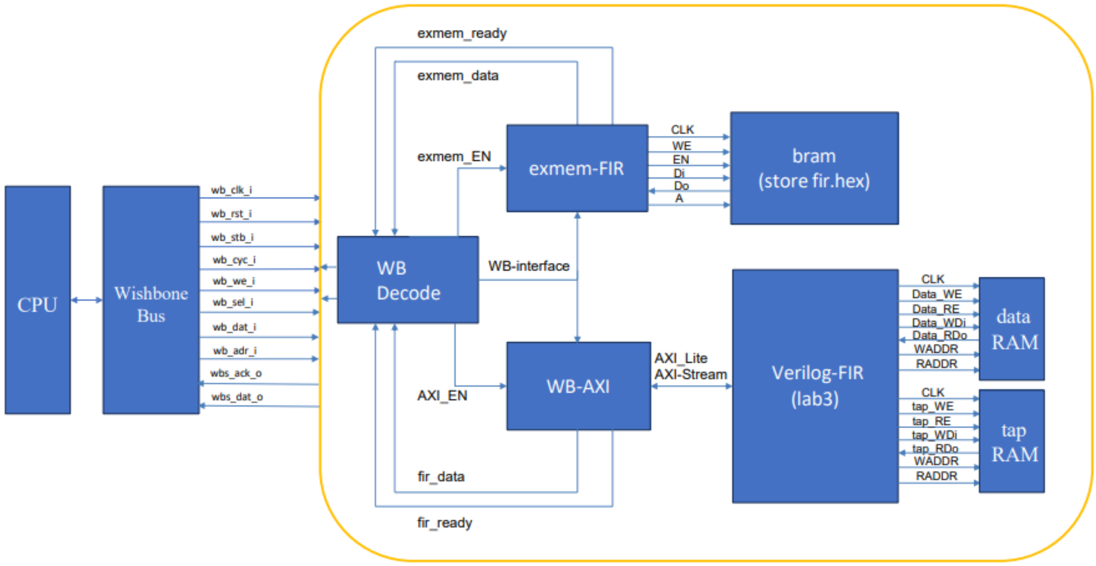
112061603 李柏叡
112061620 郭邑哲
112061621 貢暐家

SOC Design Lab4-2

- Block Diagram



User Project Wrapper

- The interface protocol between firmware, user project and testbench

以 Tap 為例: CPU 根據 FIRMWARE 的指令，將 tap 值藉由 Wishbone interface(wbs_data_i)給 user project(FIR)存到 tap Ram，firmware outputs a Start Mark (0XA5) on mprj[23:16] to notify Testbench。

以 input x 為例: CPU 根據 FIRMWARE 的指令，將 x 值藉由 Wishbone interface(wbs_data_i)給 user project(FIR)存到 data Ram，經過 fir 運算後，CPU 根據 FIRMWARE 的指令接收 y，再根據 FIRMWARE 的指令將最後一個 Y 透過 mprj 給 TESTBENCH。

Firmware code 和 testbench 之間溝通的 interface 為 mprj；Firmware code 和 user project 之間溝通的 interface 為 wishbone。

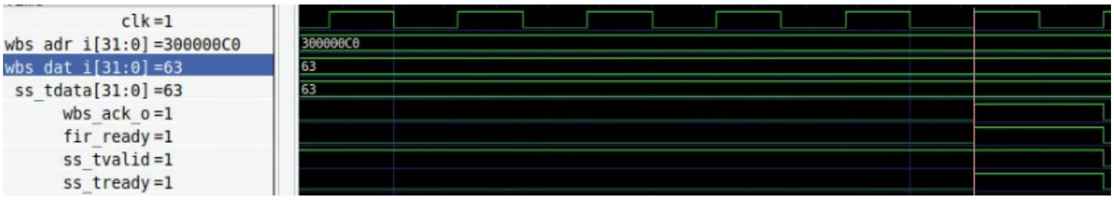
- Waveform and analysis of the hardware/software behavior.

Software:輸入 64 個 X 值，分別為 0~63 到地址 0X300000C0，透過 FIRMWARE 指令，CPU 將 X 輸入給 FIR。

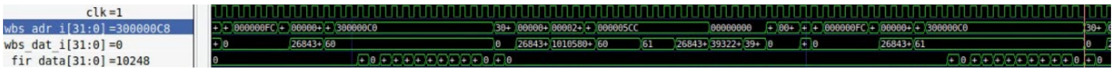
<Input x>

First 0 :

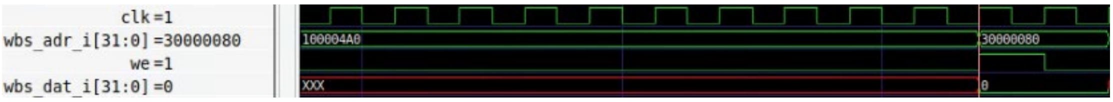
Last 63 :



<output y>

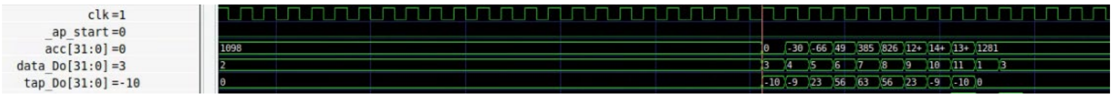


<tap>



Hardware:

下圖為 FIR 運算中，相乘相加的結果



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

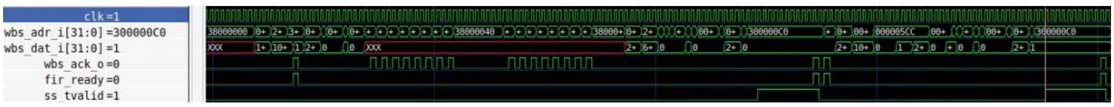
  theoretical throughput:11cycle(per output data y)

  Actually measured throughput:52cycle(per output data y)

- What is latency for firmware to feed data?

  CPU 藉由 firmware 傳 0 到 fir 接收 0 的時間為: 12cycle

Firmware feed data 0 到 feed data 1 的時間為:52cycle



- 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，因為此次 lab 並不需要儲存計算後的 Y 值。

- Can you suggest other method to improve the performance?

第一版:

```
int* __attribute__ ( ( section ( ".mprjram" ) ) ) fir(){
        initfir();
        int x;
        for(n = 0; n < 63; n++) {
            reg_user_x = n;
            outputsignal[n] = reg_user_y;
        }
        reg_user_lastx = n;
        outputsignal[n] = reg_user_y;
        return outputsignal;
}
```

```
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.
LA1 Test 1 started
LA1 Test 2 passed:      9611cycles
LA2 Test 1 started
LA2 Test 2 passed:      9611cycles
LA3 Test 1 started
LA3 Test 2 passed:      9611cycles
```

第二版:

```
int __attribute__ ( ( section ( ".mprjram" ) ) ) fir(){
        initfir();
        int n;
        int y;
        for(n = 0; n < 63; n++) {
            reg_user_x = n;
            reg_user_y;
        }
        reg_user_x = n;
        y = reg_user_y;
        return y;
}
```

```
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.
LA1 Test 1 started
LA1 Test 2 passed:      5835cycles
LA2 Test 1 started
LA2 Test 2 passed:      4775cycles
LA3 Test 1 started
LA3 Test 2 passed:      4775cycles
```

我們將 fir.c 檔，原本輸出 Y 會存到記憶體，但因為中間的 Y 我們並不需要做驗證，因此改成輸出 Y 不會存到記憶體，只有最後一筆 Y 會需要儲存到記憶體。

- Any other insights ?

1.

原本:

0x10-13 - data-length

0x40-7F – Tap parameters, (e.g., 0x40-43 Tap0, in sequence …)

0x80-83 – X[n] input (r/w)

0x84-87 – Y[n] output (ro)

改良:

0X10-13 – data_length

0X80-0XA8 – Tap parameters

0XC0-0XC3 – X[n](r/w)

0XC8-0XCB – Y[n](ro)

0X00 – config(r/w)

```
#define reg_user_tap0   (*(volatile uint32_t*)0x30000080)
#define reg_user_tap1   (*(volatile uint32_t*)0x30000084)
#define reg_user_tap2   (*(volatile uint32_t*)0x30000088)
#define reg_user_tap3   (*(volatile uint32_t*)0x3000008C)
#define reg_user_tap4   (*(volatile uint32_t*)0x30000090)
#define reg_user_tap5   (*(volatile uint32_t*)0x30000094)
#define reg_user_tap6   (*(volatile uint32_t*)0x30000098)
#define reg_user_tap7   (*(volatile uint32_t*)0x3000009C)
#define reg_user_tap8   (*(volatile uint32_t*)0x300000A0)
#define reg_user_tap9   (*(volatile uint32_t*)0x300000A4)
#define reg_user_tap10  (*(volatile uint32_t*)0x300000A8)
#define reg_user_x      (*(volatile uint32_t*)0x300000C0)
#define reg_user_y      (*(volatile uint32_t*)0x300000C8)
#define reg_user_config (*(volatile uint32_t*)0x30000000)
#define reg_user_len    (*(volatile uint32_t*)0x30000010)
```

透過直接將 Tap 地址從 0X80 開始，可以使我們 Gate Counts 下降。

2.

以下兩張圖為將 Lab4-2 與 Lab4-1 做比較，而輸入 X 統一設定輸入 11 筆資料。可以發現 FIR 運算以硬體的方式(Lab4-2)所需之 Cycle 較以軟體的方式(Lab4-1)所需之 Cycle 少。

```
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.
LA1 Test 1 started
LA1 Test 2 passed:      3703cycles
```

```
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.
LA Test 1 started
LA Test 2 passed     169343 cycle
```

- Synthesis report

LUT and FF :

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	402	0	0	53200	0.76
LUT as Logic	346	0	0	53200	0.65
LUT as Memory	56	0	0	17400	0.32
LUT as Distributed RAM	56	0			
LUT as Shift Register	0	0			
Slice Registers	241	0	0	106400	0.23
Register as Flip Flop	209	0	0	106400	0.20
Register as Latch	32	0	0	106400	0.03
F7 Muxes	0	0	0	26600	0.00
F8 Muxes	0	0	0	13300	0.00

RTL Component :

```
Detailed RTL Component Info :
+---Adders :
           2 Input   32 Bit       Adders := 1
           3 Input    6 Bit       Adders := 1
           2 Input    4 Bit       Adders := 3
+---Registers :
                      32 Bit     Registers := 6
                       4 Bit     Registers := 2
                       2 Bit     Registers := 2
                       1 Bit     Registers := 6
+---Multipliers :
                     32x32  Multipliers := 1
+---RAMs :
                     64K Bit   (2048 X 32 bit)          RAMs := 1
                     352 Bit   (11 X 32 bit)          RAMs := 2
+---Muxes :
           2 Input   32 Bit        Muxes := 14
           4 Input   32 Bit        Muxes := 2
           5 Input   32 Bit        Muxes := 1
           2 Input   12 Bit        Muxes := 1
           2 Input    8 Bit        Muxes := 3
           3 Input    8 Bit        Muxes := 1
           3 Input    4 Bit        Muxes := 1
           2 Input    4 Bit        Muxes := 3
           5 Input    3 Bit        Muxes := 1
           2 Input    2 Bit        Muxes := 7
           3 Input    2 Bit        Muxes := 1
           2 Input    1 Bit        Muxes := 14
           3 Input    1 Bit        Muxes := 3
           4 Input    1 Bit        Muxes := 1
```

Report Cell Usage :

```
Report Cell Usage:
+------+---------+------+
|      |Cell     |Count |
+------+---------+------+
|1     |BUFG     |     2|
|2     |CARRY4   |    35|
|3     |DSP48E1  |     3|
|4     |LUT1     |     2|
|5     |LUT2     |    97|
|6     |LUT3     |    28|
|7     |LUT4     |   126|
|8     |LUT5     |    46|
|9     |LUT6     |   107|
|10    |RAM16X1S |    32|
|11    |RAM32M   |     5|
|12    |RAM32X1D |     2|
|13    |RAMB36E1 |     2|
|14    |FDRE     |   207|
|15    |FDSE     |     2|
|16    |LD       |    32|
|17    |IBUF     |    68|
|18    |OBUF     |   112|
|19    |OBUFT    |   128|
+------+---------+------+
```

Max delay path:

```
Slack (MET) :             4.440ns  (required time - arrival time)
  Source:                 FIR/tap_idx_reg[0]/C
                            (rising edge-triggered cell FDRE clocked by wb_clk_i  {rise@0.000ns fall@12.500ns period=25.000ns})
  Destination:            wbs_dat_o[31]
                            (output port clocked by wb_clk_i  {rise@0.000ns fall@12.500ns period=25.000ns})
  Path Group:             wb_clk_i
  Path Type:              Max at Slow Process Corner
  Requirement:            25.000ns  (wb_clk_i rise@25.000ns - wb_clk_i rise@0.000ns)
  Data Path Delay:        17.145ns  (logic 11.635ns (67.867%)  route 5.509ns (32.133%))
  Logic Levels:           14  (CARRY4=5 DSP48E1=2 LUT2=1 LUT4=3 LUT6=2 OBUF=1)
  Output Delay:           0.000ns
  Clock Path Skew:        -3.380ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    0.000ns = ( 25.000 - 25.000 )
    Source Clock Delay      (SCD):    3.380ns
    Clock Pessimism Removal (CPR):    0.000ns
  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
```

Slack met

```
                     (clock wb_clk_i rise edge)
                                                 25.000    25.000 r
                     clock pessimism              0.000    25.000
                     clock uncertainty           -0.035    24.965
                     output delay                -0.000    24.965
-------------------------------------------------------------------
                     required time                         24.965
                     arrival time                         -20.525
-------------------------------------------------------------------
                     slack                                  4.440
```

- Metrics to measure the fir system
  Metrics=4775*20.525*(346+241)= 57530035.625

Github link : https://github.com/816-allen/SOC-Design-Lab4-2?search=1