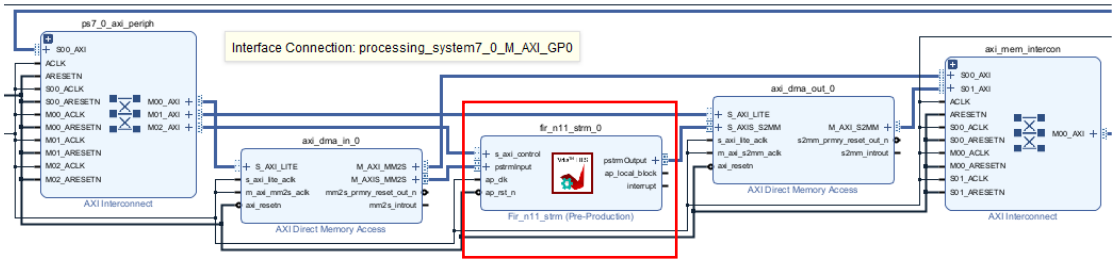
312510147 陳建嘉

SoC Design Lab3

Brief introduction about the overall system:

這一次要設計的主要是 lab2 中的 FIR Module



設計的部分包含 lab2 中有使用到的 apctrl signal，兩顆用來存儲 data 和 tap 的 sram(Bram11.v)，系統中由 AXI_LITE 來進行 configuration 的 transmit，透過 AXIStream 作為 data transfer 的 protocal，整體是一次相當能夠令學生熟悉 Verilog coding 的 lab.

Recap for I/O protocal:

AXI

Advanced eXtensible Interface，是由 ARM 所定義的接口協議

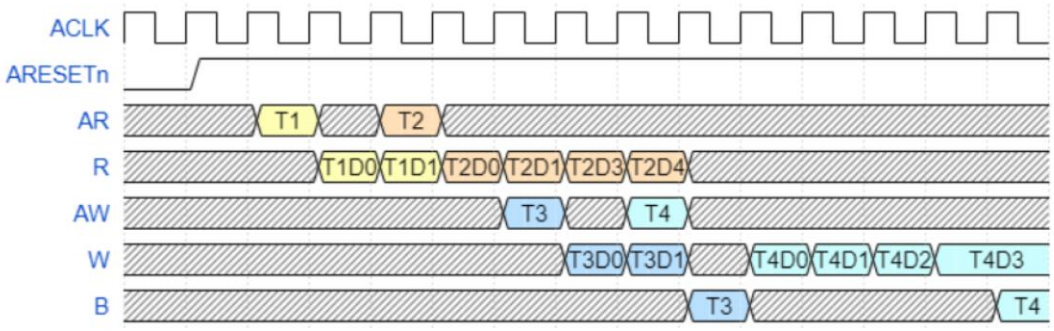
基本上 AXI4 分為三種:

AXI4-FULL ： 用於滿足高性能存儲 interface 需求

AXI4-Lite ： 用於簡單的低吞吐量通信

AXI4-Stream ： 用於高速傳輸，一般會和 DMA 一併使用

AXI Master Transaction Waveform:



Example of multiple transactions: T1 (arlen=1), T2 (arlen=3) read transactions and T3 (awlen=1), T4 (awlen=3) write transactions.

ACLK:時鐘訊號，posedge trigger

Reset: asynchronize reset

在 reset 訊號期間要滿足以下 requirement

1.A master interface must drive ARVALID, AWVALID, and WVALID LOW.

2.A slave interface must drive RVALID and BVALID LOW.

3.All other signals can be driven to any value.

AR : address read channel

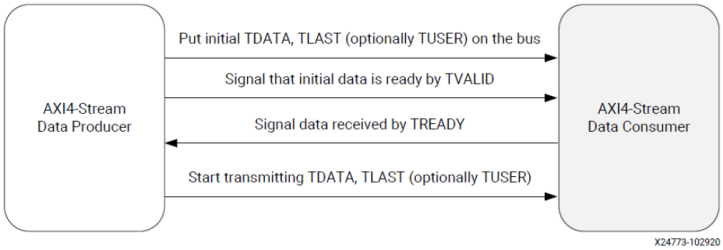
AW : address write channel

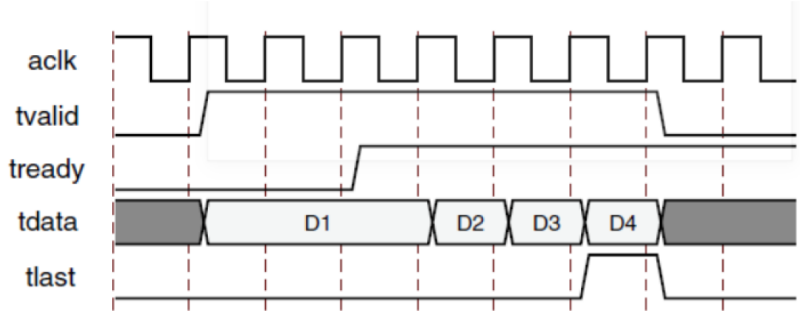
R : data read channel

W : data write channel

B : write response channel (在寫完數據後，Master 需要確認 Slave 有没有收完數據，Slave 收到完整數據後，会通过 Write Response Channel 给 Master 一個 completion signal，表示完成)

Stream





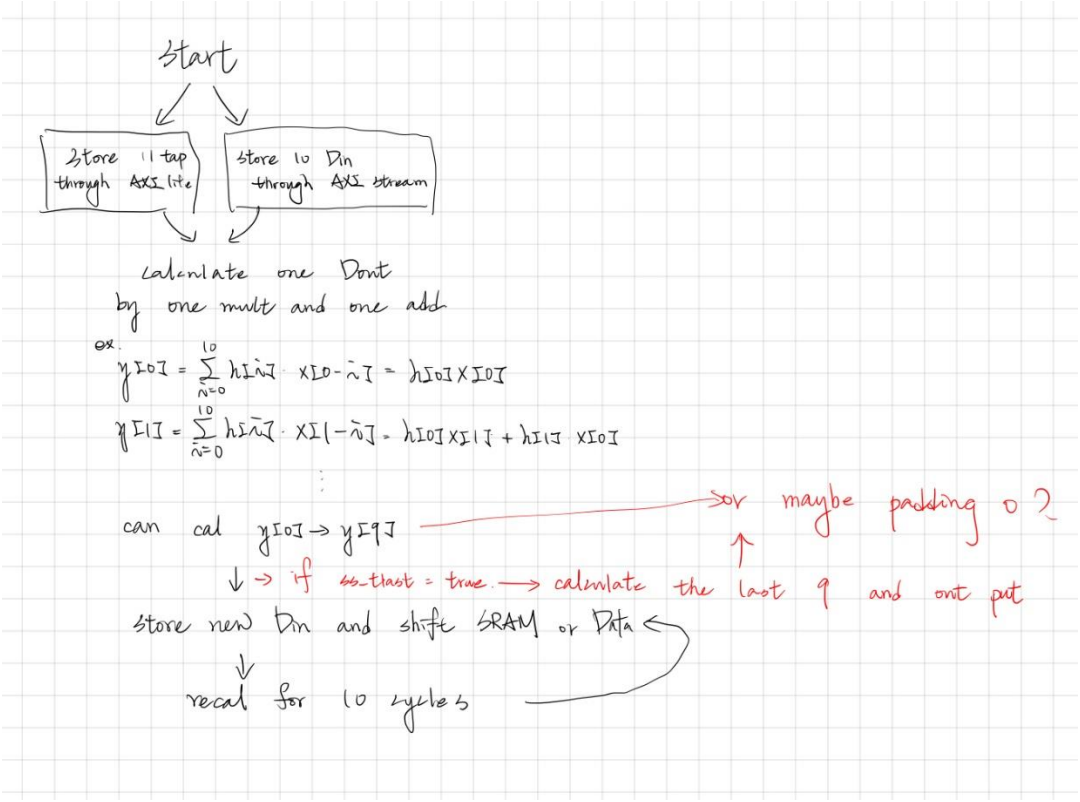
TVALID : master 表示目前訊號有效

TREADY : slave 表示準備好接收訊號

TDATA : master 數據傳輸

TLAST : master 表示當前數據是最後一筆

Flow chart:

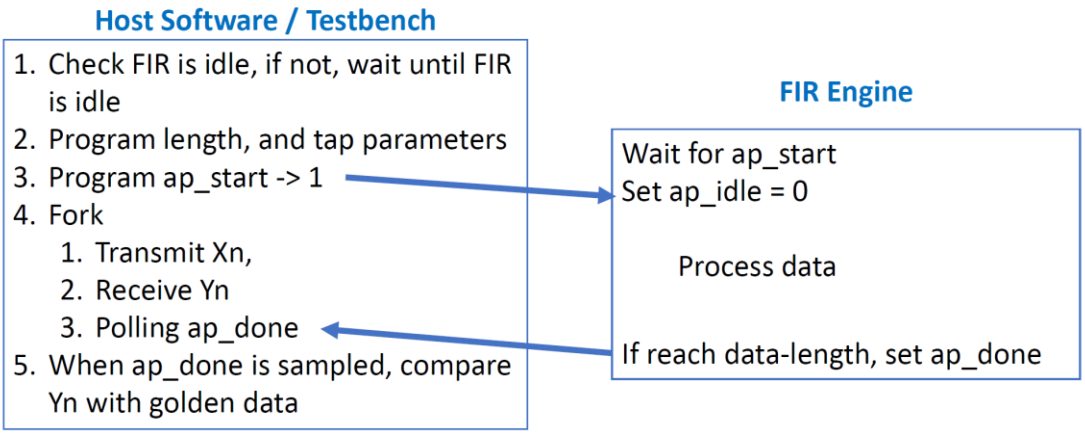


整體流程大致如上圖，基本上我會在一開始對兩邊 input 先進行處理，從 axilite 那邊與 tb 去進行 tap 的傳輸與驗證值(存入 bram)，另一方面先取第一筆的 data 並將其存入 bram 中，在寫 data 的同時，我會先將 data_ram 進行 reset，意思就是我會先將 data*1 + 0*(tap_length-1)，共 tap_length 筆數據先對 bram 進行第一次的寫值，待兩邊都準備好後就會進入 FIR_out computation，在一次只能使用一組乘加器的規範下，我們在每一個 cycle 都會將對應的值取出進行運算，直到計算完所有的部分，並拉起 sm 進行 output，至此就是完成一筆 data 的計算，整體的 flow 基本上就是對上述的計算流程執行 data_length 次。

值得注意的是，在處理完一筆 data output 後，必須要對於 data_ram 做 shift，對此我的設計方式其實有一點蠢，我將 data 反覆的取出寫入取出寫入，重複數次後變完成了資料的搬運，就可以準備寫入下一筆資料，不過更好的做法應該是使用 pointer 去指向實際上該處理的位置，只是我當初覺得這樣想起來不太像 shift 就沒有處理了，之後如果有機會的話再對此修正。

除了上面的問題，我記得老師上課有說過使用 FSM 設計相對來說是在執行上比較沒有效率的方法，但我對於此還不太熟悉，因此我還是使用了兩個部份的 fsm，去分別對於 AXILite 和 AXIStream 做處理，這部分我想我還有很大的進步空間。

AP signal



可以看到上圖為 host 和 engine 之間的關係，ap_idle 會在我接收到 ap_start 後去 pull high，至於後面對於 ap_done 的處理方式我想有兩個，一是在 FIR engine 內去使用 cnt 來數總共的輸出個數，因為在一開始 write config 時 tb 會將總共的 data length 送進 engine，不過使用這樣的設計方式會有一個問題，那便是我必須開一個 10bit 的 cnt 來專門對其處理，在資源的共用上我想不太理想，而另一種方法是，因為我對於 Xn Yn 的處理是一進一出，換言之我只要確保我在處理的 Xn 是最後一筆，就可以判斷 Yn 最後一筆輸出，ap_done 也會在最後一筆 trasmit 出去後拉高，其他組 pattern 就是周而復始。

Some screendump

UTIL

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	343	0	0	53200	0.64
LUT as Logic	343	0	0	53200	0.64
LUT as Memory	0	0	0	17400	0.00
Slice Registers	163	0	0	106400	0.15
Register as Flip Flop	163	0	0	106400	0.15
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

312510147 陳建嘉

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
57	Yes	-	Reset
2	Yes	Set	-
104	Yes	Reset	-

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	140	0.00
RAMB36/FIFO*	0	0	0	140	0.00
RAMB18	0	0	0	280	0.00

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	3	0	0	220	1.36
DSP48E1 only	3				

4. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	329	0	0	125	263.20
Bonded IPADs	0	0	0	2	0.00
Bonded IOPADs	0	0	0	130	0.00
PHY_CONTROL	0	0	0	4	0.00
PHASER_REF	0	0	0	4	0.00
OUT_FIFO	0	0	0	16	0.00
IN_FIFO	0	0	0	16	0.00
IDELAYCTRL	0	0	0	4	0.00
IBUFDS	0	0	0	121	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	16	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	16	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	200	0.00
ILOGIC	0	0	0	125	0.00
OLOGIC	0	0	0	125	0.00

基本上可以看到 I/O 大炸裂，不過並不要緊，這主要是因為在本次設計中有許多的 I/O pin 其實都只是要與其他 block 溝通用途，之後將整個 system 弄好時就不會需要使用到這麼多 I/O，也因此這次實驗也只進行到 Syn，而沒有往下繼續做。

Timing report

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.150 ns	Worst Hold Slack (WHS):	0.140 ns	Worst Pulse Width Slack (WPWS):	1.700 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:	242	Total Number of Endpoints:	242	Total Number of Endpoints:	165

All user specified timing constraints are met.

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.150ns  (required time - arrival time)
  Source:                 araddr_buf_reg[10]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@2.200ns period=4.400ns})
  Destination:            tap_cnt_reg[0]/R
                            (rising edge-triggered cell FDRE clocked by axis_clk  {rise@0.000ns fall@2.200ns period=4.400ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            4.400ns  (axis_clk rise@4.400ns - axis_clk rise@0.000ns)
  Data Path Delay:        3.513ns  (logic 1.021ns (29.063%)  route 2.492ns (70.937%))
  Logic Levels:           3  (LUT4=2 LUT6=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 6.528 - 4.400 )
    Source Clock Delay      (SCD):    2.456ns
    Clock Pessimism Removal (CPR):    0.184ns
  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
```

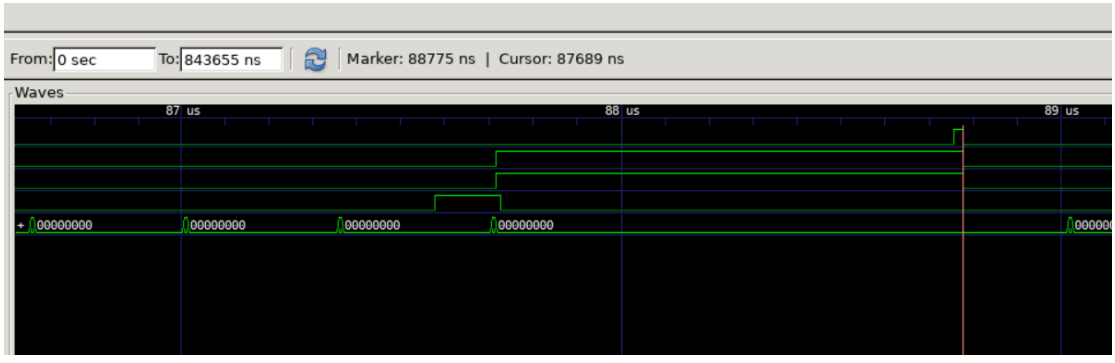
```
  Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
                       (clock axis_clk rise edge)
                                                  0.000     0.000 r
                                                  0.000     0.000 r  axis_clk (IN)
                       net (fo=0)                 0.000     0.000    axis_clk
                                                                 r  axis_clk_IBUF_inst/I
                       IBUF (Prop_ibuf_I_O)       0.972     0.972 r  axis_clk_IBUF_inst/O
                       net (fo=1, unplaced)       0.800     1.771    axis_clk_IBUF
                                                                 r  axis_clk_IBUF_BUFG_inst/I
                       BUFG (Prop_bufg_I_O)       0.101     1.872 r  axis_clk_IBUF_BUFG_inst/O
                       net (fo=164, unplaced)     0.584     2.456    axis_clk_IBUF_BUFG
                       FDCE                                        r  araddr_buf_reg[10]/C
-------------------------------------------------------------------    -------------------
                       FDCE (Prop_fdce_C_Q)       0.478     2.934 f  araddr_buf_reg[10]/Q
                       net (fo=3, unplaced)       0.759     3.693    araddr_buf[10]
                                                                 f  tap_A_OBUF[11]_inst_i_15/I1
                       LUT4 (Prop_lut4_I1_O)      0.295     3.988 f  tap_A_OBUF[11]_inst_i_15/O
                       net (fo=2, unplaced)       0.460     4.448    tap_A_OBUF[11]_inst_i_15_n_0
                                                                 f  tap_A_OBUF[6]_inst_i_2/I0
                       LUT4 (Prop_lut4_I0_O)      0.124     4.572 f  tap_A_OBUF[6]_inst_i_2/O
                       net (fo=4, unplaced)       0.473     5.045    tap_A_OBUF[6]_inst_i_2_n_0
                                                                 f  tap_cnt[3]_i_1/I5
                       LUT6 (Prop_lut6_I5_O)      0.124     5.169 r  tap_cnt[3]_i_1/O
                       net (fo=4, unplaced)       0.800     5.969    tap_cnt[3]_i_1_n_0
                       FDRE                                        r  tap_cnt_reg[0]/R
-------------------------------------------------------------------    -------------------

                       (clock axis_clk rise edge)
                                                  4.400     4.400 r
                                                  0.000     4.400 r  axis_clk (IN)
                       net (fo=0)                 0.000     4.400    axis_clk
                                                                 r  axis_clk_IBUF_inst/I
                       IBUF (Prop_ibuf_I_O)       0.838     5.238 r  axis_clk_IBUF_inst/O
                       net (fo=1, unplaced)       0.760     5.998    axis_clk_IBUF
                                                                 r  axis_clk_IBUF_BUFG_inst/I
                       BUFG (Prop_bufg_I_O)       0.091     6.089 r  axis_clk_IBUF_BUFG_inst/O
                       net (fo=164, unplaced)     0.439     6.528    axis_clk_IBUF_BUFG
                       FDRE                                        r  tap_cnt_reg[0]/C
                       clock pessimism            0.184     6.711
                       clock uncertainty         -0.035     6.676
                       FDRE (Setup_fdre_C_R)     -0.557     6.119    tap_cnt_reg[0]
-------------------------------------------------------------------
                       required time                         6.119
                       arrival time                         -5.969
-------------------------------------------------------------------
                       slack                                 0.150

Slack (MET) :             0.150ns  (required time - arrival time)
  Source:                 araddr_buf_reg[10]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@2.200ns period=4.400ns})
  Destination:            tap_cnt_reg[1]/R
                            (rising edge-triggered cell FDRE clocked by axis_clk  {rise@0.000ns fall@2.200ns period=4.400ns})
  Path Group:             axis_clk
```

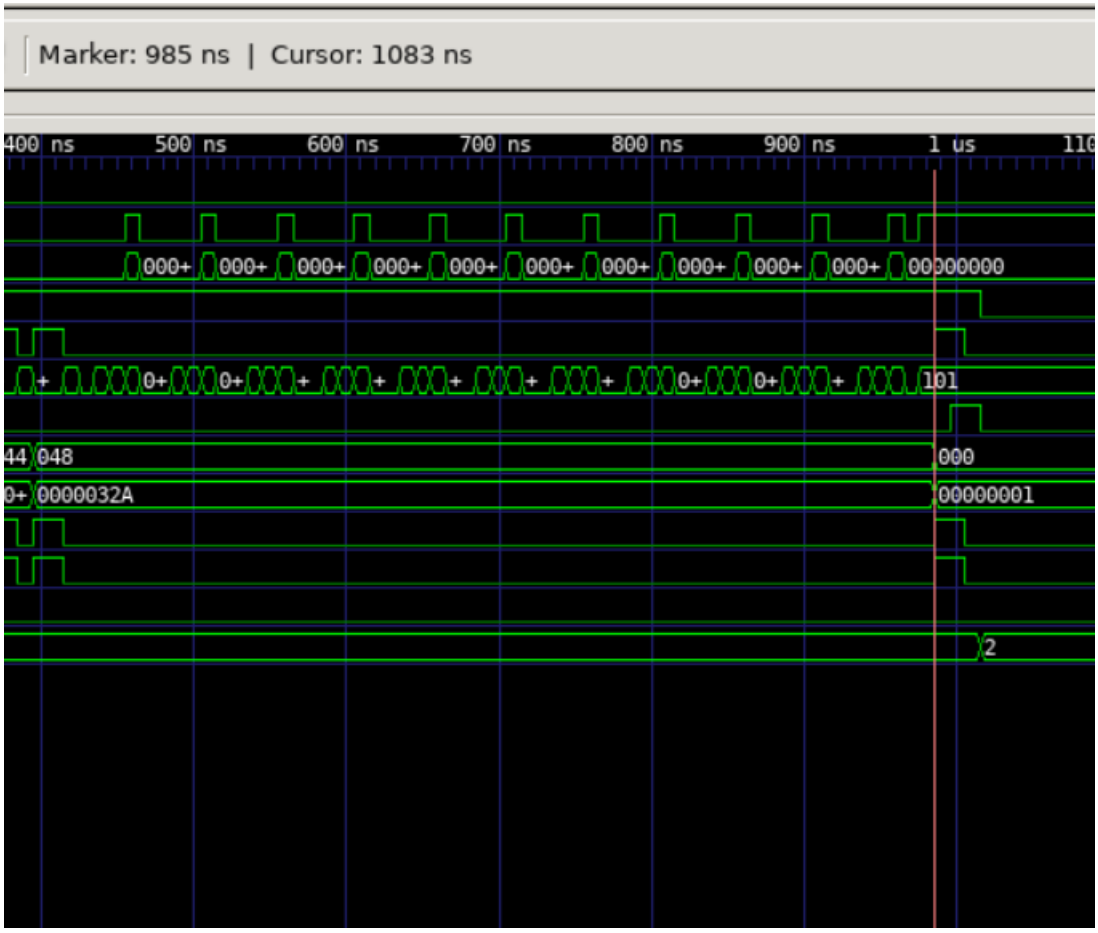
本次 lab 我將 clock period 設為 4.4，可以看到 report 中有列出我們的 critical path，並且有 meet timing constraint。

Ap done



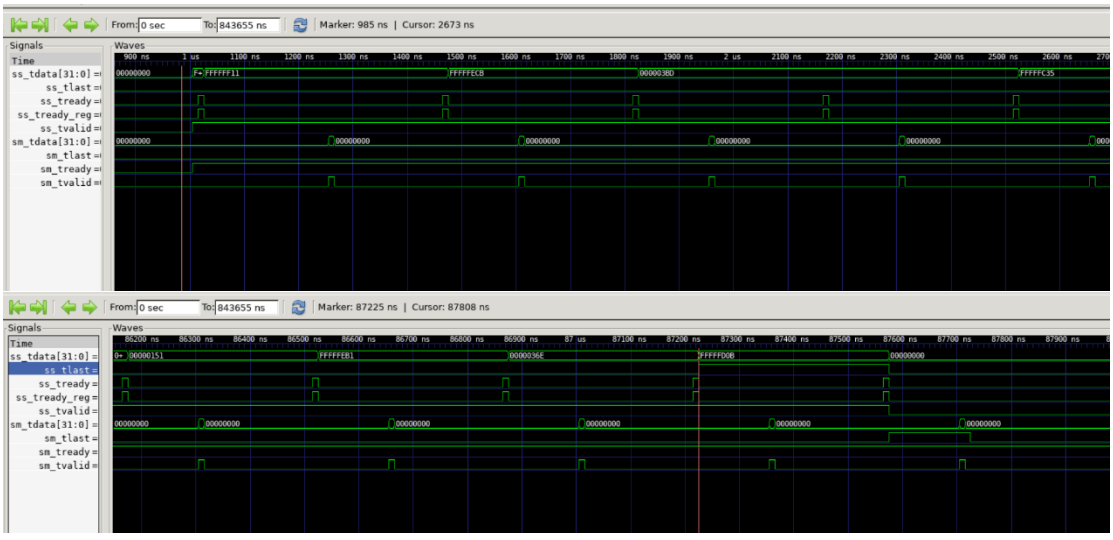
Ap start

配合上面 ap done 的時間可以計算出我總共花了多少 cycle 在 computing，需要注意的是這筆 pattern 並不是 600 筆 data length，所以計算出來的數字也許會比想像中小，但因為 shiftram 花費大量 cycle 所以整體下來應該還是十分巨大。



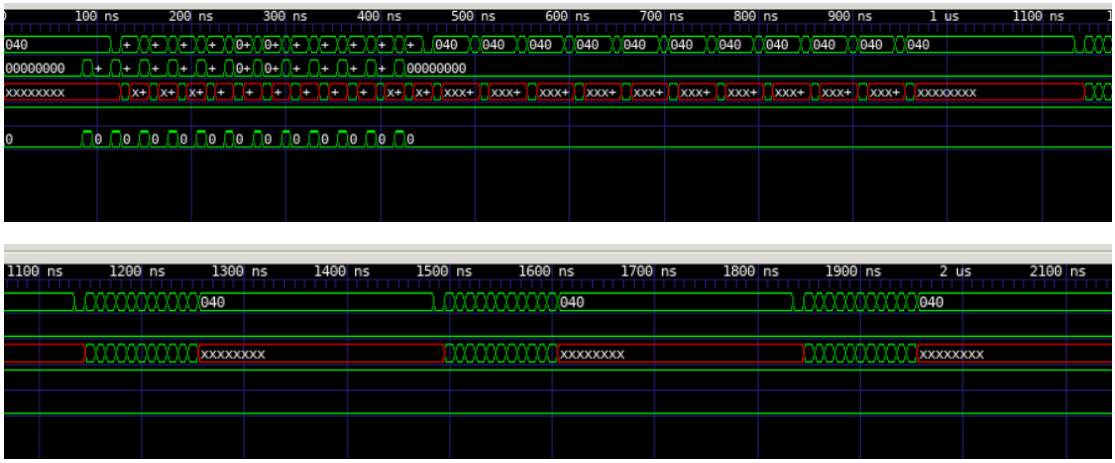
ss & sm transmit

基本上就是一筆進一筆出



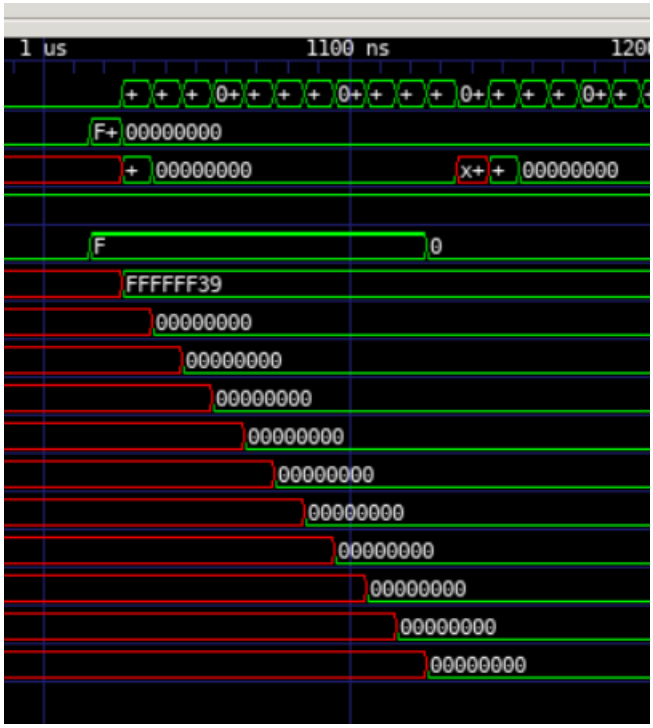
RAM control

Tap ram:

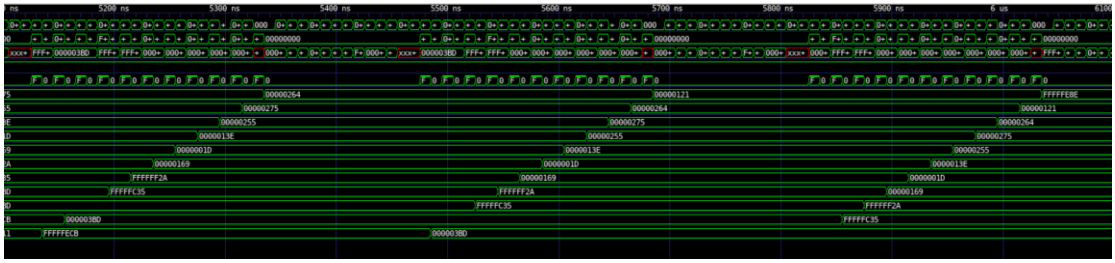


上圖是與 tb 拿 tap 寫入 ram 以及與 tb 對答案的波型，下面則是每一次 compute fir 時取出值的過程，tap 這邊相對來說比較方便因為並不需要做 shift。

Data ram:



一開始存取時先進行歸零並讀入第一筆 data



可以看到對於每一筆 fir 處理完我都會對整個 ram 進行重新讀寫的 shift

312510147 陳建嘉

FSM

LITE:

```
always@(posedge axis_clk) begin
    if(!axis_rst_n)     cs_lite <= LITE_idle;
    else                cs_lite <= ns_lite;
end



localparam LITE_idle     = 3'd0;
localparam LITE_wfinish  = 3'd1;
localparam LITE_arready  = 3'd2;
localparam LITE_rreq     = 3'd3;
localparam LITE_read     = 3'd4;
localparam LITE_done     = 3'd5;
```

```
always@(*) begin
    case(cs_lite)
        LITE_idle: begin
            if(tap_cnt >= Tape_Num)         ns_lite = LITE_done;
            else if(arvalid)                    ns_lite = LITE_arready;
            else if(wready_reg && awready_reg)  ns_lite = LITE_wfinish;
            else                                ns_lite = cs_lite;
        end
        LITE_wfinish:// by the time, axilite has already received awaddr and wdata
            ns_lite = LITE_idle;
        LITE_arready: begin
            if(arready && arvalid)  ns_lite = LITE_rreq;
            else                    ns_lite = cs_lite;
        end
        LITE_rreq: begin
            if(rready)  ns_lite = LITE_read;
            else        ns_lite = cs_lite;
        end
        LITE_read:      ns_lite = LITE_idle;
        LITE_done:  if(cs_str == STR_OUT && finish_flag) ns_lite = LITE_idle;
                    else ns_lite = LITE_done;
        default:        ns_lite = LITE_idle;
    endcase
end
```

312510147 陳建嘉

STREAM: 這邊我就定義了比較多 state 去處理

```
localparam STR_IDLE = 1;
localparam STR_FIRST = 9;
localparam STR_RESET_DATARAM = 2;
localparam STR_WAIT_LITE = 3;
localparam STR_CAL = 4;
localparam STR_OUT = 5;
localparam STR_SHIFT_READ = 6;
localparam STR_SHIFT_WRITE = 7;
localparam STR_NEWIN = 8;
localparam STR_STORE = 10;



always@(posedge axis_clk) begin
    if(!axis_rst_n)     cs_str <= STR_IDLE;
    else                cs_str <= ns_str;
end
always@(*) begin
    case(cs_str)
        STR_IDLE: ns_str = STR_FIRST;
        STR_FIRST:
            if(ss_tvalid) ns_str = STR_RESET_DATARAM;
            else ns_str = cs_str;
        STR_RESET_DATARAM: begin
            if(cnt >= 10) ns_str = STR_WAIT_LITE;
            else ns_str = cs_str;
        end
        STR_WAIT_LITE: begin
            if(cs_lite == LITE_done) ns_str = STR_CAL;
            else ns_str = cs_str;
        end
        STR_CAL:
            if(cnt > 10) ns_str = STR_OUT;
            else ns_str = cs_str;
        STR_OUT:    if(finish_flag) ns_str = STR_IDLE;
                    else ns_str = STR_SHIFT_READ;
        STR_SHIFT_READ: ns_str = STR_SHIFT_WRITE;
        STR_SHIFT_WRITE:
            if(cnt == 10) ns_str = STR_NEWIN;
            else ns_str = STR_SHIFT_READ;
        STR_NEWIN:
            if(ss_tvalid) ns_str = STR_STORE;
            else ns_str = cs_str;
        STR_STORE: ns_str = STR_CAL;
        default : ns_str = STR_IDLE;
    endcase
end
```