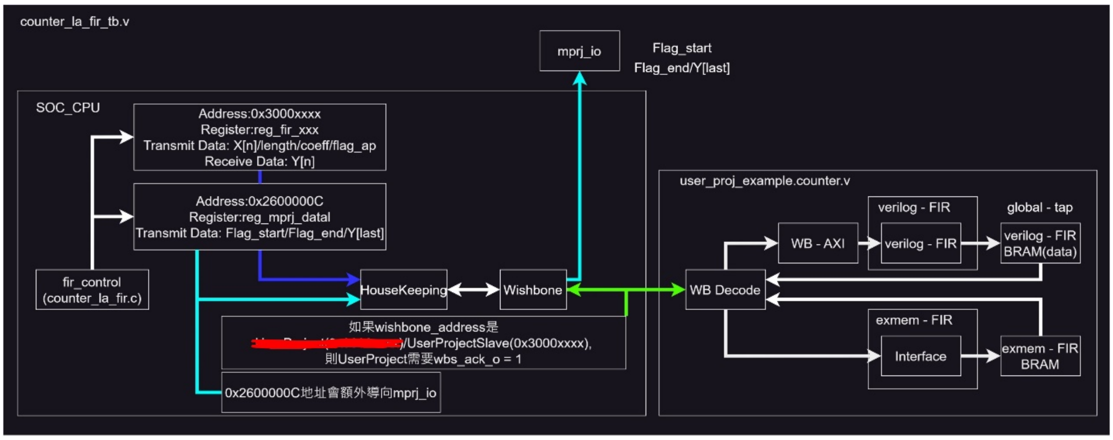
SOC LAB 4-2

511506022 何佳玲
312611096 黃鉦淳
312605003 王語

➢ Design block diagram



➢ The interface protocol between firmware, user project and testbench

- <u>Testbench: 透過 flash control 將資料傳達給 CPU 執行 firmware code</u>

```
caravel uut (
    .clock    (clock),
    .gpio     (gpio),
    .mprj_io  (mprj_io),
    .flash_csb(flash_csb),
    .flash_clk(flash_clk),
    .flash_io0(flash_io0),
    .flash_io1(flash_io1),
    .resetb   (RSTB)
);

spiflash #(
    .FILENAME("counter_la_fir.hex")
) spiflash (
    .csb(flash_csb),
    .clk(flash_clk),
    .io0(flash_io0),
    .io1(flash_io1),
    .io2(),          // not used
    .io3()           // not used
);
```

- <u>Firmware(參考 fir.c):</u>

```
109 reg_wb_enable = 1;
110
111 // Flag - Start latency-timer (in testbench)
112 reg_mprj_datal = (0xA5<<16);
113
114 // FIR - data length
115 reg_fir_data_length = data_length;
116
117
118 // FIR - tap coefficient
119 for(int n = 0; n < N; n++)
120   reg_fir_coeff(n) = taps[n];
121
122 // AXI-lite(Write/Read) - Correct
123
124 // FIR - ap_start
125 reg_fir_control = (1 << bits_control_ap_start);
126
127 int y = 0;
128 for (int n = 0; n < data_length; n++)
129 {
130   // Wait for FIR ready to write x
131   while((reg_fir_control & (1 << bits_control_x_readyWrite)) != (1 << bits_control_x_readyWrite)) ;
132
133   // Send x[n] to FIR
134   reg_fir_x = n;
135
136   // Wait for FIR ready to read y
137   while((reg_fir_control & (1 << bits_control_y_readyRead)) != (1 << bits_control_y_readyRead)) ;
138
139   // Receive y[n] from FIR
140   y = reg_fir_y;
141 }
142
143 // FIR - ap_idle
144 while((reg_fir_control & (1 << bits_control_ap_idle)) != (1 << bits_control_ap_idle)) ;
145
146 // Flag - Stop latency-timer (in testbench)
147 reg_mprj_datal = (y<<24) | (0x5A<<16);
```

程式前面定義各個 ip 的位置，以下解說程式

112 :

reg_mprj_datal 有 32bit，前 16 -> CPU ; 後 16 -> user_project

因為在 cpu 運作所以需要左移 16bits

125:

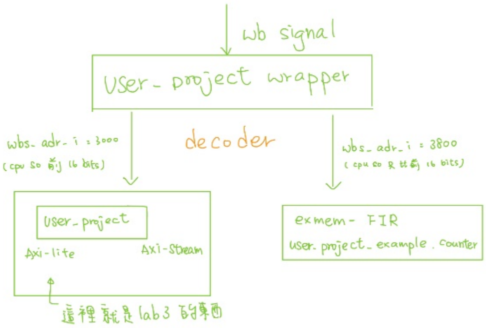
ap_start signal

127~147 :

用 bits_control => ap_start , ap_done, ap_idle

這邊的訊號會透過 mprj 傳輸到 fir.v

- **<u>User_project</u>**

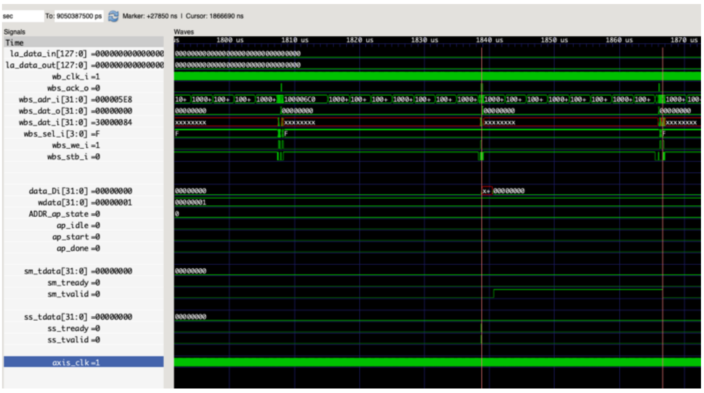


➢ **Waveform and analysis of the hardware/software behavior.**

Software :

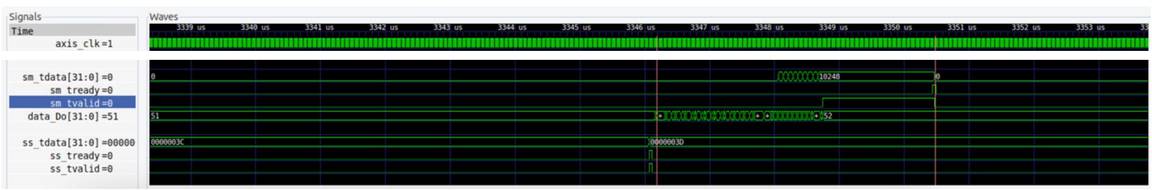
1. 將 Firmware 放在 SPI Flash 運行

   從傳送 X[n]至接收 Y[n] 約歷經 26 μs



2. 將 Firmware 放在 User RAM 運行 (0x3800_xxxx)

   從傳送 X[n]至接收 Y[n] 約歷經 4.5 μs

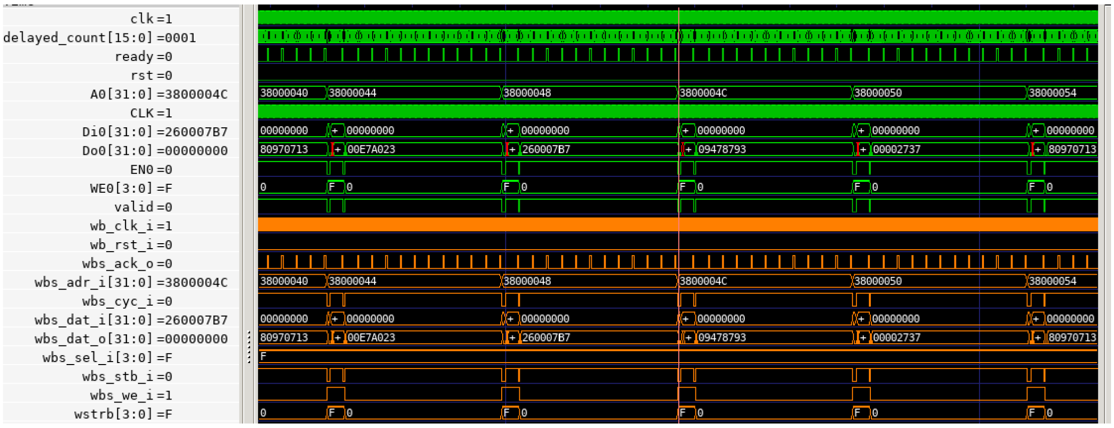


3. 結論

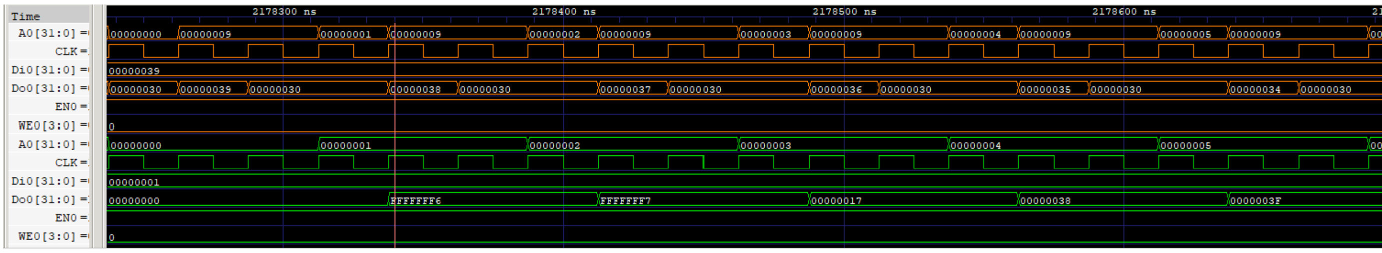
   由於 Firmware 主掌資料流的控制權，意味著 Firmware 的效能限縮硬體的運算速度，從接收到發送歷經上百個 cycle，皆超出計算單筆資料的運算時間，因此我們使用 Lab3 中資源使用最少的 FIR 作為本次實驗的硬體端設計。

Hardware:

- **0x3800 for exmem-FIR**

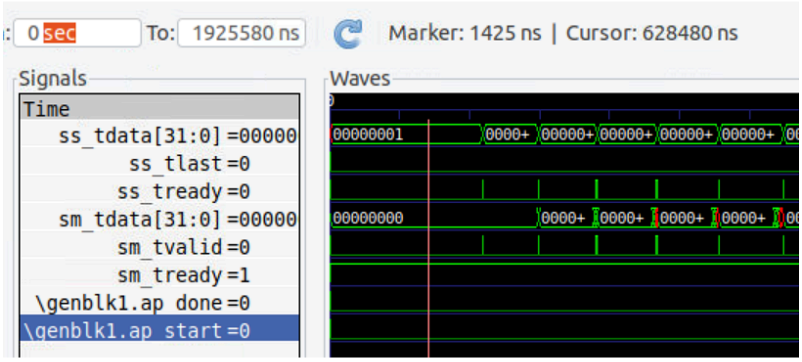


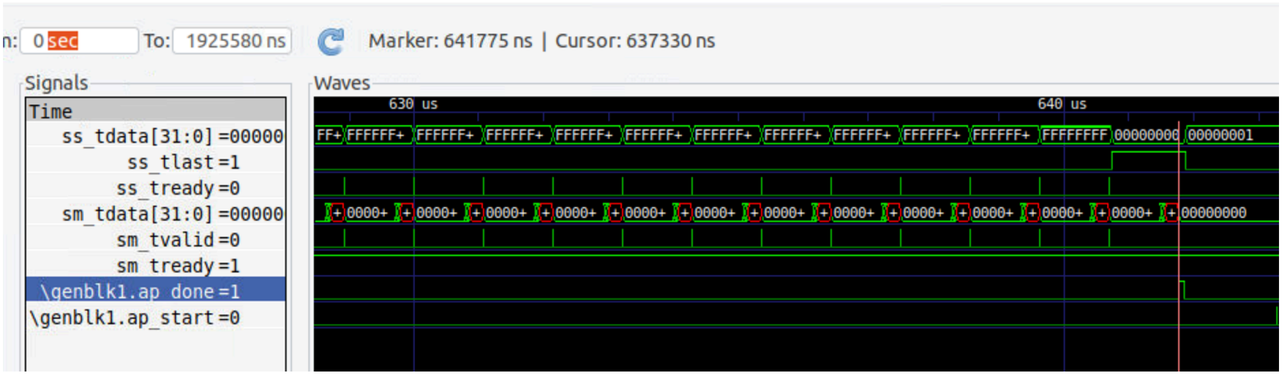
- **0x3000 for FIR-AXILITE(BRAM) & FIR-AXISTREAM(BRAM)**



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

Ap_start assert at 1425 ns ,and Ap_done assert at 641775 ns





以 Lab3 結果而言，最理想的情況下可以在 640350 ns 內完成 600 筆資料的運算（每筆資料相當於 11 次乘法，10 次加法）。

Data rate = 9370000

Throughput = 196,770,000

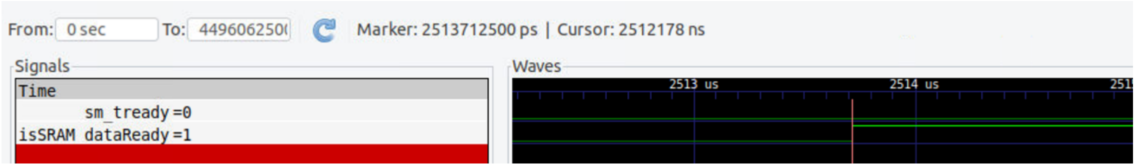
```
----- Times# 1 -----
Start latency-timer
Stop latency-timer
Elapsed time:      959150 [ns]
--------------------
Correct, ans = 118, golden_ans = 118

----- Times# 2 -----
Start latency-timer
Stop latency-timer
Elapsed time:      970900 [ns]
--------------------
Correct, ans = 118, golden_ans = 118

----- Times# 3 -----
Start latency-timer
Stop latency-timer
Elapsed time:      970900 [ns]
--------------------
Correct, ans = 118, golden_ans = 118
```

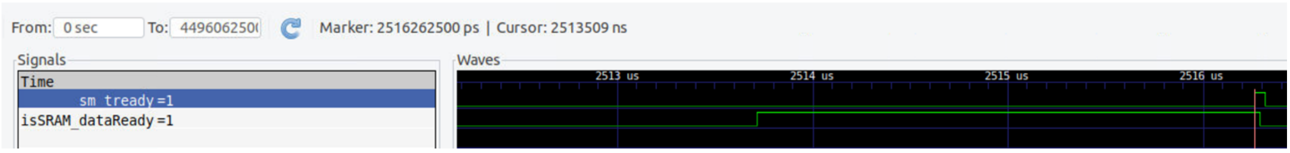
但是實際上量測到的結果，在 970900 ns 內只能完成 64 筆資料的運算

Data rate = 65920

Throughput = 1,384,320

- **What is latency for firmware to feed data?**





Latency : 2,550 ns

- **What techniques used to improve the throughput?**

由於目前效能受限於 Firmware，可以嘗試以下方法提升 throughput

1. 增加資料傳輸效率，以本次實驗為例，單筆 x[n]不會用到 32bits，所以可以合併多筆 x[n]一次傳送，假設 x[n]的範圍為 1~64，代表只需要使用到 7bits，可以讓 data[31:25]代表 x[n]，data[24:18]代表 x[n+1]，以此類推，Y[n]的部分也可以仿效。
2. 增加 Input/Output Buffer，如此一來可以配合 CPU 節奏收發資料。
3. 加快 Caraval SoC 運作頻率

- **Does bram12 give better performance, in what way?**

多出更多 RAM 能讓運算更有彈性，減少 control 負擔，但在本次實驗中成效有限，因為硬體做再快也會受限於韌體速度。

Bram12 多了一個記憶體位置，可以當作 Buffer 使用，也就是說當 FIR 還未完成當前運算的期間，CPU 就能先將下一次運算需要的 x[n]寫入，並且不會覆蓋掉可能需要用到的資料。

- **Can you suggest other method to improve the performance?**

進一步推廣單次傳送多筆資料的想法，並且推廣至更廣泛的數值，譬如使用 BF16 格式進行資料傳輸，單次傳輸可以傳送 2 筆資料，而且數值範圍廣泛，雖然增加硬體負擔，需要多進行一個解碼、編碼的任務，但以上百個 cycle 內完成的前提下硬體依舊游刃有餘。

使用 Share memory 進行資料傳輸，CPU 不必等待 FIR 接收資料，可以事先將多筆 x[n]資料儲存在共用記憶體，等到 FIR 需要時再讀取，反之 FIR 也能將運算完成的 Y[n]存放在共享記憶體，等 CPU 有空再去讀取。

- **Any other insights ?**

  對比將 Firmware 存放在 SPI Flash 以及 User RAM 之運行時間

  Initial :8063625ns

```
λ make
Reading counter_la_fir.hex
counter_la_fir.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_fir.vcd opened for output.
Start latency-timer
Stop latency-timer
Elapsed time:    8063625 [ns]
counter_la_fir_tb.v:196: $finish called at 9050387500 (1ps)
```

  Improved:970900 ns

```
----- Times# 1 -----
Start latency-timer
Stop latency-timer
Elapsed time:     959150 [ns]
--------------------
Correct, ans = 118, golden_ans = 118

----- Times# 2 -----
Start latency-timer
Stop latency-timer
Elapsed time:     970900 [ns]
--------------------
Correct, ans = 118, golden_ans = 118

----- Times# 3 -----
Start latency-timer
Stop latency-timer
Elapsed time:     970900 [ns]
--------------------
Correct, ans = 118, golden_ans = 118
```

  控制 feedback 的時間，如下圖

```
always @(posedge clk) begin
    if (rst) begin
        ready <= 1'b0;
        delayed_count <= 16'b0;
    end else begin
        ready <= 1'b0;
        if (wbs_adr_i[31:16]==16'h3800) begin
            // $display("0x3800");
            if (valid && !ready) begin
                // $display("delay_count = %d",delayed_count);
                if (delayed_count == DELAYS) begin
                    delayed_count <= 16'b0;
                    ready <= 1'b1;
                    // $display("ack");
                end
                else begin
                    delayed_count <= delayed_count + 1;
                end
            end
        end
    end
end
```

➢ TIME REPORT

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.530 ns	Worst Hold Slack (WHS):	0.132 ns	Worst Pulse Width Slack (WPWS):	
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TP	
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	
Total Number of Endpoints:	416	Total Number of Endpoints:	416	Total Number of Endpoints:	

All user specified timing constraints are met.

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

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |  380 |     0 |          0 |     53200 |  0.71 |
|   LUT as Logic          |  380 |     0 |          0 |     53200 |  0.71 |
|   LUT as Memory         |    0 |     0 |          0 |     17400 |  0.00 |
| Slice Registers         |  400 |     0 |          0 |    106400 |  0.38 |
|   Register as Flip Flop |  260 |     0 |          0 |    106400 |  0.24 |
|   Register as Latch     |  140 |     0 |          0 |    106400 |  0.13 |
| F7 Muxes                |    0 |     0 |          0 |     26600 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     13300 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
* Warning! The Final LUT count, after physical optimizations and full implementa
completed, for a more realistic count.


1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 7     |          Yes |           - |          Set |
| 375   |          Yes |           - |        Reset |
| 0     |          Yes |         Set |            - |
| 18    |          Yes |       Reset |            - |
+-------+--------------+-------------+--------------+
```

- Design Sources (2)
  - Non-module Files (1)
    - counter.v
  - **user_project_wrapper** (user_project_wrapper.v) (2)
    - mprj : user_proj_example (user_project.v) (4)
      - bram_fir_exmem : bram (bram.v)
      - fir_DUT : fir (fir.v)
      - tap_RAM : bram (bram.v)
      - data_RAM : bram (bram.v)
    - counter : user_proj_counter (user_proj_example.counter.v) (1)
      - user_bram : bram (bram.v)
- Constraints (1)
  - constrs_1 (1)
    - system_xdc.xdc (target)
- Simulation Sources (2)
  - sim_1 (2)
    - Non-module Files (1)
      - counter.v
    - **user_project_wrapper** (user_project_wrapper.v) (2)
      - mprj : user_proj_example (user_project.v) (4)
      - counter : user_proj_counter (user_proj_example.counter.v) (1)
- Utility Sources (1)
  - utils_1 (1)
    - Design Checkpoint (1)
      - user_proj_counter.dcp