
            if(WE0[0]) RAM[A0[N-1:0]][7:0] <= Di0[7:0];
            if(WE0[1]) RAM[A0[N-1:0]][15:8] <= Di0[15:8];
            if(WE0[2]) RAM[A0[N-1:0]][23:16] <= Di0[23:16];
            if(WE0[3]) RAM[A0[N-1:0]][31:24] <= Di0[31:24];
        end
        else
            Do0 <= 32'b0;
endmodule
```

- 其餘 Data RAM 和 Tap bram 被合成為 LUTs 或 FF

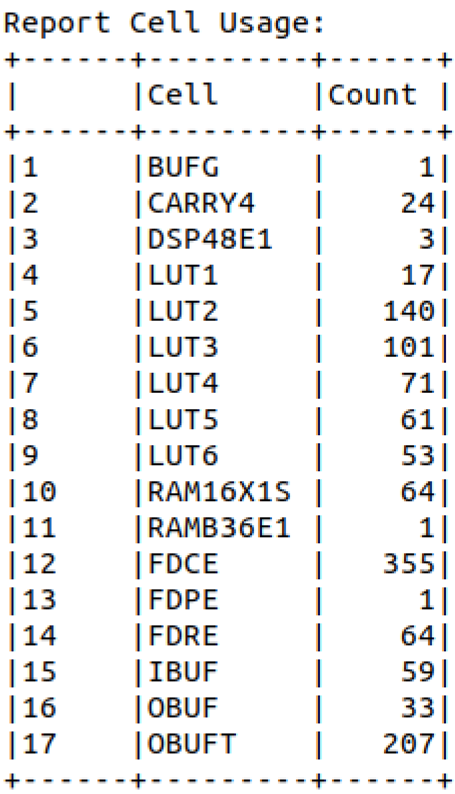
```
1. Slice Logic
--------------

+----------------------------+------+-------+------------+-----------+-------+
|          Site Type         | Used | Fixed | Prohibited | Available | Util% |
+----------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*                |  381 |     0 |          0 |     53200 |  0.72 |
|   LUT as Logic             |  317 |     0 |          0 |     53200 |  0.60 |
|   LUT as Memory            |   64 |     0 |          0 |     17400 |  0.37 |
|     LUT as Distributed RAM |   64 |     0 |            |           |       |
|     LUT as Shift Register  |    0 |     0 |            |           |       |
| Slice Registers            |  420 |     0 |          0 |    106400 |  0.39 |
|   Register as Flip Flop    |  420 |     0 |          0 |    106400 |  0.39 |
|   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 |
+----------------------------+------+-------+------------+-----------+-------+
```

Synthesis report

- **RTL Component:**

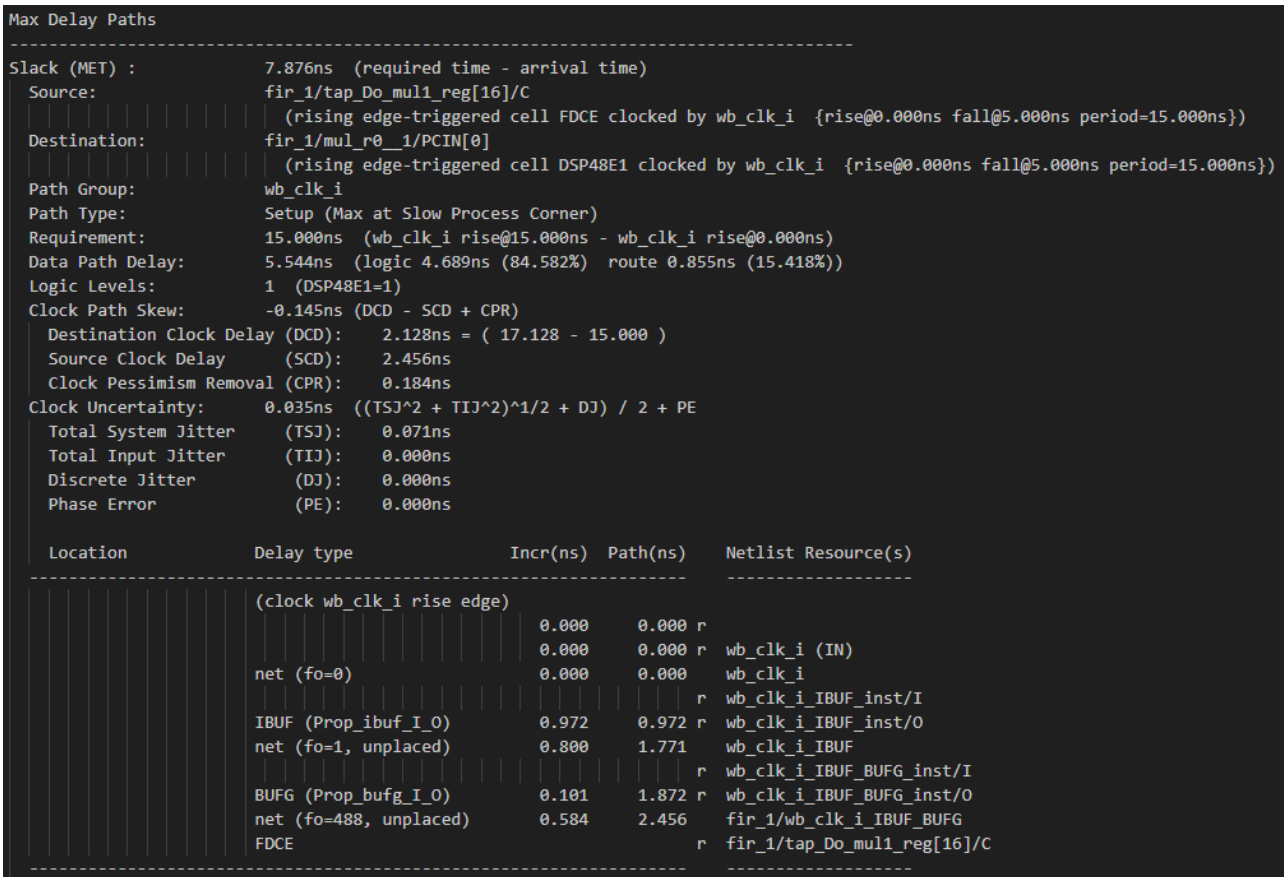
```
Start RTL Component Statistics
---------------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
	   2 Input   32 Bit       Adders := 2
	   2 Input   12 Bit       Adders := 2
	   2 Input   10 Bit       Adders := 1
	   2 Input    7 Bit       Adders := 1
	   2 Input    4 Bit       Adders := 1
+---Registers :
	               32 Bit    Registers := 11
	               16 Bit    Registers := 1
	               12 Bit    Registers := 2
	               10 Bit    Registers := 1
	                6 Bit    Registers := 1
	                4 Bit    Registers := 3
	                1 Bit    Registers := 26
+---Multipliers :
	              32x32  Multipliers := 1
+---RAMs :
	              32K Bit	(1024 X 32 bit)          RAMs := 1
	              352 Bit	(11 X 32 bit)          RAMs := 2
+---Muxes :
	   2 Input   32 Bit        Muxes := 14
	   3 Input   32 Bit        Muxes := 2
	   2 Input   12 Bit        Muxes := 3
	   3 Input   12 Bit        Muxes := 1
	   3 Input   10 Bit        Muxes := 1
	   2 Input    8 Bit        Muxes := 1
	   2 Input    7 Bit        Muxes := 3
	   2 Input    6 Bit        Muxes := 6
	   3 Input    6 Bit        Muxes := 1
	   2 Input    4 Bit        Muxes := 1
	   3 Input    4 Bit        Muxes := 1
	   4 Input    4 Bit        Muxes := 1
	   2 Input    1 Bit        Muxes := 25
	   3 Input    1 Bit        Muxes := 11
	   4 Input    1 Bit        Muxes := 1
---------------------------------------------------------------------------------
Finished RTL Component Statistics
```

- **Report Cell Usage:**



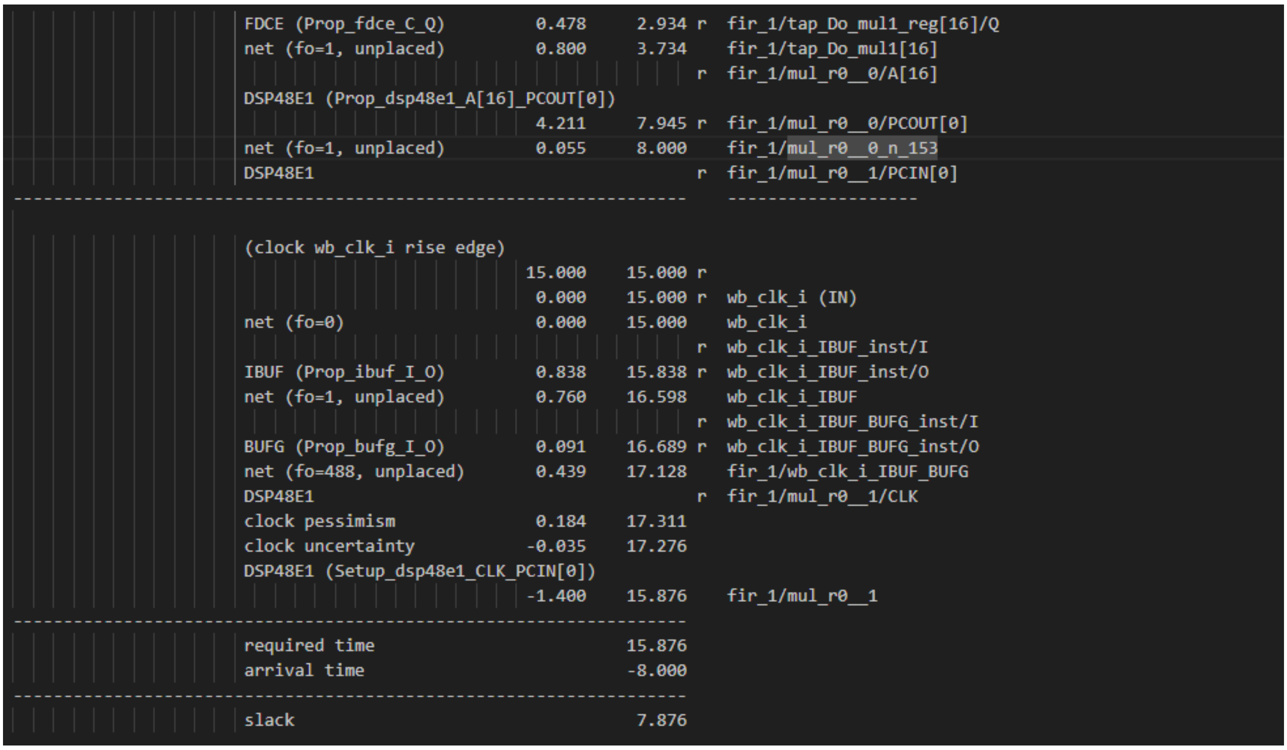
```
Report Cell Usage:
+------+---------+------+
|      |Cell     |Count |
+------+---------+------+
|1     |BUFG     |     1|
|2     |CARRY4   |    24|
|3     |DSP48E1  |     3|
|4     |LUT1     |    17|
|5     |LUT2     |   140|
|6     |LUT3     |   101|
|7     |LUT4     |    71|
|8     |LUT5     |    61|
|9     |LUT6     |    53|
|10    |RAM16X1S |    64|
|11    |RAMB36E1 |     1|
|12    |FDCE     |   355|
|13    |FDPE     |     1|
|14    |FDRE     |    64|
|15    |IBUF     |    59|
|16    |OBUF     |    33|
|17    |OBUFT    |   207|
+------+---------+------+
```

- **Max delay path:**



```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             7.876ns  (required time - arrival time)
  Source:                 fir_1/tap_Do_mul1_reg[16]/C
                            (rising edge-triggered cell FDCE clocked by wb_clk_i  {rise@0.000ns fall@5.000ns period=15.000ns})
  Destination:            fir_1/mul_r0__1/PCIN[0]
                            (rising edge-triggered cell DSP48E1 clocked by wb_clk_i  {rise@0.000ns fall@5.000ns period=15.000ns})
  Path Group:             wb_clk_i
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            15.000ns  (wb_clk_i rise@15.000ns - wb_clk_i rise@0.000ns)
  Data Path Delay:        5.544ns  (logic 4.689ns (84.582%)  route 0.855ns (15.418%))
  Logic Levels:           1  (DSP48E1=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 17.128 - 15.000 )
    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

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
                         (clock wb_clk_i rise edge)
                                                      0.000     0.000 r
                                                      0.000     0.000 r  wb_clk_i (IN)
                         net (fo=0)                   0.000     0.000    wb_clk_i
                                                                      r  wb_clk_i_IBUF_inst/I
                         IBUF (Prop_ibuf_I_O)         0.972     0.972 r  wb_clk_i_IBUF_inst/O
                         net (fo=1, unplaced)         0.800     1.771    wb_clk_i_IBUF
                                                                      r  wb_clk_i_IBUF_BUFG_inst/I
                         BUFG (Prop_bufg_I_O)         0.101     1.872 r  wb_clk_i_IBUF_BUFG_inst/O
                         net (fo=488, unplaced)       0.584     2.456    fir_1/wb_clk_i_IBUF_BUFG
                         FDCE                                         r  fir_1/tap_Do_mul1_reg[16]/C
  -------------------------------------------------------------------    -------------------
```

- **Slack met:**



```
FDCE (Prop_fdce_C_Q)          0.478     2.934 r  fir_1/tap_Do_mul1_reg[16]/Q
net (fo=1, unplaced)          0.800     3.734    fir_1/tap_Do_mul1[16]
                                              r  fir_1/mul_r0__0/A[16]
DSP48E1 (Prop_dsp48e1_A[16]_PCOUT[0])
                              4.211     7.945 r  fir_1/mul_r0__0/PCOUT[0]
net (fo=1, unplaced)          0.055     8.000    fir_1/mul_r0__0_n_153
DSP48E1                                       r  fir_1/mul_r0__1/PCIN[0]
-------------------------------------------------------------------    -------------------

(clock wb_clk_i rise edge)
                             15.000    15.000 r
                              0.000    15.000 r  wb_clk_i (IN)
net (fo=0)                    0.000    15.000    wb_clk_i
                                              r  wb_clk_i_IBUF_inst/I
IBUF (Prop_ibuf_I_O)          0.838    15.838 r  wb_clk_i_IBUF_inst/O
net (fo=1, unplaced)          0.760    16.598    wb_clk_i_IBUF
                                              r  wb_clk_i_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)          0.091    16.689 r  wb_clk_i_IBUF_BUFG_inst/O
net (fo=488, unplaced)        0.439    17.128    fir_1/wb_clk_i_IBUF_BUFG
DSP48E1                                       r  fir_1/mul_r0__1/CLK
clock pessimism               0.184    17.311
clock uncertainty            -0.035    17.276
DSP48E1 (Setup_dsp48e1_CLK_PCIN[0])
                             -1.400    15.876    fir_1/mul_r0__1
-------------------------------------------------------------------
required time                          15.876
arrival time                           -8.000
-------------------------------------------------------------------
slack                                   7.876
```

Github link

- https://github.com/Sheng08/SoC-Lab-FIR_Lab4

🚀 補充

> 無