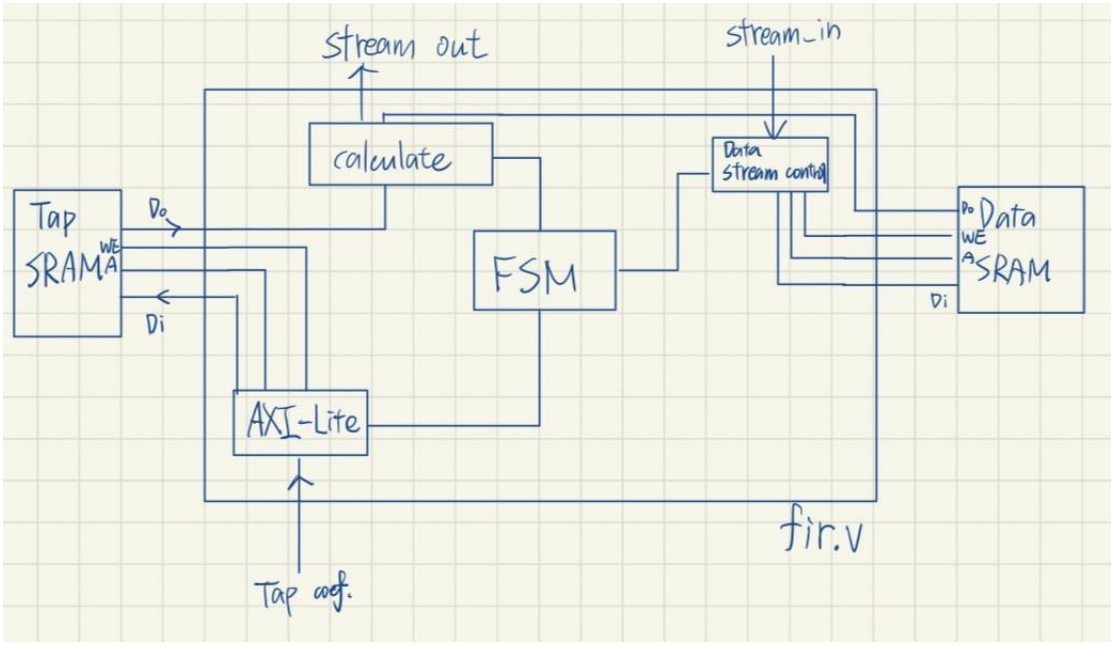
NYCU 電子研究所 呂紹愷 311510187 Lab3

- **Block Diagram**

  電路會先讀取 tap coef.，而在這次 design 中我一次就先讀完，並都存入 TapRAM，同時我也先把 data 也一次讀取 11 個值，並放入 dataRAM，接著 tb 會先檢查我們的 coef.是否正確，正確才會繼續跑。利用 FSM 去控制在前 11 次的乘加(因為前 11 次不會乘完全部的 coef. ，因此我分成前 11 次和後面的乘加)，前 11 次乘加完就先輸出了，然後跳至 11 次後面的乘加，並要特別注意 dataRAM 和 TapRAM 的是怎麼 shift 的，在後面的報告中有更詳細的描述，控制好 data 和 tap 就可以完成此次的 FIR design 了。



- **Describe operation**
  - ◆ How to receive data-in and tap parameters and place into SRAM

  I. data-in

  data 是透過 AXI-stream 來做傳輸，當我們把 ss_tready 拉起來，代表我們可以收 data 了，這時候 tb 就會吐值到 design。在這個 lab 中，我先把 11 個 data 放進 data SRAM，同時也把 11 個 tap 值也放進 SRAM，到計算的 state 再開始做運算，值都成完時再把舊的 data 洗掉，重新再拉 ss_tready 跟 tb 要值。

  II. tap parameters

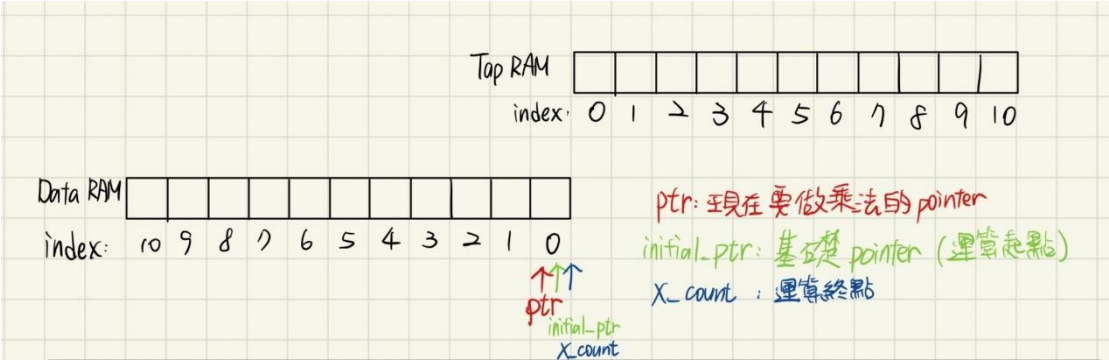
  tap parameters 是透過 AXI-Lite 來做傳輸，主要要熟悉 AXI-lite 的 protocol，tb 會先傳送 data 的地址，我們要拉 awready high，這樣才知道現在傳的 data address，接著 tb 會把 wready 拉 high，代表 tb 方準備好可以寫入了，而我們的 design 準備好時再把 wvalid 拉 high，tb 就會吐值了。在這個 lab 中我也是先把 11 個值都先讀進 SRAM 存好，後

面到運算的 state 時再開始運算。

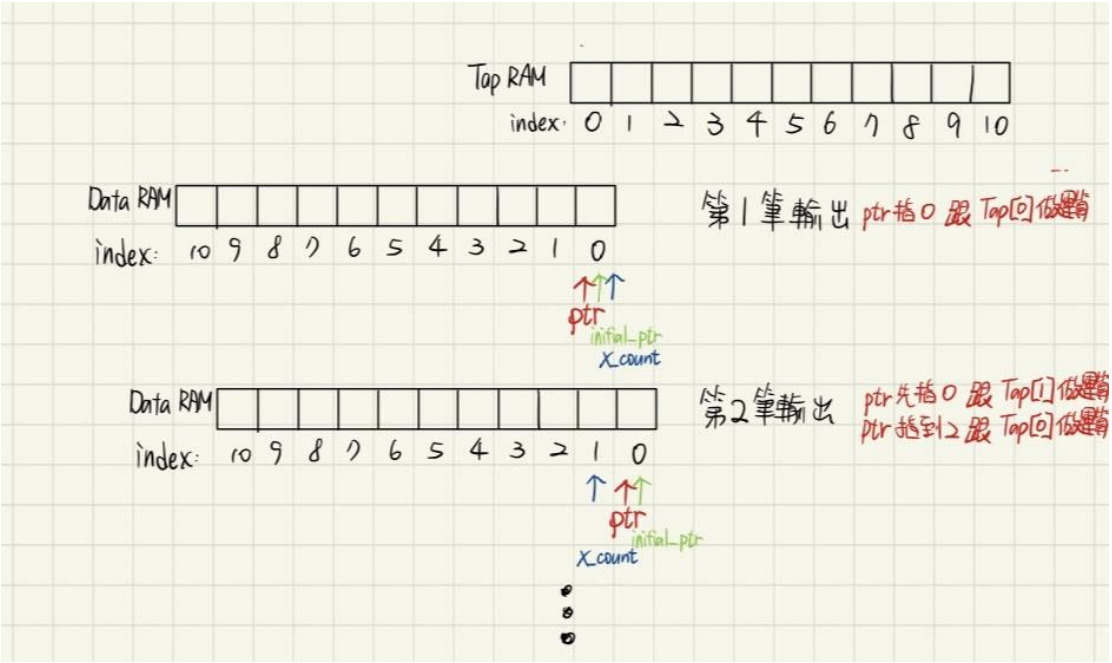
III. SRAM

SRAM 的操作就簡單許多，要注意的地方就三點，1. data 的地址 2. write enable 3.輸入 data，把這三樣東西準備好，並對齊，就可以在正確的時間地點輸入進 SRAM。

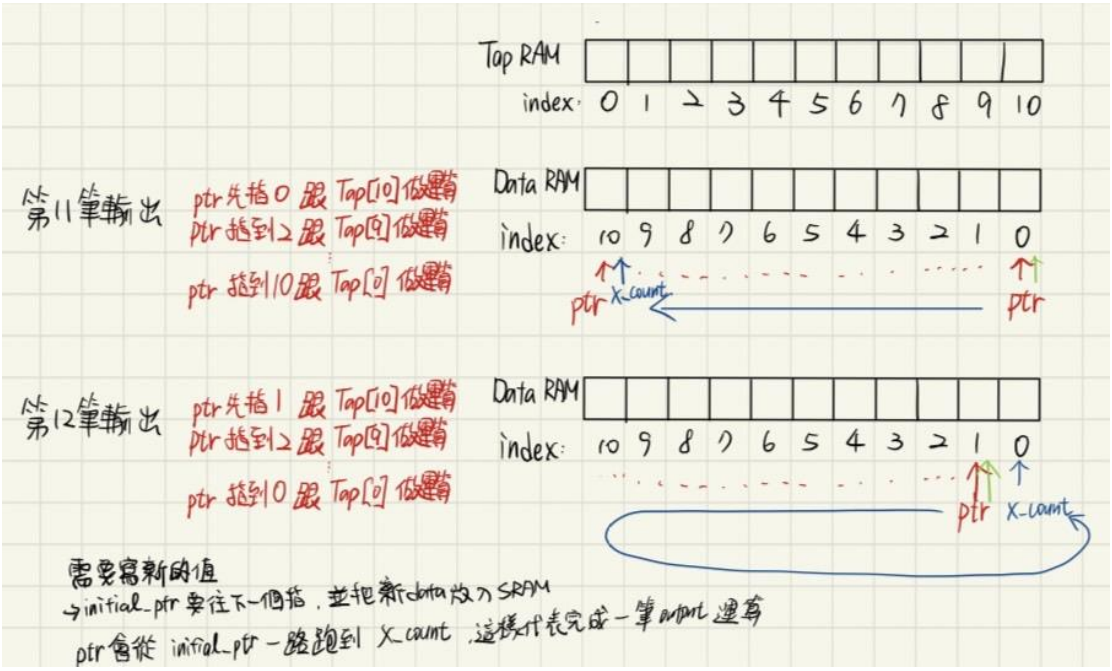
◆ How to access shiftram and tapRAM to do computation



上圖為 DataRAM 和 TapRAM 的示意圖，會去控制 ptr，來決定這次要從 SRAM 讀哪個位置，並讓 ptr 從 initial_ptr 跑到 X_count 來完成一次輸出運算。



我分為前 11 次運算以及 12~600 次運算為一組，在前 11 次不會乘滿整個 tap，上圖為第 1 筆以及第 2 筆的運作方式，以此類推至第 11 筆。11 筆後就是全乘了，而且會需要寫新值進去 dataRAM，initial_ptr 也會改變，如下圖所示，藉由控制 ptr 來做讀取和寫，如下方波形圖所示，將 tap, data 的要乘的位置對好，再做運算。



- How ap_done is generated

  使用計數器來看現在是第幾筆 data 輸入。

```
always @(posedge axis_clk or negedge axis_rst_n) begin
  if(~axis_rst_n)
    ap_done <= 0;
  else if(pattern_number == 600)
    ap_done <= 1;
  else
    ap_done <= ap_done;
end
```

- How ap_start is generated

  當 awaddr==0 和 wdata==1 時就拉起來，為了讓它 hold 一個 cycle 就好，所以使用 state 來控制。

```
always @(posedge axis_clk or negedge axis_rst_n) begin
  if(~axis_rst_n)
    ap_start <= 0;
  else if(awaddr == 0 && wdata == 1 && cs != input_1 )
    ap_start <= 0;
  else if(awaddr == 0 && wdata == 1 && cs == input_1 )
    ap_start <= 1;
  else
    ap_start <= ap_start;
end
```

- How ap_start is generated

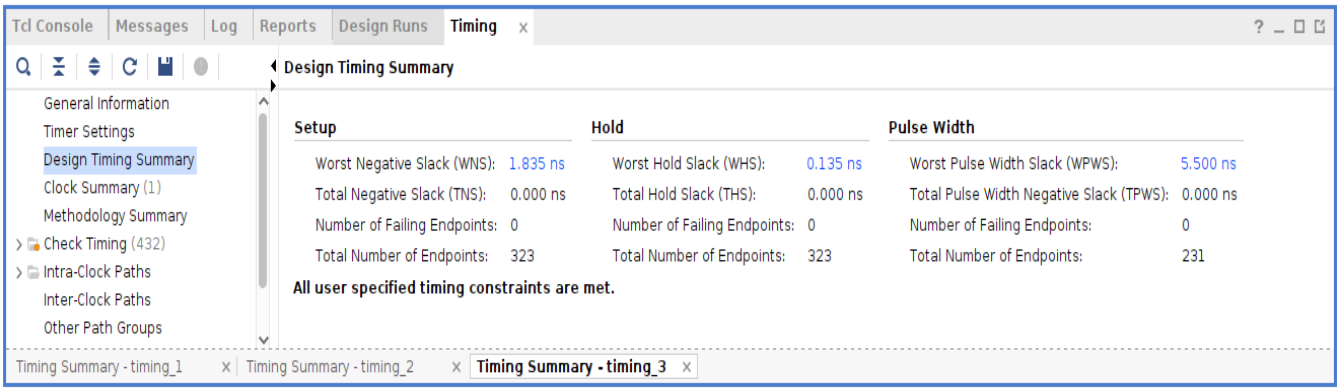
  idle 的控制相對簡單，只需要跟 start, done 做好搭配即可。

```
always @(posedge axis_clk or negedge axis_rst_n) begin
  if(~axis_rst_n)
    ap_idle <= 1;
  else if(ap_start)
    ap_idle <= 0;
  else if(ap_done)
    ap_idle <= 1;
  else
    ap_idle <= ap_idle ;
end
```

- Resource usage

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	314	0	0	53200	0.59
LUT as Logic	314	0	0	53200	0.59
LUT as Memory	0	0	0	17400	0.00
Slice Registers	233	0	0	106400	0.22
Register as Flip Flop	230	0	0	106400	0.22

- Timing Report



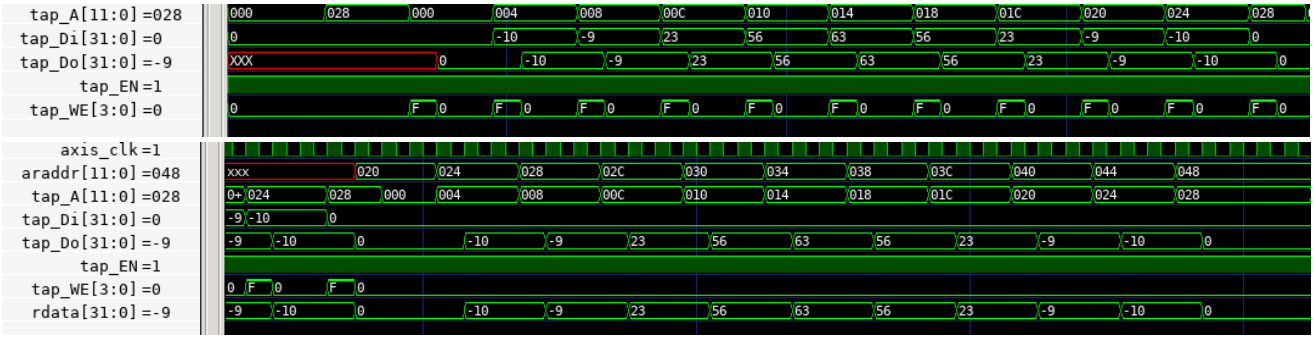
longest path

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             1.835ns  (required time - arrival time)
  Source:                 genblk1.cs_reg[1]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@6.000ns period=12.000ns})
  Destination:            genblk1.multi_result_reg[29]/D
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@6.000ns period=12.000ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            12.000ns  (axis_clk rise@12.000ns - axis_clk rise@0.000ns)
  Data Path Delay:        10.060ns  (logic 7.555ns (75.096%)  route 2.505ns (24.904%))
  Logic Levels:           8  (CARRY4=4 DSP48E1=2 LUT5=1 LUT6=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 14.128 - 12.000 )
    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
```

```
Slack (MET) :             1.841ns  (required time - arrival time)
  Source:                 genblk1.cs_reg[1]/C
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@6.000ns period=12.000ns})
  Destination:            genblk1.multi_result_reg[31]/D
                            (rising edge-triggered cell FDCE clocked by axis_clk  {rise@0.000ns fall@6.000ns period=12.000ns})
  Path Group:             axis_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            12.000ns  (axis_clk rise@12.000ns - axis_clk rise@0.000ns)
  Data Path Delay:        10.054ns  (logic 7.549ns (75.081%)  route 2.505ns (24.919%))
  Logic Levels:           8  (CARRY4=4 DSP48E1=2 LUT5=1 LUT6=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 14.128 - 12.000 )
    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
```

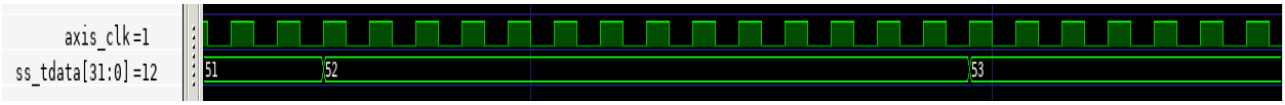
- Simulation Waveform
  - Coefficient program, and read back

    先把 11 個 coef.寫進 TapRAM，tb 要檢查時則透過 AXI-lite 去做資料傳輸。



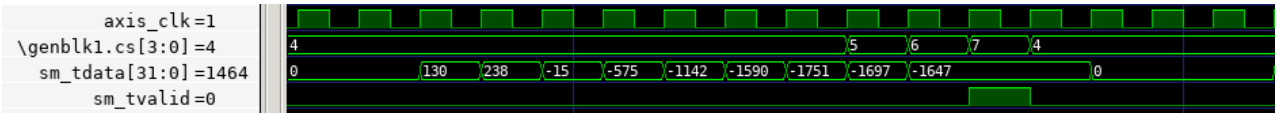
  - Data-in stream-in

    前 11 筆 data 直接拿，後面 data 則是有需要才拿



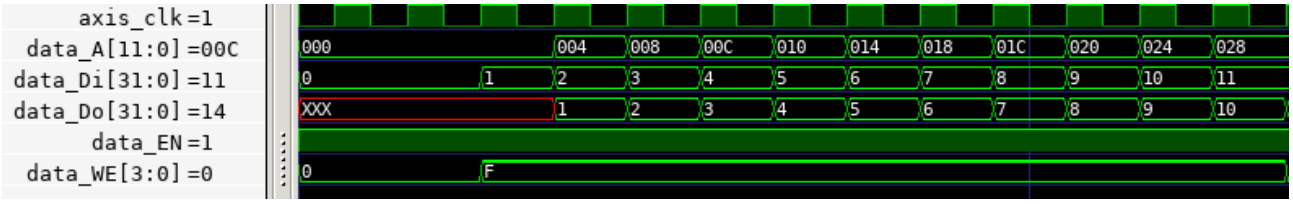
  - Data-out stream-out

    在 cs ==7 的時候輸出正確 data



  - Bram access control

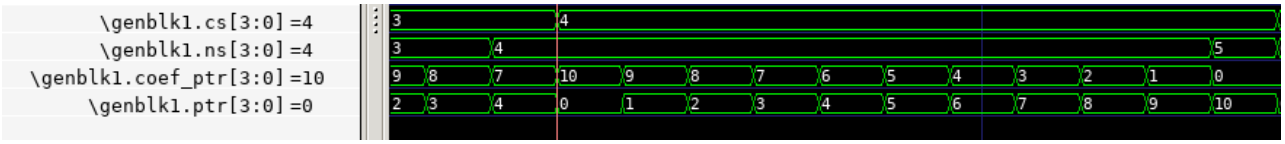
    前 11 筆 data 直接連續存進 SRAM



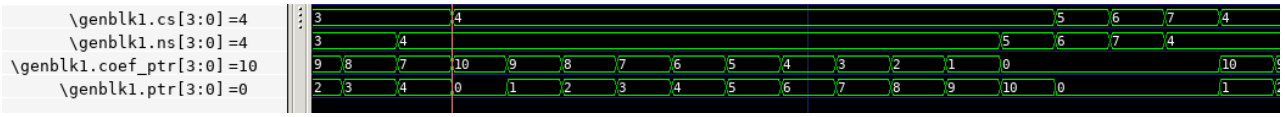
- FSM

  在 state1 是把要存進 tapRAM 的地址準備好，state2 時寫入 tapRAM

在 state 3,4 時，會做運算，把 tap 和 data 的值都對齊來做乘法，3, 4state 差別在於是否有乘滿整個 tap coef.。



在 5, 6, 7 則是做最後一次乘法，然後累加，輸出數值，並把新 data 寫進 dataRAM。




```
always@(*) begin
    case(cs)
        idle: //0
        begin
          ns = input_1;
        end

        input_1: // 1
        // addr
        begin
          if(awaddr == 12'h000 && wdata == 32'd1)
            ns = calculte_1 ;
          else if(awvalid)
            ns = input_2;
          else
            ns = input_1;
        end
        input_2:
        // write
        begin
          if(awaddr == 12'h000 && wdata == 32'd1)
            ns = calculte_1;
          else if(wvalid)
            ns = input_1;
          else
            ns = input_2;
        end

        calculte_1 : //2
        begin
          if(out_count == 10)
            ns = calculte_2;
          else
            ns = calculte_1;
        end
        calculte_2 : //3
        begin
          if(X_count == ptr )
            ns = last_multi;
          else
            ns = calculte_2;
        end

        last_multi:
        begin
          ns = accumulate;
        end

        accumulate:
        begin
          ns = out_state;
        end

        out_state :
        begin
          ns = calculte_2;
        end

        default:
            ns= idle;
    endcase
end
```