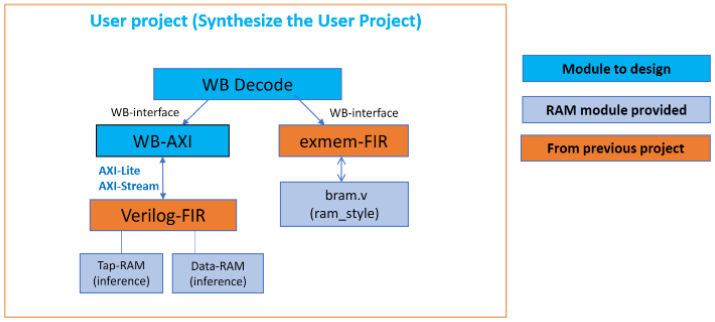
# SoC Lab 4-2 Report

312510140 何律明
312510220 張承宗

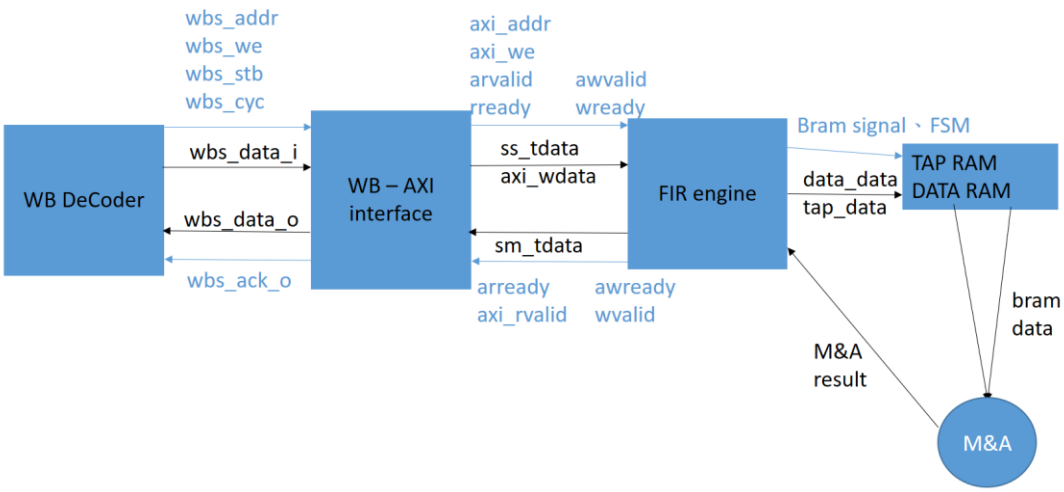
- ## Design block diagram and protocol between WB & Verilog-FIR (design)

以圖片中的橘色框框的左半邊為參考，wbs_addr_i 都是以 0x30 為開頭就是我們 user project 執行 fir 計算的部分(Verilog-FIR)，至於 firmware 都是 wb_addr_i 0x38 開頭也就是去拿資料的部分(exmem-FIR)




The designs are included in user project wrapper

我們先專注於 WB & Verilog-FIR 之間的互動。可以大致上得到這個 block diagram：黑色的箭頭是 data path，藍色的箭頭是 control signal。



Protocol 的表現上都是參考 workbook 的波型圖 (wishbone 要轉成 axi 向我們 design 發送 r/w data)：

AXI write & AXI-read：

Figure: AXI4-Lite Write Timing Diagram

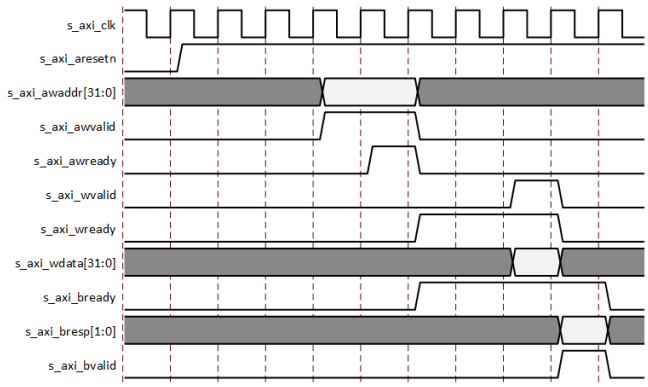
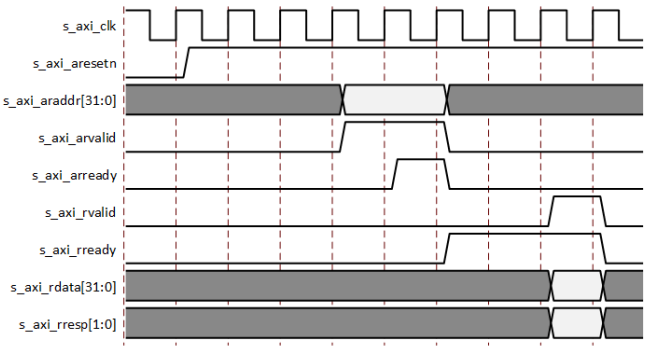
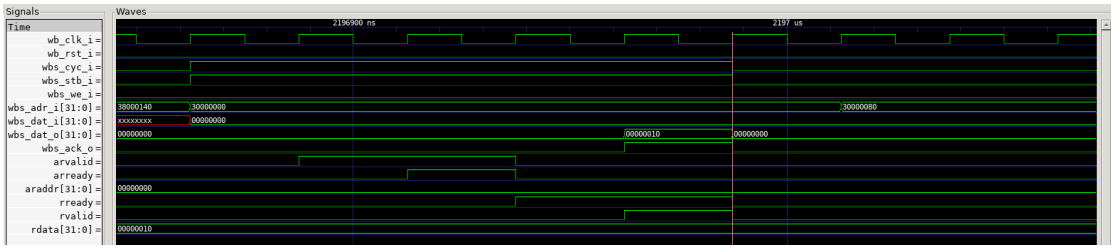



Figure: AXI4-Lite Read Timing Diagram

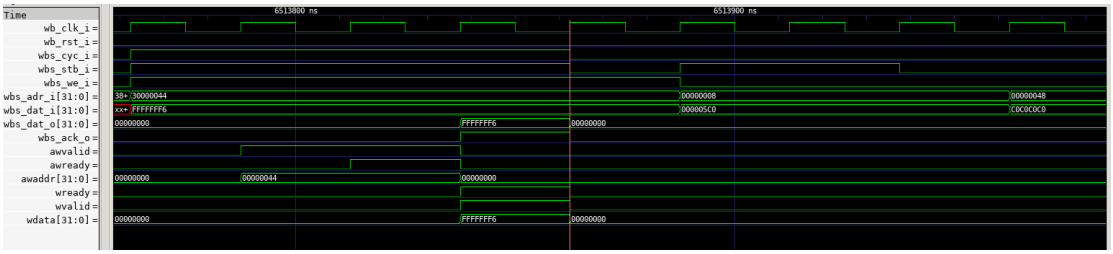



- # Wishbone & Design interactions waveforms:

Wishbone to Axi-lite read:

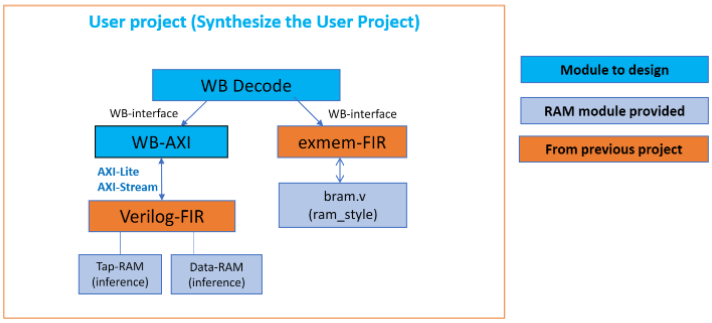



Wishbone to Axi-lite write:



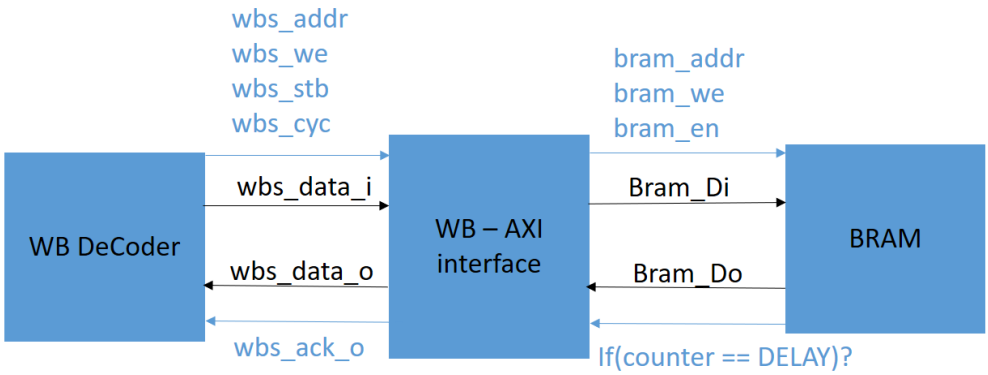
- Design block diagram and protocol between WB & exmem-FIR (firmware)

橘色框框中右半邊的部分就是韌體(exmem-FIR)，wbs_addr_i 以 0x38 為開頭

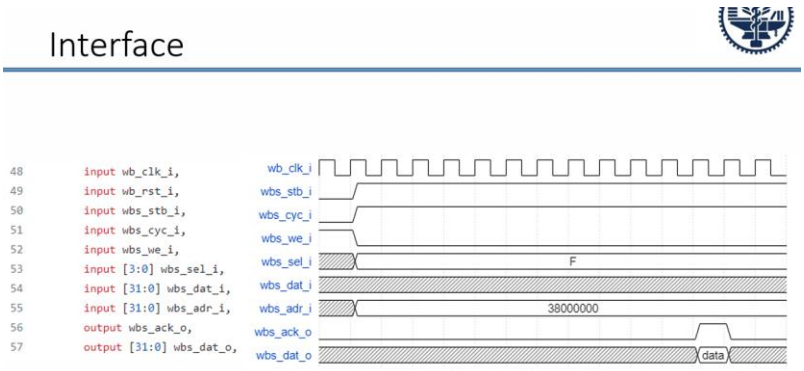



The designs are included in user project wrapper

可以大致上將 control 訊號和 data 訊號整理為下圖：

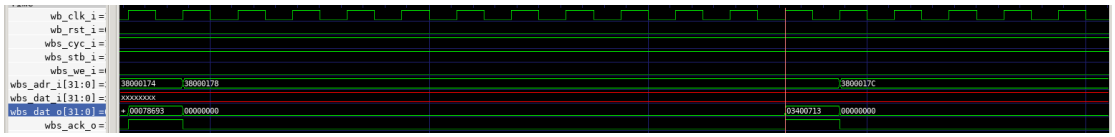



User 和 firmware 用到的 wishbone interface 都遵守下圖的 behavior：

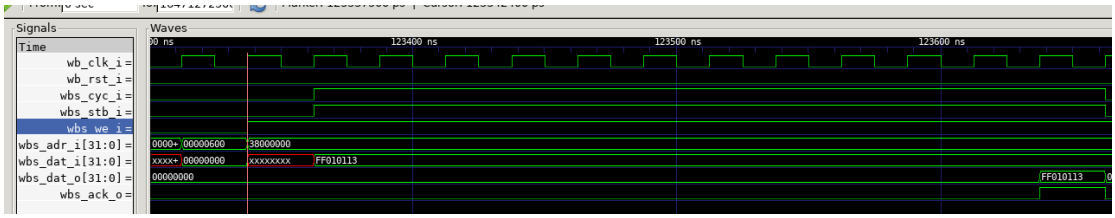


- Wishbone & Firmware interactions waveforms:

Firmware read ：

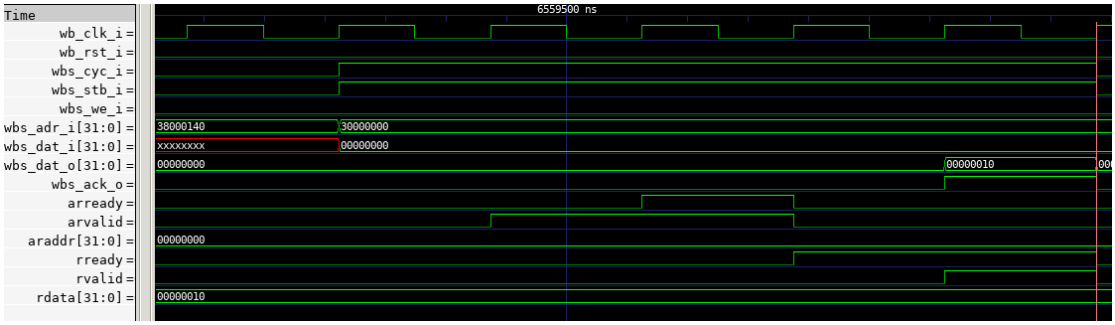



Firmware write：

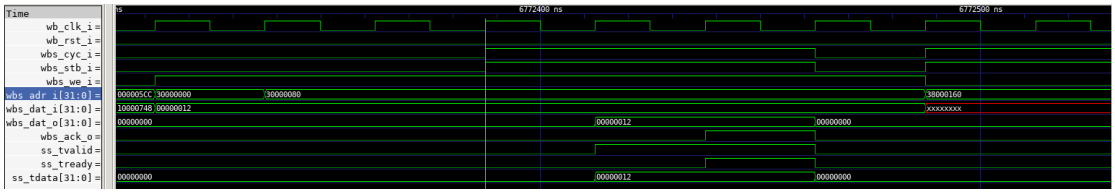



Wishbone to axi stream input

先用 WB→AXI-LITE-read 確定 ap[4]是不是 1：(5 cycle)

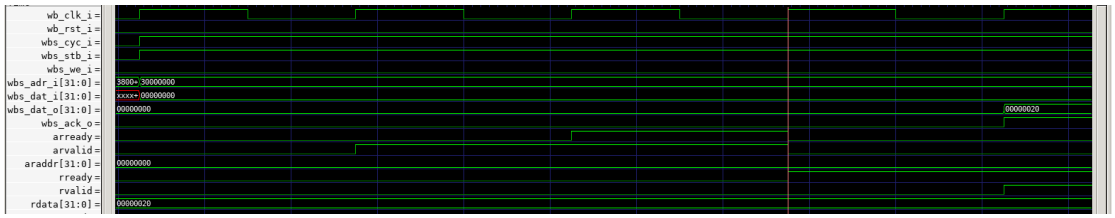



再用 30000080 的 wbs_addr 輸入 data (Xn): (3 cycle)

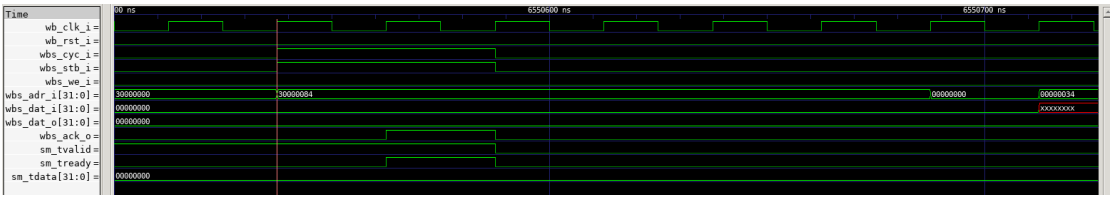



Wishbone to AXI stream output ：

先用 WB→AXI-LITE-read 確定 AP[5]是不是 1：(5 cycle)




再用 30000084 的 wbs_addr 將 data(Yn)讀出：(3 cycle)




- FIR engine theoretical throughput versus actual throughput

一共有 64 個 output data，每個 data 需要經過 11 次乘法和 10 次加法，所以總共的 operation 為 21*64 = 1344。

理論上的需要時間可以簡單計算：將 axi-lite 的一次寫入或讀出當作 4 個 cycle，2 for address handshake，2 for data handshake。一次 axi-stream 的寫入或讀出當作 1 個 cycle。

Overall operation：輸入 11 個 tap bram data，算一次 output 需要一個 input cycle，11 個計算 cycle 及一個 output cycle。所以總時間為 12*5 (data length & 11 tap parameters) + 5 (ap_start = 1) + 64*(11 (fir)+5 +3+5+3) = 1793 個 cycle。這樣是算出 64 筆 fir output 所需要的 cycle 數。

```
// start timer
reg_mprj_datal = (0xA5 << 16);

//write down your fir
for (int i=0; i<64; i++) {
    while ((AP >> 4) & 1 != 1) {
        // wait until Xn [4] is ready
    }
    Xn = i;

    while ((AP >> 5) & 1 != 1) {
        // wait until Yn [5] is ready
    }
    // ans = Yn;
    outputsignal[i] = Yn;
}

// output to mprj[31:24], and stop timer
reg_mprj_datal = ((outputsignal[63] & 0xFF) << 24) | (0x5A << 16);
```

```
LA Test 1 started
Start timer:              1548788
End timer:                2360563
```

實際情況則是(2360563 - 1548788) / 25 = 32471 cycle，會造成這個情況可以從波型看到每次 stream out 的時候都要等 wishbone interface 處理完韌體的 operation，再透過 AXI-lite 確認能不能讀出，才能夠真的輸出 data，一來一回額外花了非常多的 cycle。

- ## What is latency for firmware to feed data

首先，wishbone 需要先發起 axi-lite:read 去看 ap[4]是否為 1，這樣會需要花上 5 cycle 如上 waveform，再來才是用 wb→axi-stream 的方式送 Xn，需要花 3 cycle 如上面的 waveform，所以 firmware 送一筆 Xn 到 design 至少會需要花上 8 cycle。

- ## Techniques to improve throughput

1. 在撰寫 counter_la_tb.v 的時候，有發現用 while 迴圈和 wait 迴圈實際上量出來的 start timer 和 end timer 會不太一樣，三次 FIR 可以省近 2000 個 cycle。

2. Tap parameters 其實並不用每次計算 fir 都重新送。這樣可以省下前面許多 axi-lite:write (5 cycle)送進 design 所花費的時間。

3. Design 用 wishbone protocol 輸出。

4. Design 開 11 個乘法器。

5. 直接宣告 tap_0 = 0;而非 tap_0 = taps[0];這樣可以少 load 的 delay。

- ## Other methods to improve performance

其實想把 throughput 提升，最直接的方法就是將韌體和 user 的 interface 分開，而不是共用一個 wishbone interface。

每個 FIR 算完要 OUTPUT 的時候往往 wishbone interface 都在處理韌體相關的 protocol，導致 output state 都要 hang 上數十個 cycle，如果有辦法做到像 lab3 那樣的 throughput 的話想必能大大提高整體的 performance。

或者是直接我們的 Design 也不要採用 Axi 的 protocol 全部採用 wishbone，這樣也可以快上許多。

- ## Bram12 better?

我並不這麼認為，因為我可以做到 11 cycle 作完 11 個 tap coeff 的 FIR 運算，這也是最快的了，所以 bram12 並不會為我的 design 加速，但對其他的演算方法可能會快一些。

## ● Synthesis information

Synthesis report 檔案連結:
https://github.com/holyuming/NYCU-2023-SoCLab/blob/master/lab-caravel_fir/synthesis/synthesis.runs/synth_1/user_project_wrapper_utilization_synth.rpt

我們是對整個 user_project_wrapper 去進行合成。包含了 user_project_counter.v & bram.v & design.v

```
1. CLB Logic
------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| CLB LUTs*               |  406 |     0 |          0 |    230400 |  0.18 |
|   LUT as Logic          |  406 |     0 |          0 |    230400 |  0.18 |
|   LUT as Memory         |    0 |     0 |          0 |    101760 |  0.00 |
| CLB Registers           |  219 |     0 |          0 |    460800 |  0.05 |
|   Register as Flip Flop |  219 |     0 |          0 |    460800 |  0.05 |
|   Register as Latch     |    0 |     0 |          0 |    460800 |  0.00 |
| CARRY8                  |   12 |     0 |          0 |     28800 |  0.04 |
| F7 Muxes                |    0 |     0 |          0 |    115200 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     57600 |  0.00 |
| F9 Muxes                |    0 |     0 |          0 |     28800 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
```

BRAM：
因為 interface 那邊連接了三個 bram：user project 的 tap 和 data bram，以及韌體的 bram。

```
2. BLOCKRAM
-----------

+-------------------+------+-------+------------+-----------+-------+
|     Site Type     | Used | Fixed | Prohibited | Available | Util% |
+-------------------+------+-------+------------+-----------+-------+
| Block RAM Tile    |  1.5 |     0 |          0 |       312 |  0.48 |
|   RAMB36/FIFO*    |    0 |     0 |          0 |       312 |  0.00 |
|   RAMB18          |    3 |     0 |          0 |       624 |  0.48 |
|     RAMB18E2 only |    3 |       |            |           |       |
| URAM              |    0 |     0 |          0 |        96 |  0.00 |
+-------------------+------+-------+------------+-----------+-------+
```

Arithemetic & IO

```
3. ARITHMETIC
-------------

+----------------+------+-------+------------+-----------+-------+
|    Site Type   | Used | Fixed | Prohibited | Available | Util% |
+----------------+------+-------+------------+-----------+-------+
| DSPs           |    3 |     0 |          0 |      1728 |  0.17 |
|   DSP48E2 only |    3 |       |            |           |       |
+----------------+------+-------+------------+-----------+-------+


4. I/O
------

+------------+------+-------+------------+-----------+-------+
|  Site Type | Used | Fixed | Prohibited | Available | Util% |
+------------+------+-------+------------+-----------+-------+
| Bonded IOB |  297 |     0 |          0 |       360 | 82.50 |
+------------+------+-------+------------+-----------+-------+
```

因為將 design.v 的 fir module 也加入合成，所以有使用這三個 DSP 乘法器。

Primitives：

```
8. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| OBUFT    |  195 |                 I/O |
| LUT6     |  177 |                 CLB |
| FDCE     |  158 |            Register |
| LUT4     |   92 |                 CLB |
| LUT2     |   90 |                 CLB |
| LUT5     |   77 |                 CLB |
| INBUF    |   69 |                 I/O |
| IBUFCTRL |   69 |              Others |
| LUT3     |   65 |                 CLB |
| FDRE     |   55 |            Register |
| OBUF     |   33 |                 I/O |
| CARRY8   |   12 |                 CLB |
| FDPE     |    5 |            Register |
| RAMB18E2 |    3 |           BLOCKRAM |
| DSP48E2  |    3 |          Arithmetic |
| LUT1     |    1 |                 CLB |
| FDSE     |    1 |            Register |
| BUFGCE   |    1 |               Clock |
+----------+------+---------------------+
```

- Timing report:

Timing report 檔案連結: https://github.com/holyuming/NYCU-2023-SoCLab/blob/master/lab-caravel_fir/synthesis/timing_report.txt

```
Clock     Waveform(ns)         Period(ns)      Frequency(MHz)
-----     ------------         ----------      --------------
wb_clk_i  {0.000 2.500}        5.000           200.000
```

Period = 5ns

```
---------------------------------------------------------------------------------------------------
From Clock:  wb_clk_i
  To Clock:  wb_clk_i

Setup :            0  Failing Endpoints,  Worst Slack        0.159ns,  Total Violation        0.000ns
Hold  :           NA  Failing Endpoints,  Worst Slack           NA  ,  Total Violation           NA
PW    :            0  Failing Endpoints,  Worst Slack        1.958ns,  Total Violation        0.000ns
---------------------------------------------------------------------------------------------------
```

Slack Met, no timing violations