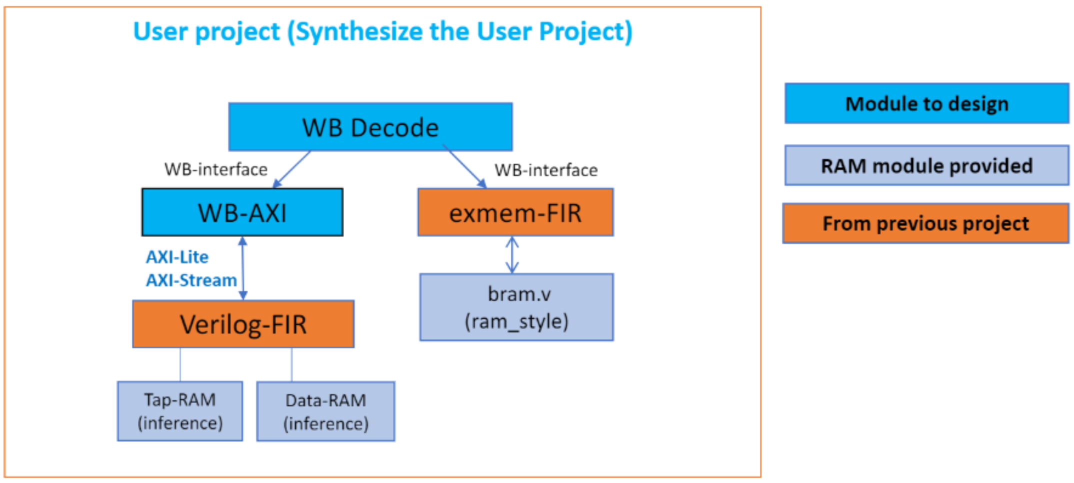
SoC Design Lab 4 Report

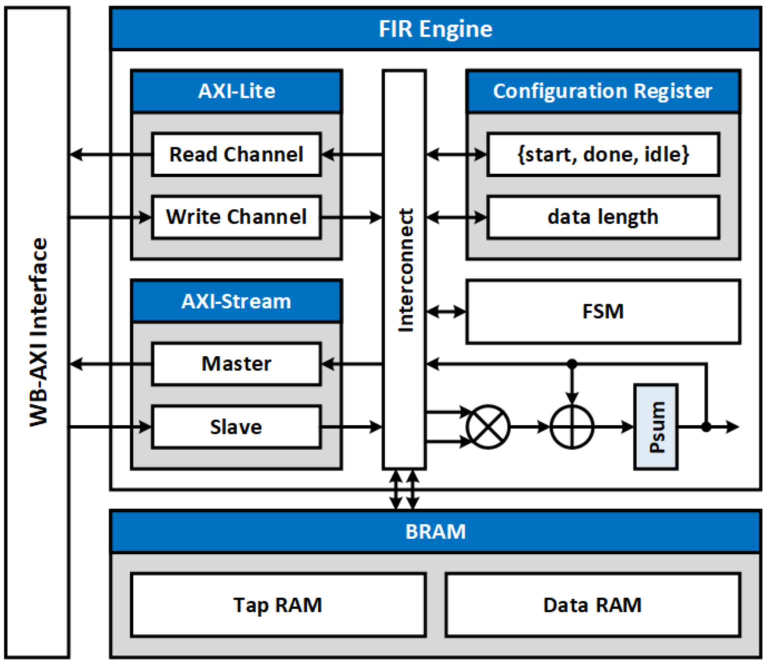
R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元

1. Block diagram

(1) User project



(2) Verilog-FIR



Configuration register address map:

Base Address	Offset	Bit	Description
0x3000_0000	0x00	0	ap_start: set to 1 to start the FIR engine
		1	ap_done: assert when FIR engine processes and transfers all the data
		2	ap_idle: indicate whether FIR engine is actively processing data
		3	Reserved zero

		4	FIR engine is ready to accept input x[n]
		5	Output y[n] is ready to be read
	0x10 – 0x13	31:0	Data length
	0x40 – 0x7F	31:0	Tap coefficients
	0x80 – 0x83	31:0	Input x[n]
	0x84 – 0x87	31:0	Output y[n]
0x3800_0000			Execution memory that stores firmware code

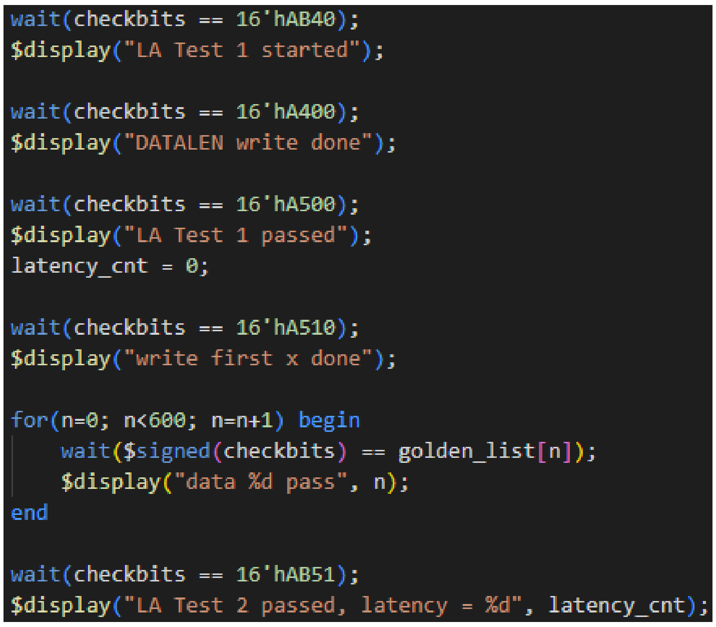
2. System description

(1) Firmware control

- ✓ Configure the mprj port (i.e. select output from CPU or user project)
- ✓ Define MMIO registers
- ✓ Program tap coefficients and data length into FIR engine
- ✓ CPU sends x[n] to FIR engine if FIR engine is ready to accept inputs
- ✓ CPU receives y[n] from FIR engine if the output of FIR engine is ready to be read

(2) Testbench

Testbench checks the execution progress and correctness by monitoring the mprj port, and prints out the latency timer after the calculation is completed.



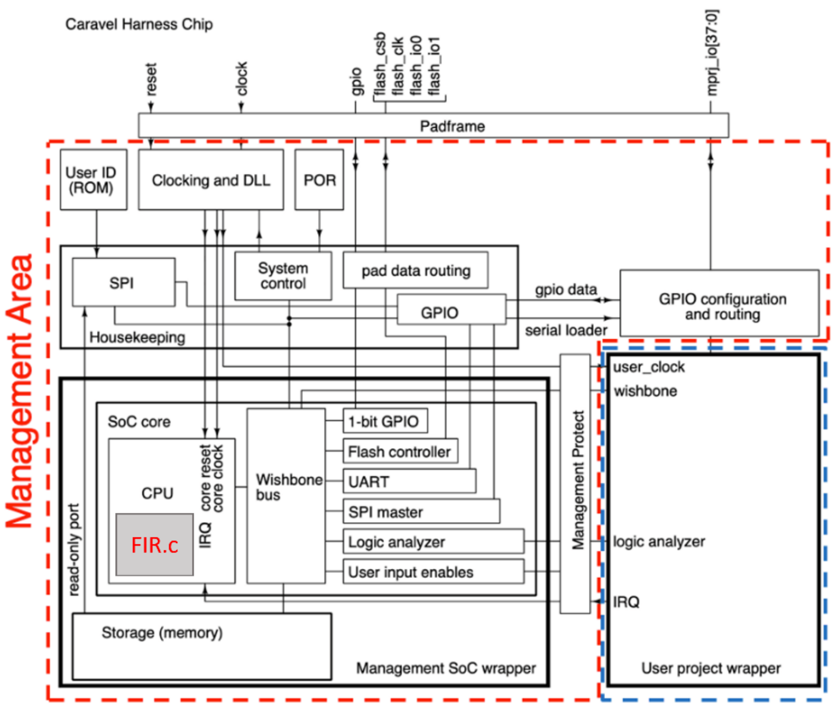
```
wait(checkbits == 16'hAB40);
$display("LA Test 1 started");

wait(checkbits == 16'hA400);
$display("DATALEN write done");

wait(checkbits == 16'hA500);
$display("LA Test 1 passed");
latency_cnt = 0;

wait(checkbits == 16'hA510);
$display("write first x done");

for(n=0; n<600; n=n+1) begin
    wait($signed(checkbits) == golden_list[n]);
    $display("data %d pass", n);
end

wait(checkbits == 16'hAB51);
$display("LA Test 2 passed, latency = %d", latency_cnt);
```

(3) User project

Lab4-1 (SW)

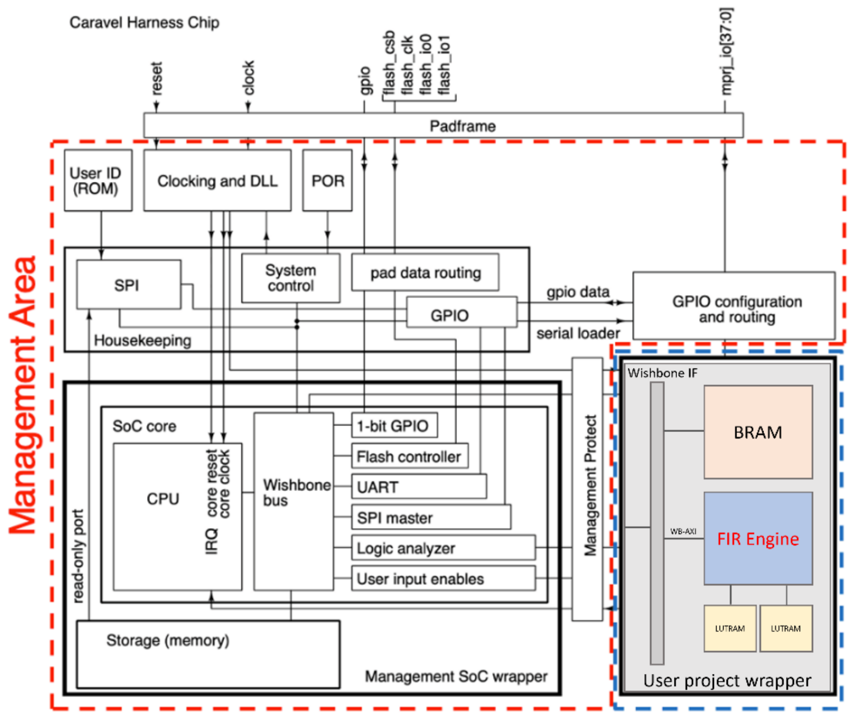
- ✓ The compiled firmware code fir.hex is loaded into execution memory in user project area
- ✓ CPU reads firmware code from user project and calculates $y[n] = \sum_{i=0}^{10} h[i] \times x[n-i]$

R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元



Lab4-2 (SW-HW)

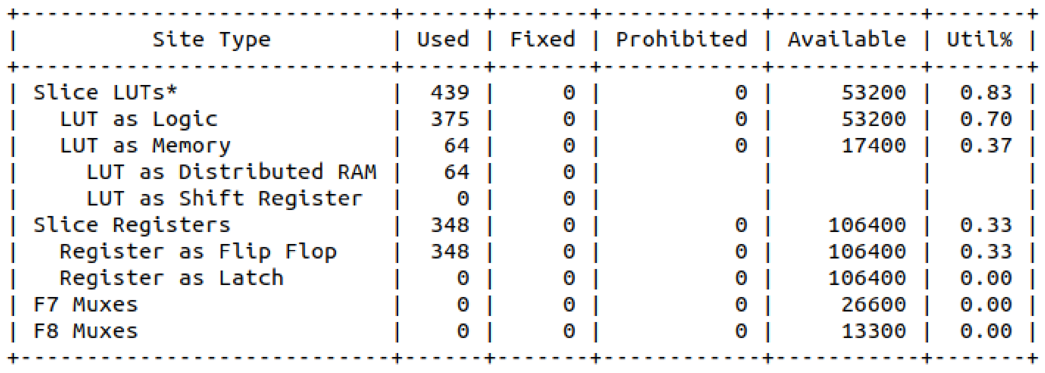
- ✓ Integrate Lab3 Verilog-FIR and Lab4-1 exmem-FIR into user project area
- ✓ Decode wishbone transaction address and dispatch to exmem-FIR or WB-AXI interface
- ✓ WB-AXI interface: conversion between Wishbone, AXI-Lite and AXI-Stream
- ✓ Use firmware to move data in/out and control the FIR engine
- ✓ FIR is calculated by the hardware accelerator in user project area



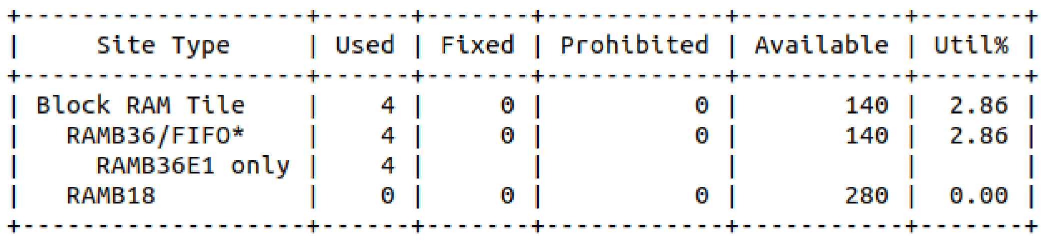
R12943003 劉冠亨　　R12943016 謝言鼎　　R11943006 張力元

3. Resource usage

Utilization report including FF, LUT and BRAM:



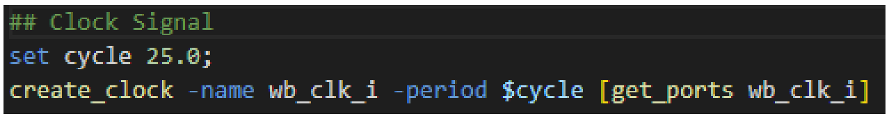
```
+----------------------------+------+-------+------------+-----------+-------+
|          Site Type         | Used | Fixed | Prohibited | Available | Util% |
+----------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*                |  439 |     0 |          0 |     53200 |  0.83 |
|   LUT as Logic             |  375 |     0 |          0 |     53200 |  0.70 |
|   LUT as Memory            |   64 |     0 |          0 |     17400 |  0.37 |
|     LUT as Distributed RAM |   64 |     0 |            |           |       |
|     LUT as Shift Register  |    0 |     0 |            |           |       |
| Slice Registers            |  348 |     0 |          0 |    106400 |  0.33 |
|   Register as Flip Flop    |  348 |     0 |          0 |    106400 |  0.33 |
|   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 |
+----------------------------+------+-------+------------+-----------+-------+
```



```
+-------------------+------+-------+------------+-----------+-------+
|     Site Type     | Used | Fixed | Prohibited | Available | Util% |
+-------------------+------+-------+------------+-----------+-------+
| Block RAM Tile    |    4 |     0 |          0 |       140 |  2.86 |
|   RAMB36/FIFO*    |    4 |     0 |          0 |       140 |  2.86 |
|     RAMB36E1 only |    4 |       |            |           |       |
|   RAMB18          |    0 |     0 |          0 |       280 |  0.00 |
+-------------------+------+-------+------------+-----------+-------+
```

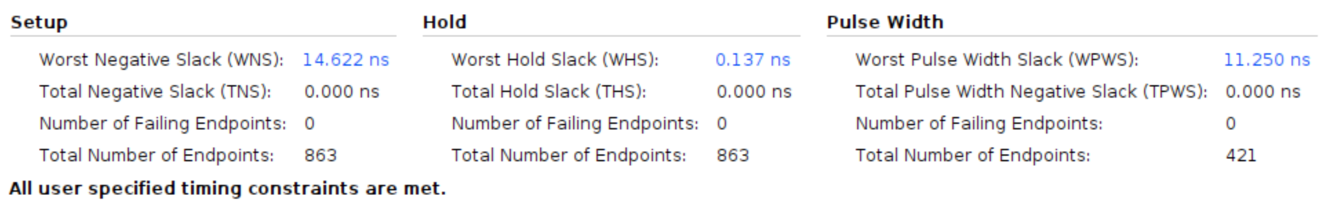
4. Timing report

Clock constraint for synthesis:



```
## Clock Signal
set cycle 25.0;
create_clock -name wb_clk_i -period $cycle [get_ports wb_clk_i]
```

Setup time and hold time slack:



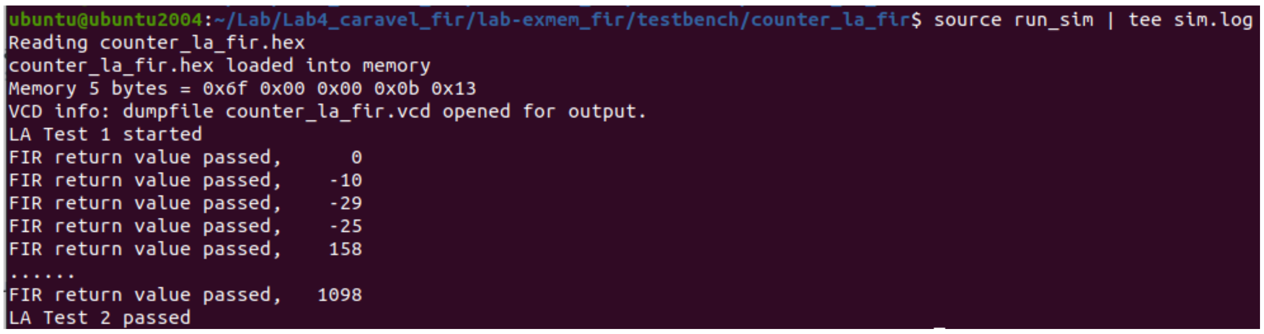
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	14.622 ns	Worst Hold Slack (WHS):	0.137 ns	Worst Pulse Width Slack (WPWS):	11.250 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	863	Total Number of Endpoints:	863	Total Number of Endpoints:	421

All user specified timing constraints are met.

5. Simulation results

(1) Simulation log

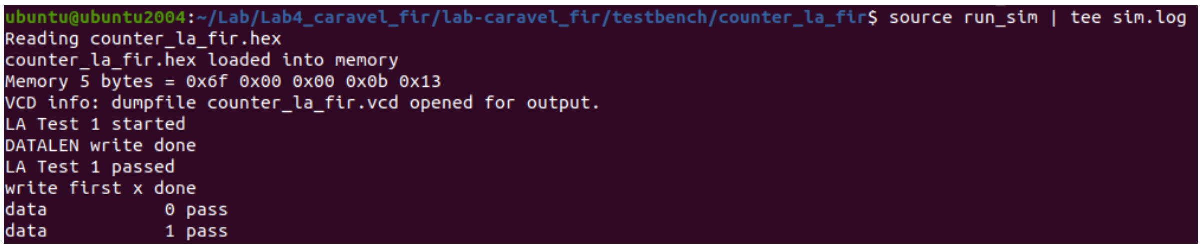
Lab4-1



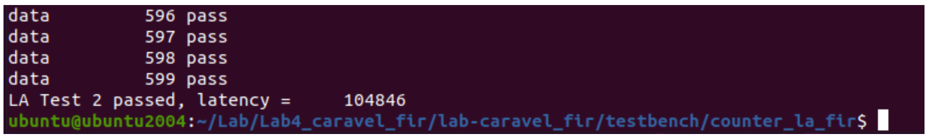
```
ubuntu@ubuntu2004:~/Lab/Lab4_caravel_fir/lab-exmem_fir/testbench/counter_la_fir$ source run_sim | tee sim.log
Reading counter_la_fir.hex
counter_la_fir.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_fir.vcd opened for output.
LA Test 1 started
FIR return value passed,      0
FIR return value passed,    -10
FIR return value passed,    -29
FIR return value passed,    -25
FIR return value passed,    158
......
FIR return value passed,   1098
LA Test 2 passed
```

R12943003 劉冠亨　　R12943016 謝言鼎　　R11943006 張力元

Lab4-2



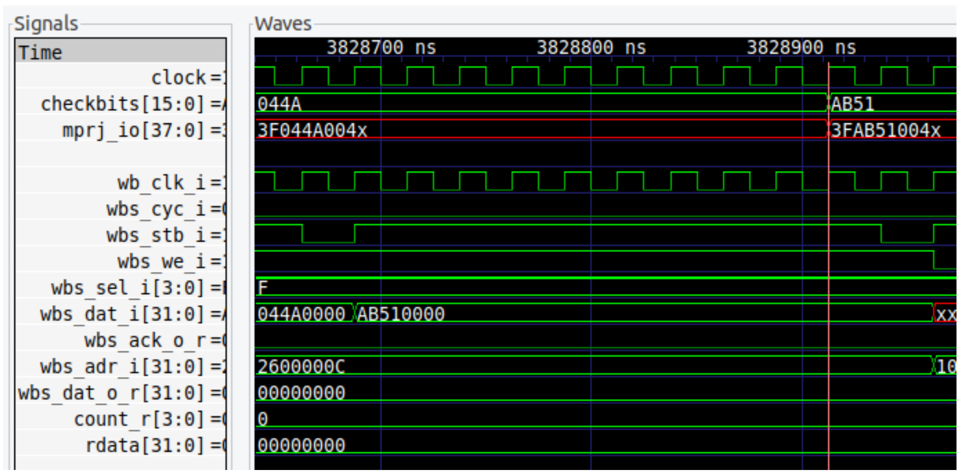
```
ubuntu@ubuntu2004:~/Lab/Lab4_caravel_fir/lab-caravel_fir/testbench/counter_la_fir$ source run_sim | tee sim.log
Reading counter_la_fir.hex
counter_la_fir.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_fir.vcd opened for output.
LA Test 1 started
DATALEN write done
LA Test 1 passed
write first x done
data           0 pass
data           1 pass
```



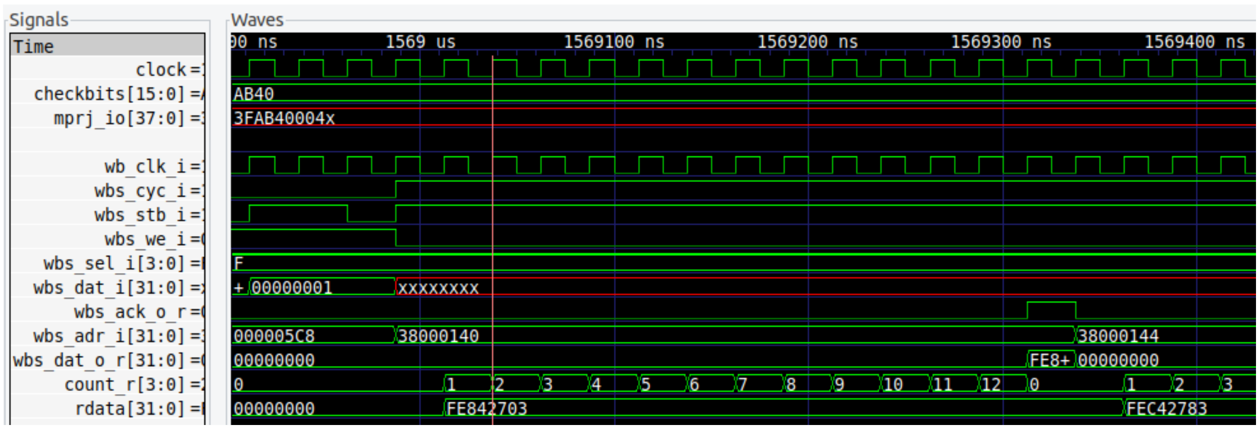
```
data         596 pass
data         597 pass
data         598 pass
data         599 pass
LA Test 2 passed, latency =      104846
ubuntu@ubuntu2004:~/Lab/Lab4_caravel_fir/lab-caravel_fir/testbench/counter_la_fir$
```

(2) Waveform

mprj I/O monitoring



WB and execution memory



6. Discussion

(1) Interface protocol between firmware, user project and testbench

在 fir.c 中，使用 mmio 定義好 fir 的 input、output 等。先將 data_len、taps 傳輸進 fir engine 中，等待 ap_idle，即可將 ap_start 拉高，開始傳輸 input 並接收 output。使用的 input 為 lab3 的三角波，在傳輸一筆資料後便會接收一筆 output data。在過程中，不同步驟完成後有使用

到 reg_mprj_datal，將其設為特定 pattern，在 testbench 接收到這些 pattern 就可以知道步驟已經順利完成。

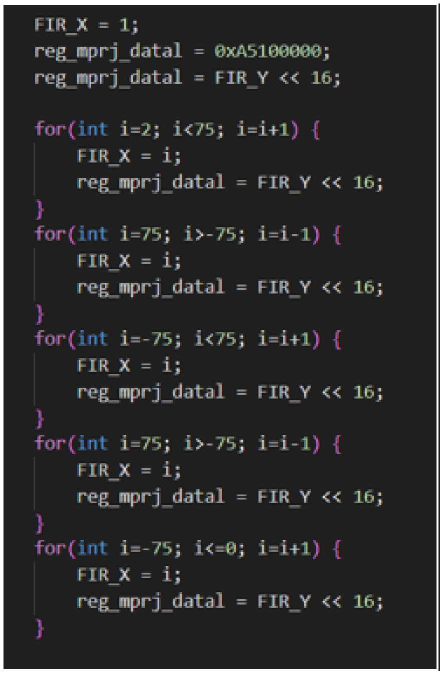
而 testbench 寫法則更為簡單，先 check firmware 中傳的特定 pattern 都有送出，開始計算 cycle，並等待每一筆 output，確認結果與 golden 相同，在最後一筆成功產生後就可停止。

fir.c 與 lab3 所使用的一樣，不過在 lab3 中 testbench 的寫法會讓 ready 一直拉為 high，因此需要稍微修改 AXI protocol 的相關邏輯才可順利運作。

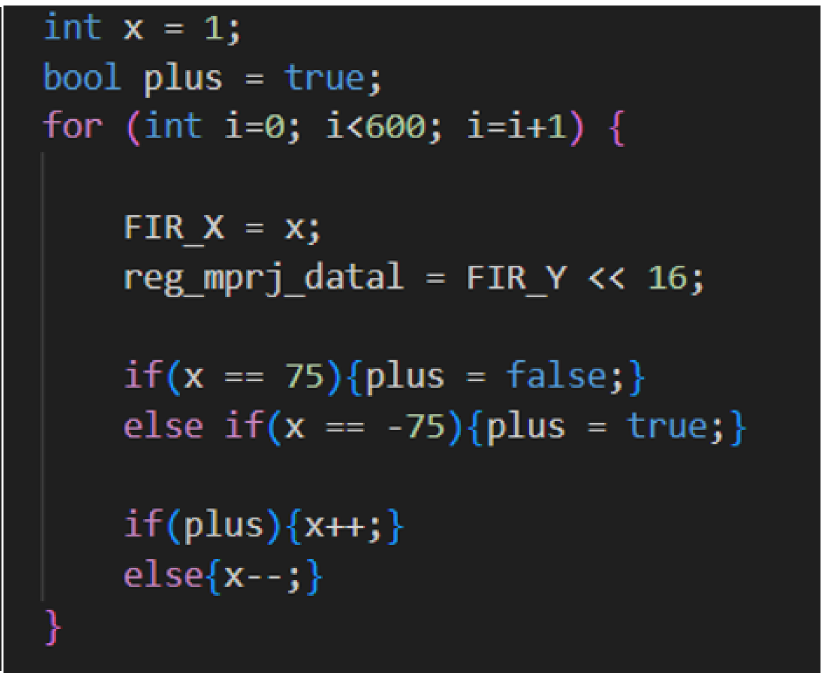
(2) Throughput

在理想上，fir.v 的寫法若在 engine ready 時外部都可以寫值與讀值，沒有任何 delay 的話，throughput 會是 12 個 cycle 可以算出一個結果，也就是 0.083 outputs/cycle。

但實際上，受限於韌體的運算時間以及 data 傳輸時的 delay，運行的結果為 600/104846 = 0.0057 outputs/cycle。經過測試，firmware 的寫法會大大影響 performance，而 engine 的表現其實並不重要。下圖為兩種三角波的計算方式，左圖為右圖展開一部分 for loop，也是本次使用的寫法，若使用右圖的方法則整體的 delay 會進一步下降到 187349 個 cycle。



```
FIR_X = 1;
reg_mprj_datal = 0xA5100000;
reg_mprj_datal = FIR_Y << 16;

for(int i=2; i<75; i=i+1) {
    FIR_X = i;
    reg_mprj_datal = FIR_Y << 16;
}
for(int i=75; i>-75; i=i-1) {
    FIR_X = i;
    reg_mprj_datal = FIR_Y << 16;
}
for(int i=-75; i<75; i=i+1) {
    FIR_X = i;
    reg_mprj_datal = FIR_Y << 16;
}
for(int i=75; i>-75; i=i-1) {
    FIR_X = i;
    reg_mprj_datal = FIR_Y << 16;
}
for(int i=-75; i<=0; i=i+1) {
    FIR_X = i;
    reg_mprj_datal = FIR_Y << 16;
}
```



```
int x = 1;
bool plus = true;
for (int i=0; i<600; i=i+1) {

    FIR_X = x;
    reg_mprj_datal = FIR_Y << 16;

    if(x == 75){plus = false;}
    else if(x == -75){plus = true;}

    if(plus){x++;}
    else{x--;}
}
```

(3) Techniques that can improve the throughput

若使用 bram12，可以將多餘的那個空間充當 data buffer，在 fir engine 計算目前的值時先將下一筆 input 送入，減少 delay。但因為整體效能受到 fir engine 的部分影響較低，因此即使使用 bram12 能提升的幅度仍有限。

7. GitHub link

https://github.com/liyuan-chang/SoC-Lab-Caravel-FIR