SoC Design

Lab 4-2 Caravel FIR

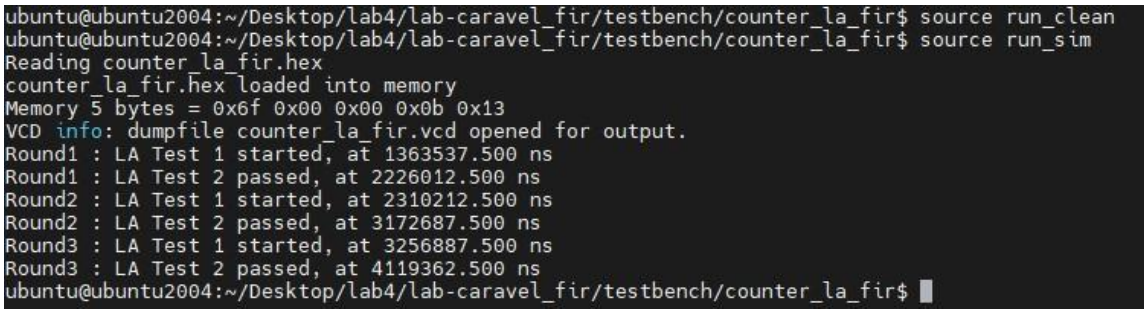
111064559 徐詠祺 112061527 紀承龍

■ Metrics:

➢ #34499(start mark to end mark) * 8.1 ns * #644 (LUT +FF)

= 179960583.6

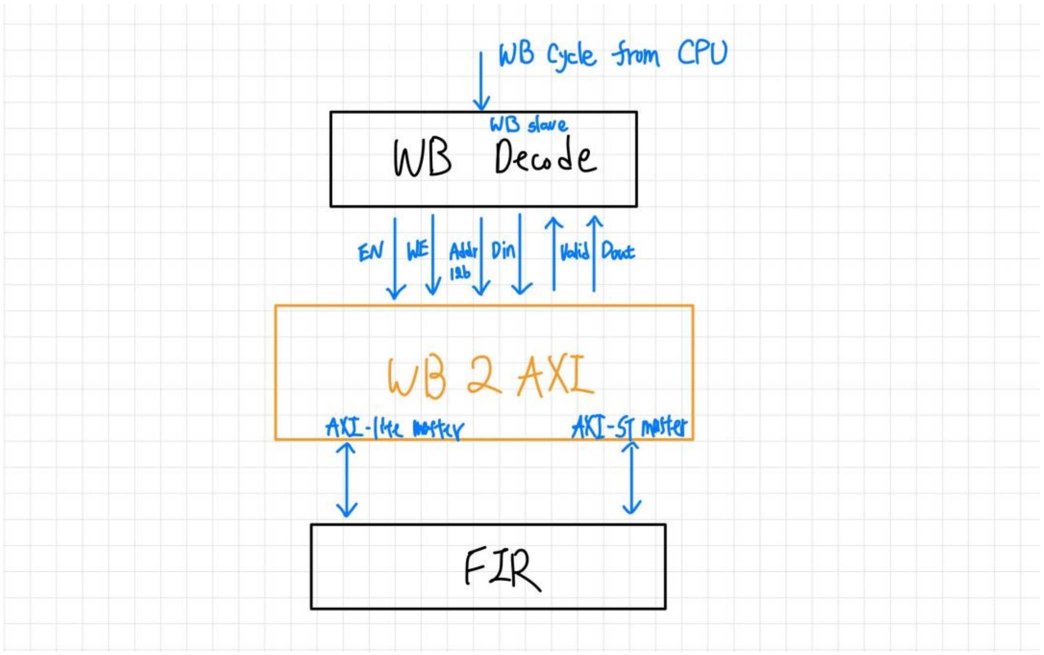
■ Run sim result:



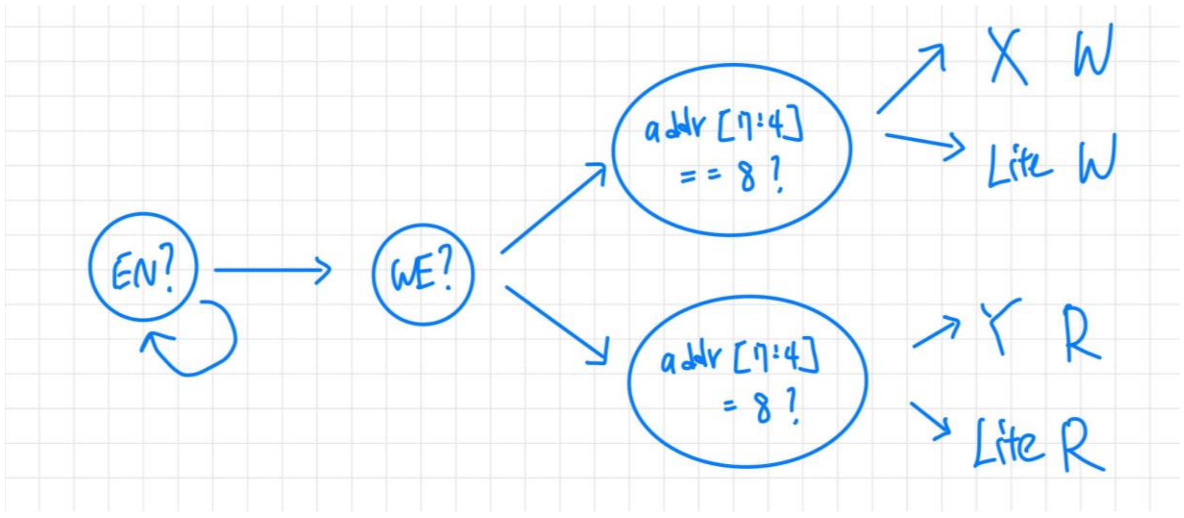
```
ubuntu@ubuntu2004:~/Desktop/lab4/lab-caravel_fir/testbench/counter_la_fir$ source run_clean
ubuntu@ubuntu2004:~/Desktop/lab4/lab-caravel_fir/testbench/counter_la_fir$ source run_sim
Reading counter_la_fir.hex
counter_la_fir.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_fir.vcd opened for output.
Round1 : LA Test 1 started, at 1363537.500 ns
Round1 : LA Test 2 passed, at 2226012.500 ns
Round2 : LA Test 1 started, at 2310212.500 ns
Round2 : LA Test 2 passed, at 3172687.500 ns
Round3 : LA Test 1 started, at 3256887.500 ns
Round3 : LA Test 2 passed, at 4119362.500 ns
ubuntu@ubuntu2004:~/Desktop/lab4/lab-caravel_fir/testbench/counter_la_fir$
```

■ Design block diagram:

➢ wb2axi module block diagram:



➢ wb2axi module WB cycle target control logic:



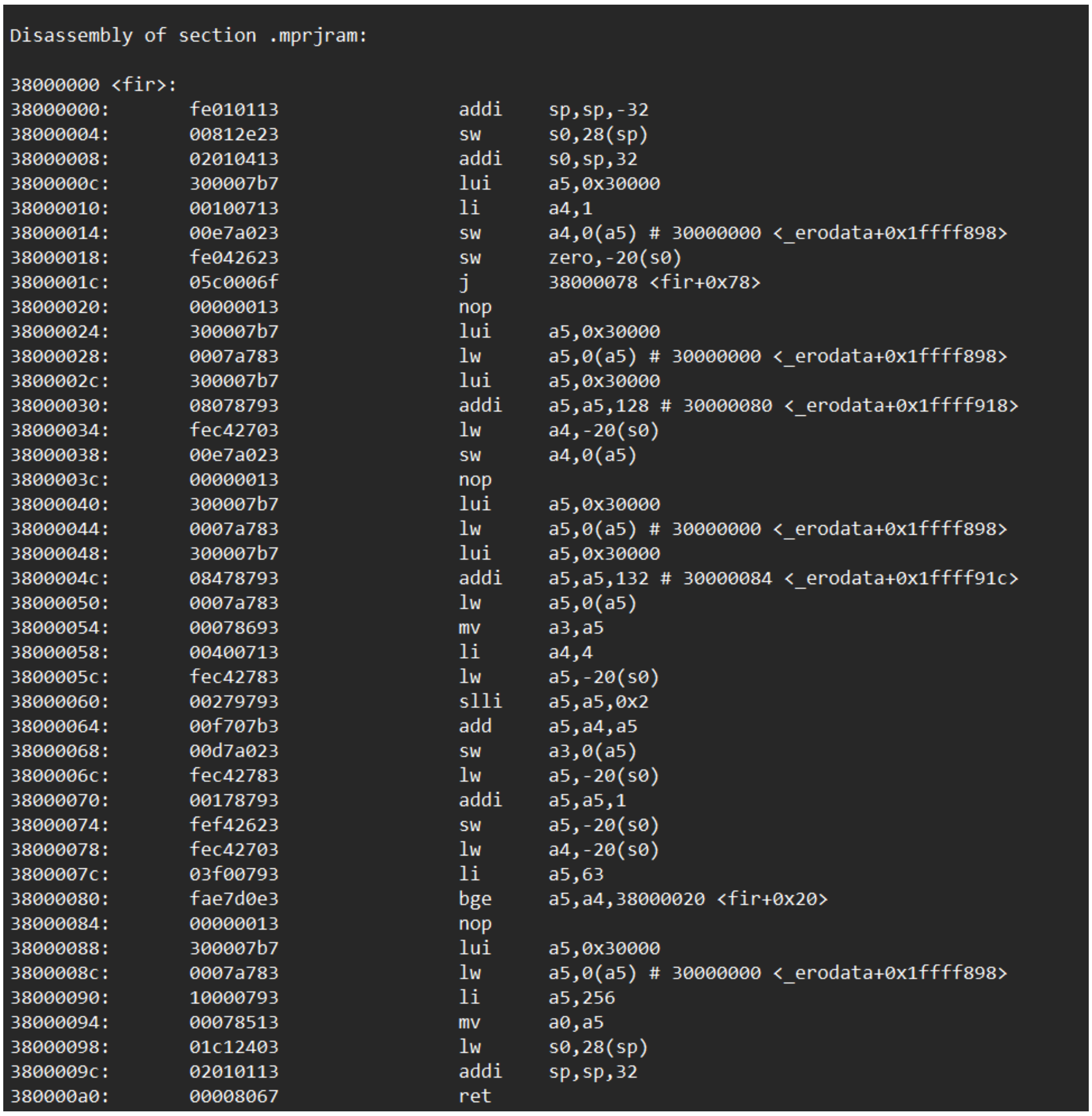
■ **The interface protocol between firmware, user project and testbench:**

我們在 firmware 中對 address map 內定義好的 user project address 進行讀寫操作，當 CPU 執行該 firmware 便會發起 wishbone cycle 來對該 user project address 同樣地進行讀寫，此時 user project example module 收到 wishbone cycle 便會進行 address decode 去判斷該次讀寫目標是對 user bram 還是 fir，當讀寫目標是 fir 時 wb2axi module 便會收到 user project example module 發起的 enable, write enable 訊號 (from wishbone)，並且 wb2axi module 會扮演 axi-lite, axi-streaming master 的角色，根據 wishbone address 來發起讀寫，讀寫的 axi handshake 完成後 wb2axi module 便會對 user project example module 回 valid 訊號，也就代表這次 wishbone cycle 對 fir 的傳輸已經完成，user project example module 便會拉高 wishbone ack 以通知

CPU 這次 wishbone cycle 完成。

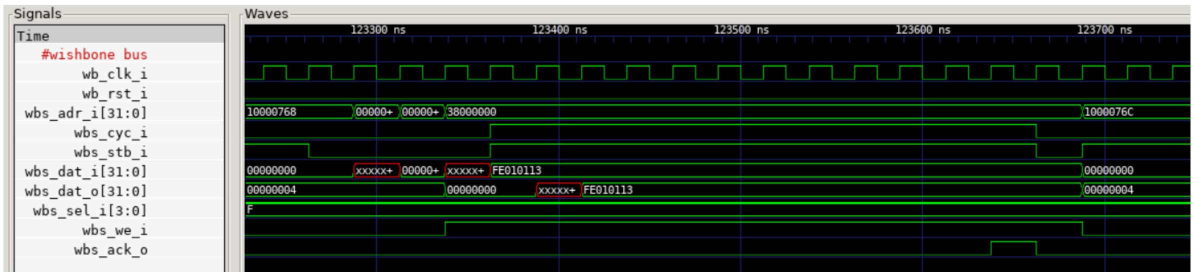
■ Waveform and analysis of the hardware/software behavior:

Firmware size : 38000000~380000a0。

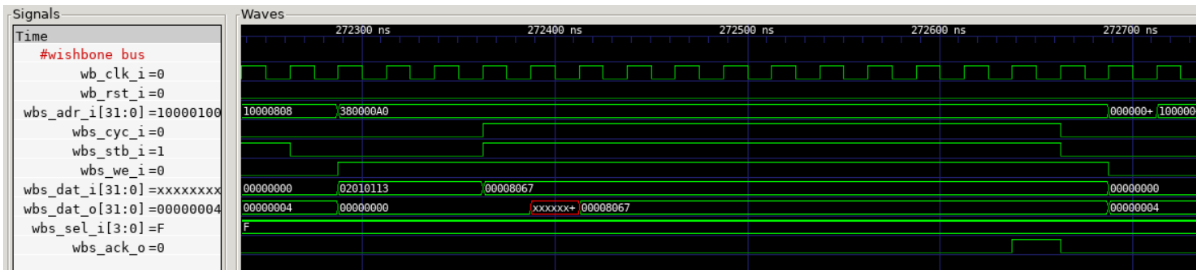


```
Disassembly of section .mprjram:

38000000 <fir>:
38000000:       fe010113                addi    sp,sp,-32
38000004:       00812e23                sw      s0,28(sp)
38000008:       02010413                addi    s0,sp,32
3800000c:       300007b7                lui     a5,0x30000
38000010:       00100713                li      a4,1
38000014:       00e7a023                sw      a4,0(a5) # 30000000 <_erodata+0x1ffff898>
38000018:       fe042623                sw      zero,-20(s0)
3800001c:       05c0006f                j       38000078 <fir+0x78>
38000020:       00000013                nop
38000024:       300007b7                lui     a5,0x30000
38000028:       0007a783                lw      a5,0(a5) # 30000000 <_erodata+0x1ffff898>
3800002c:       300007b7                lui     a5,0x30000
38000030:       08078793                addi    a5,a5,128 # 30000080 <_erodata+0x1ffff918>
38000034:       fec42703                lw      a4,-20(s0)
38000038:       00e7a023                sw      a4,0(a5)
3800003c:       00000013                nop
38000040:       300007b7                lui     a5,0x30000
38000044:       0007a783                lw      a5,0(a5) # 30000000 <_erodata+0x1ffff898>
38000048:       300007b7                lui     a5,0x30000
3800004c:       08478793                addi    a5,a5,132 # 30000084 <_erodata+0x1ffff91c>
38000050:       0007a783                lw      a5,0(a5)
38000054:       00078693                mv      a3,a5
38000058:       00400713                li      a4,4
3800005c:       fec42783                lw      a5,-20(s0)
38000060:       00279793                slli    a5,a5,0x2
38000064:       00f707b3                add     a5,a4,a5
38000068:       00d7a023                sw      a3,0(a5)
3800006c:       fec42783                lw      a5,-20(s0)
38000070:       00178793                addi    a5,a5,1
38000074:       fef42623                sw      a5,-20(s0)
38000078:       fec42703                lw      a4,-20(s0)
3800007c:       03f00793                li      a5,63
38000080:       fae7d0e3                bge     a5,a4,38000020 <fir+0x20>
38000084:       00000013                nop
38000088:       300007b7                lui     a5,0x30000
3800008c:       0007a783                lw      a5,0(a5) # 30000000 <_erodata+0x1ffff898>
38000090:       10000793                li      a5,256
38000094:       00078513                mv      a0,a5
38000098:       01c12403                lw      s0,28(sp)
3800009c:       02010113                addi    sp,sp,32
380000a0:       00008067                ret
```

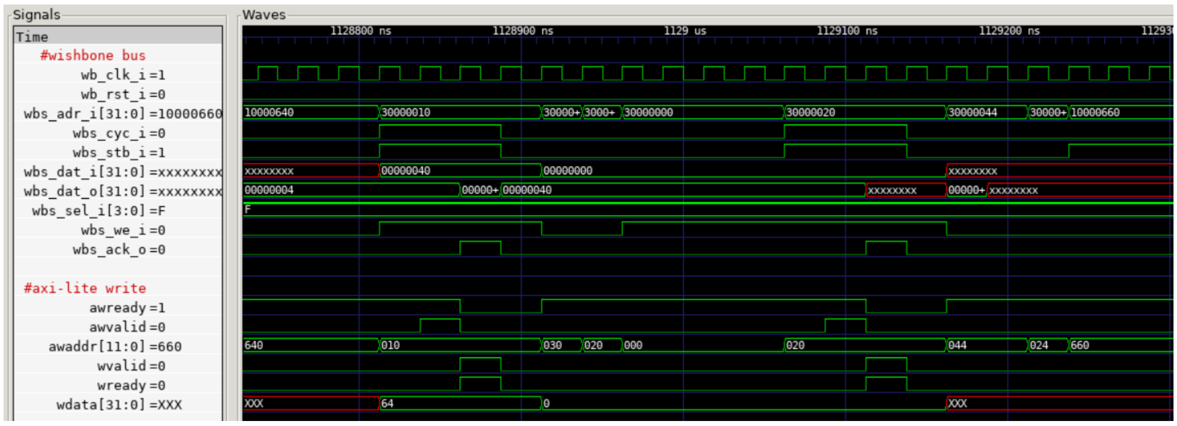
寫入 firmware 到 user project bram：



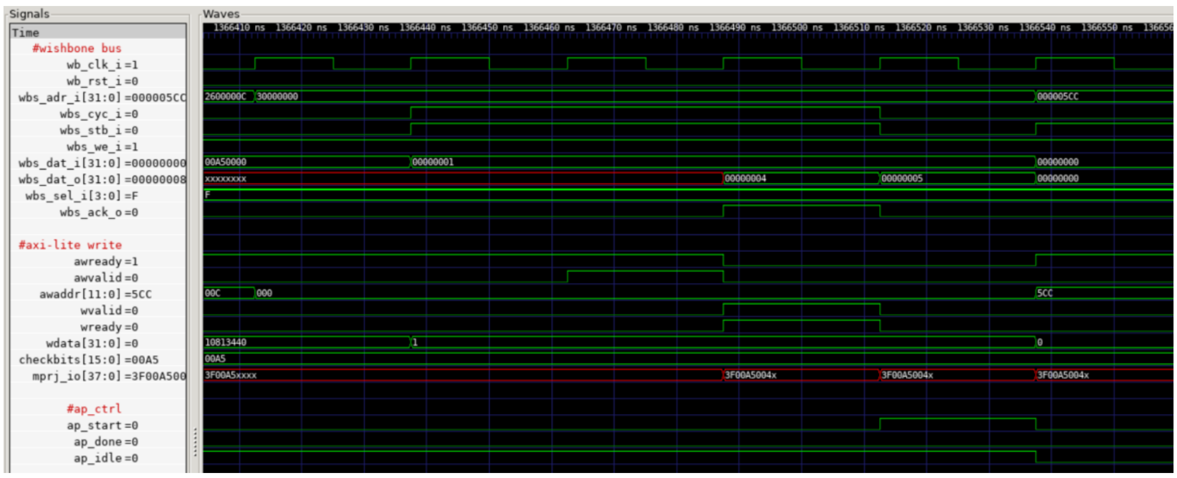
寫到 380000a0：



Wishbone to axi-lite write coefficient：



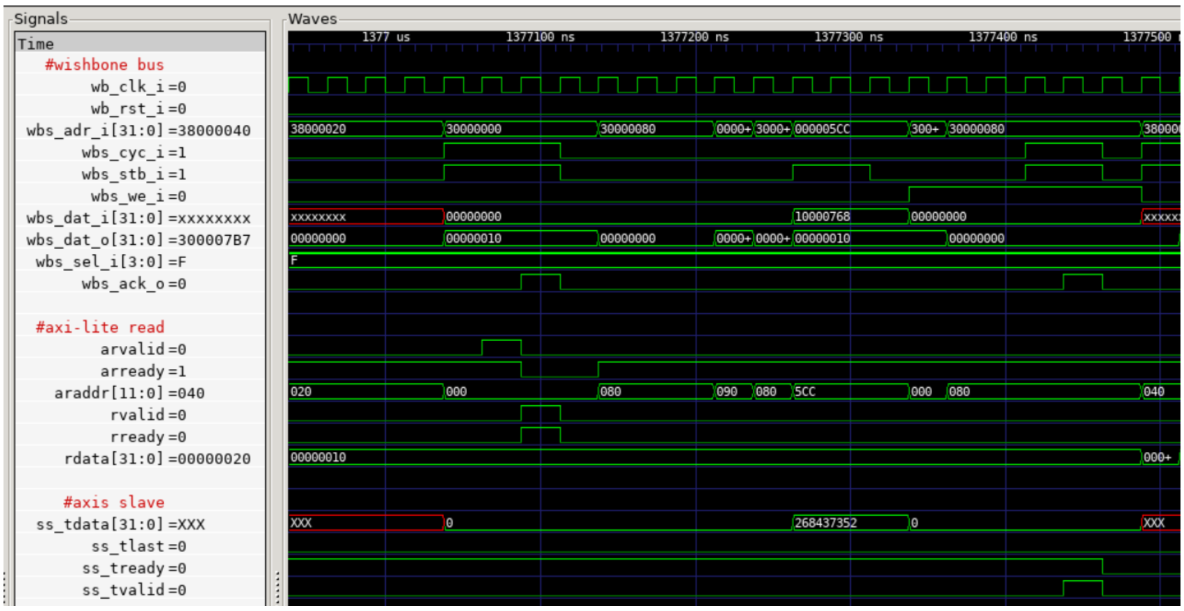
Wishbone to axi-lite write ap_start：



Wishbone to axi stream：

配合 axi-lite read 讀 X[n]_ready(ap_ctrl[4])，若讀到 ap_ctrl[4]，幾個 cycle

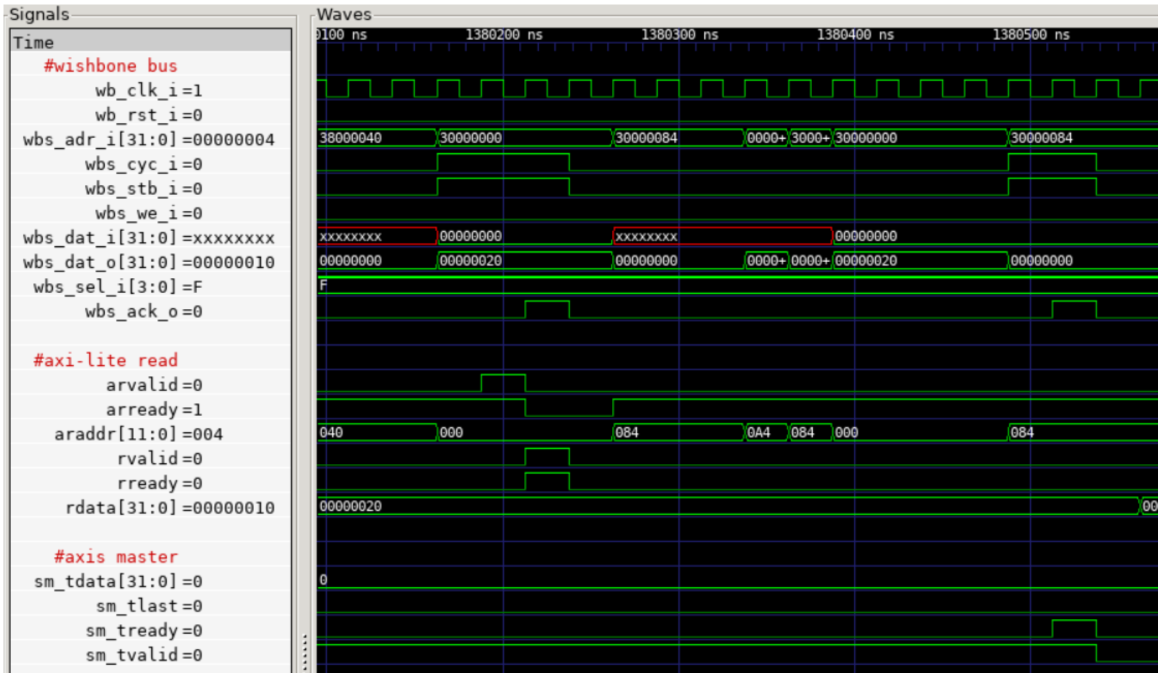
後會配合 axi stream 將 data 送入 fir engine 做運算。



```
Signals
Time
  #wishbone bus
        wb_clk_i =0
        wb_rst_i =0
 wbs_adr_i[31:0] =38000040
       wbs_cyc_i =1
       wbs_stb_i =1
        wbs_we_i =0
 wbs_dat_i[31:0] =xxxxxxxx
 wbs_dat_o[31:0] =300007B7
   wbs_sel_i[3:0] =F
       wbs_ack_o =0

  #axi-lite read
         arvalid =0
         arready =1
    araddr[11:0] =040
          rvalid =0
          rready =0
     rdata[31:0] =00000020

  #axis slave
  ss_tdata[31:0] =XXX
        ss_tlast =0
       ss_tready =0
       ss_tvalid =0
```

配合 axi-lite read 讀 Y[n]_ready(ap_ctrl[5])，若讀到 ap_ctrl[5]，幾個 cycle 後

會配合 axi stream 將結果送出。

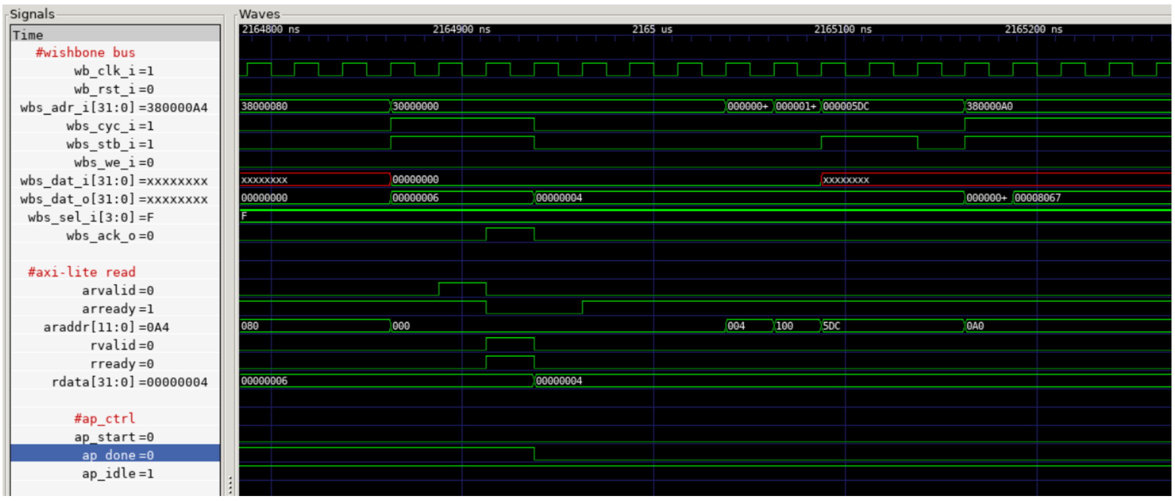


```
Signals
Time
  #wishbone bus
        wb_clk_i =1
        wb_rst_i =0
 wbs_adr_i[31:0] =00000004
       wbs_cyc_i =0
       wbs_stb_i =0
        wbs_we_i =0
 wbs_dat_i[31:0] =xxxxxxxx
 wbs_dat_o[31:0] =00000010
   wbs_sel_i[3:0] =F
       wbs_ack_o =0

  #axi-lite read
         arvalid =0
         arready =1
    araddr[11:0] =004
          rvalid =0
          rready =0
     rdata[31:0] =00000010

  #axis master
  sm_tdata[31:0] =0
        sm_tlast =0
       sm_tready =0
       sm_tvalid =0
```

Ap_done:

送完最後一筆 fir 運算後，拉起 ap_done 等待 firmware 的一些操作後，直到

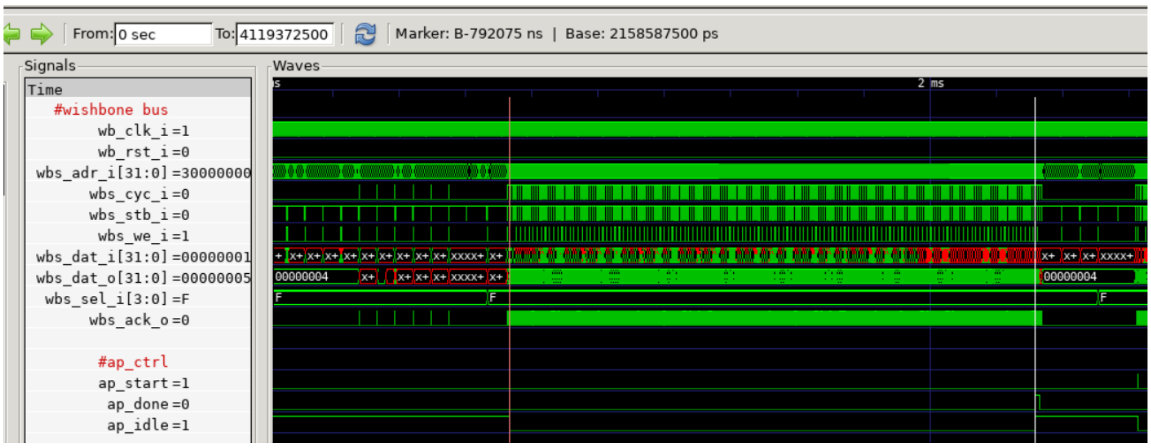
ap_done 被 wishbone to axi-lite read 讀到後 reset。



```
Signals
Time
#wishbone bus
wb_clk_i=1
wb_rst_i=0
wbs_adr_i[31:0]=380000A4
wbs_cyc_i=1
wbs_stb_i=1
wbs_we_i=0
wbs_dat_i[31:0]=xxxxxxxx
wbs_dat_o[31:0]=xxxxxxxx
wbs_sel_i[3:0]=F
wbs_ack_o=0

#axi-lite read
arvalid=0
arready=1
araddr[11:0]=0A4
rvalid=0
rready=0
rdata[31:0]=00000004

#ap_ctrl
ap_start=0
ap_done=0
ap_idle=1
```

- **FIR engine theoretical throughput & actually measured throughput:**
  - <span>Base on AP_start to AP_done read & data length = 64</span>
  - Theoretical: #778 cycles to finish #64 inputs
  - Actually measured: #31683 cycles to finish #64 inputs
  - 計算 cycle 方法:

cycle = (ap_done_posedge time - ap_start_posedge time) / clock period

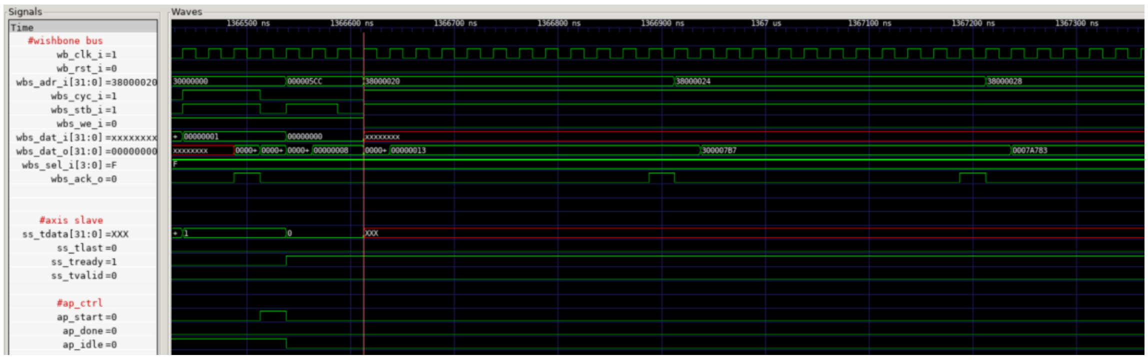


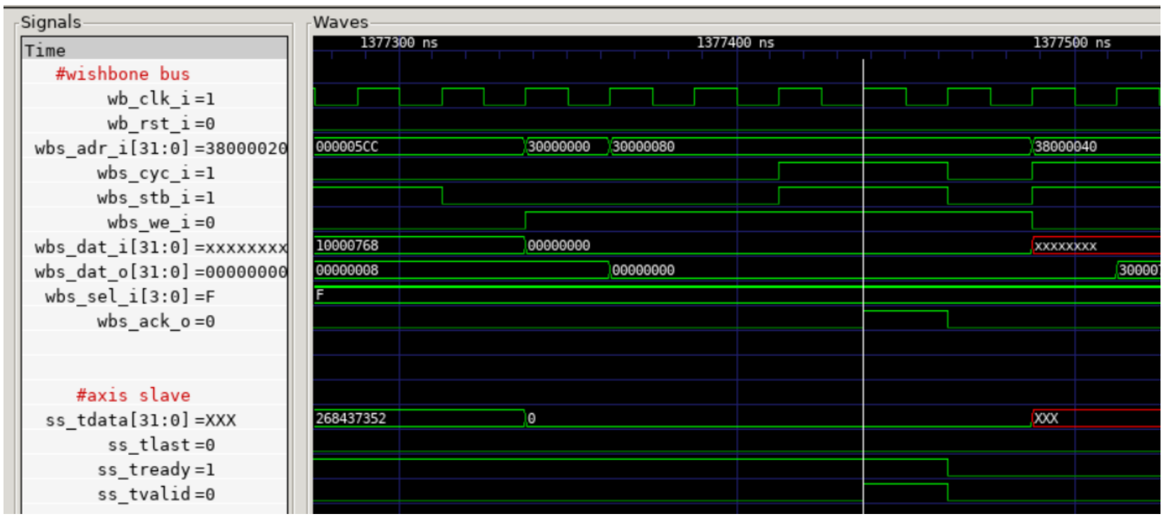
```
From: 0 sec   To: 4119372500   Marker: B-792075 ns | Base: 2158587500 ps
Signals
Time
#wishbone bus
wb_clk_i=1
wb_rst_i=0
wbs_adr_i[31:0]=30000000
wbs_cyc_i=0
wbs_stb_i=0
wbs_we_i=1
wbs_dat_i[31:0]=00000001
wbs_dat_o[31:0]=00000005
wbs_sel_i[3:0]=F
wbs_ack_o=0

#ap_ctrl
ap_start=1
ap_done=0
ap_idle=1
```

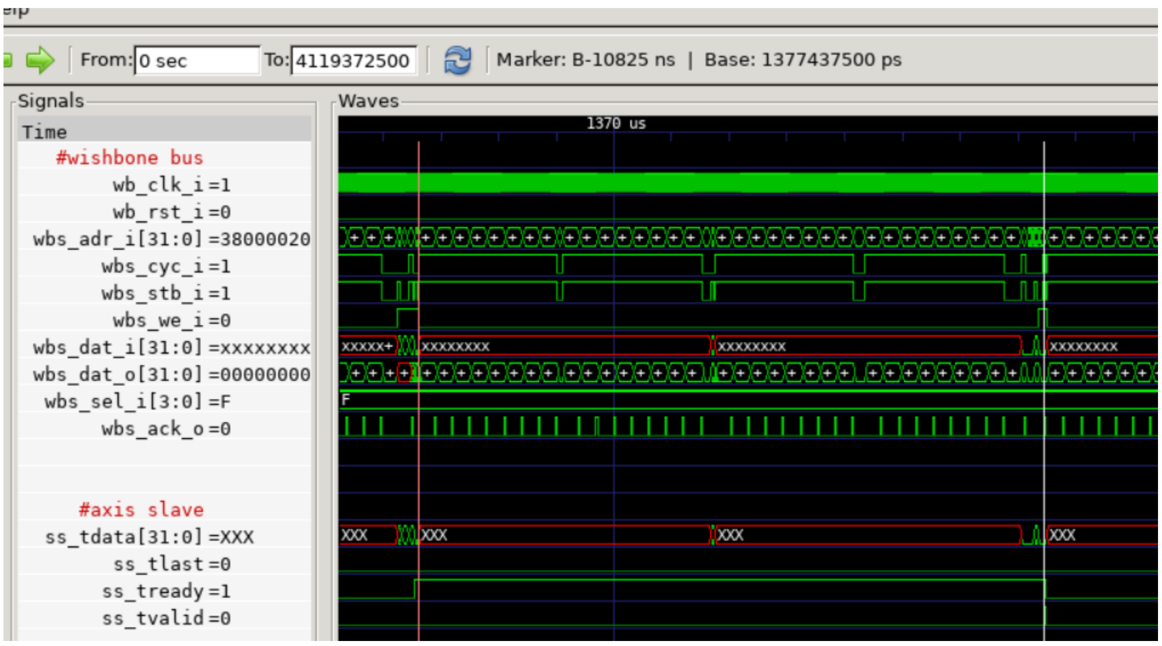
■ Latency for firmware to feed data:

Ap_start 被 set 後，從 38000020 開始讀 firmware，到 data 被餵進去總共：

10825(ns)/25(ns) = 433 cycles。







■ Techniques used to improve the throughput:

在 FIR design 中我們設計了盡可能降低 cycle 數的時序安排來盡可能縮短 X input 後到 Y output valid 所需 cycle 數，此外在 wb2axi 過程也只多花一個 cycle 來完成將 WB write cycle 轉換成 AXI streaming 輸入 FIR，Y output 的部分也只多花一個 cycle 便完成 WB read cycle 把 Y 送回 CPU 中，總而言之盡量使整個流程精簡，只保留必要的 cycle 去做到資料讀寫與 FIR 的計算。

➢ Does bram12 give better performace?

在我們的 fir design 中，使用 bram11 就可以滿足最好的 performace 了。

➢ Other method to improve the performance

1. 只使用 Wishbone bus 傳遞資料，若配合 AXI protocol 傳輸會多花幾個 cycle。
2. 使用 11 個乘法器並配合 pipeline 會有更好的 throughput。