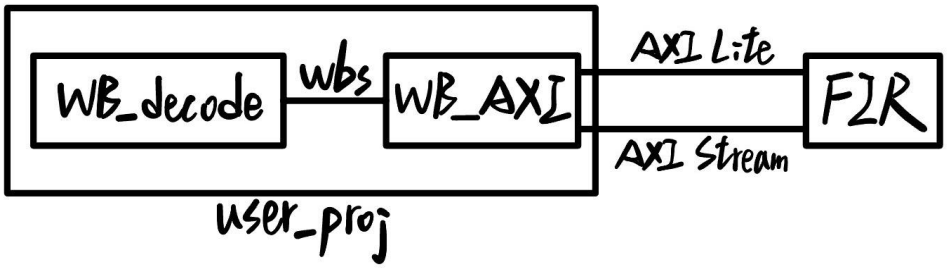
系統晶片設計 SOC Design
Lab4-2

111061534 陳翀
111061560 吳俊鋌

1. Block diagram



2. Interface

Firmware 的 c 檔案經過 RISCV 的編譯器後與 user_project 做連接，最後再用 testbench 做 rtl 的模擬並顯示波型及 latency 等相關資料。

Wishbone to AXI 我的設計是每次 wbs_cyc_i 是 1 的時候 wishbone 的計數器開始計數，到 10 之後才會將指令進行 decode 並依照不同位址傳給 AXI 或是 Lab4-1 的功能，而 wbs_ack_o 則是由 AXI Lite 或 Stream 的回傳指令來判斷是否為 1。

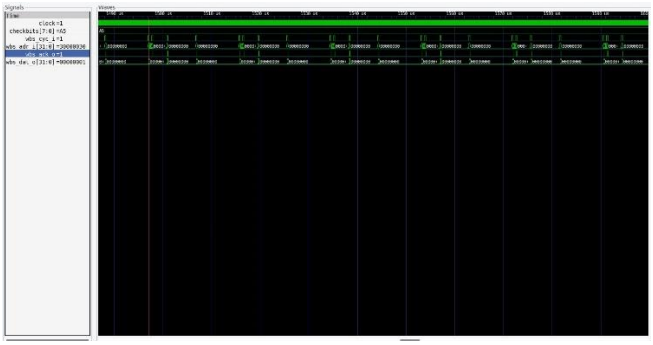
其中我的 Xn 與 Yn 不是用講義的位址而是自己訂的(0X30..030 & 0X30..034)。

3. Waveform

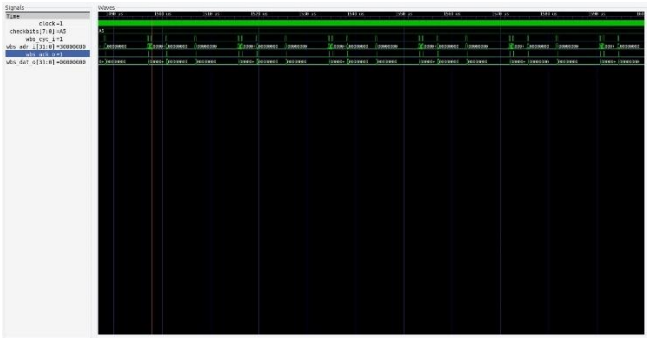
mprj[23:16] = 0xA5

mprj[23:16] = 0x5A

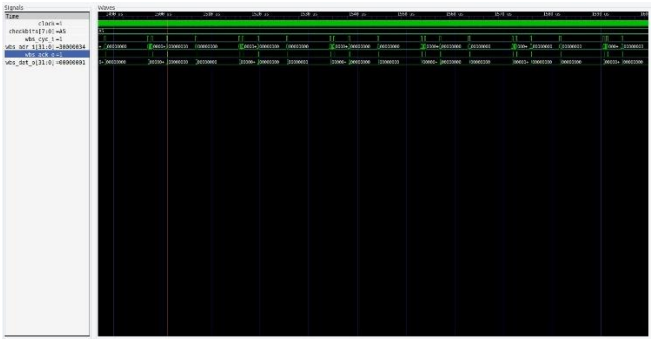
Detect if (Xn==1)



Transmit X data



Detect if (Yn==1)



Transmit Y data

4. FIR

理論的 throughput 是 : 1176470*32 bps, clock = 25ns, cycle=34

經過 wishbone 的 throughput 是 : 99502*32 bps, clock = 25ns, cycle = 402

Metrics = 51712 * 12ns * 822 = 510087168

5. Firmware latency of feed data

若是在 counter_la_fir.c 中將值輸入則每次操作間都有很大的 latency，但若是使用 fir.c(用.mprjram)的話則會有很大的改善，每兩個 wbs_cyc_i 之間有 18 個 clock。

6. Latency

執行 input data 長度為 64 時的結果，其中 input data 等於 1 到 64。

因 eeclass 上的 pdf 要求輸出最後一個 y 的後 8 bits，因此 y 的前 24(32-8) bits 沒有被我輸出至 terminal。

```
ubuntu@ubuntu2004:~/course-lab_4-2/testbench/counter_la_fir$ source run_clean
ubuntu@ubuntu2004:~/course-lab_4-2/testbench/counter_la_fir$ source run_sim
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.
=====SIMULATION START=====
Final Y :  45
Latency :              1292800 ns
                         51712 cycles
=====SIMULATION FINISH=====
```

7. Improved

1. 因改成可同時讀寫的 bram，因此當 bram 讀出最早進來的一筆資料後這筆資料就已經不會再被使用，若是原本的 bram 則需要額外的一個 clock 來寫入新的資料(最新的 X)，但 bram12 可以在讀取其他位址的資料時同時寫入，因此整體的 clock 數會被減少。