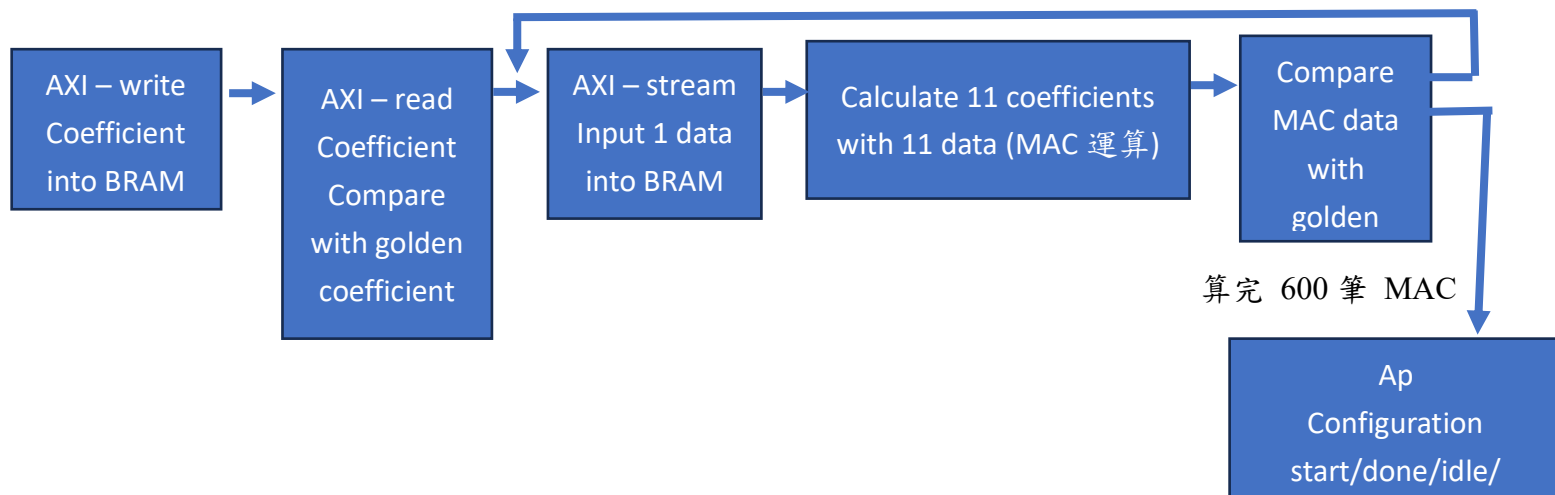


Block Diagram

• Datapath – dataflow

dataflow 可以分為以下階段：



• Control signals

// axi write

output awready, wready,

input awvalid, awaddr, wvalid, wdata,

以上的訊號為，axi-write 要將 wdata 寫進 tap 的 BRAM 時所要用的訊號
wready, wvalid handshake 時 wdata 寫進 BRAM。

// axi read

output arready, rdata,

input rready, arvalid, araddr, rvalid,

以上的訊號為，axi-read 要確認 tap BRAM 裡面的係數又正確的寫進去時
所要用檢查，rready && rvalid handshake 會把 rdata 丟到 tb 做檢查。

axi_read_cnt 為一個計算現在讀到哪裡的 counter。

// axi stream

input ss_tvalid, ss_tdata, ss_tlast, sm_tready,

output ss_tready, sm_tvalid, sm_tdata, sm_tlast,

ss_tvalid 來時代表要將 ss_tdata 丟到 DATA BRAM 裡面，進行完 11 筆的
FIR 運算後把 sm_tvalid 拉起來，將 sm_tdata 丟到 tb 做檢查。

// bram for tap RAM

output tap_WE, tap_EN, tap_Di, tap_A,

input tap_Do,

// bram for data RAM

output data_WE, data_EN, data_Di, data_A,

input data_Do, axis_clk, axis_rst_n

WE 做為可以寫 BRAM 的信號，EN 為可以 access BRAM 的信號，Di 為寫進 BRAM 的 data，A 為寫進 BRAM 的位置。

// FIR 控制訊號

fir_start_flag 為開始進行 FIR 計算的 flag。fir_mul_data 為計算 tap 和 data ram 出來的 data 之相乘值。fir_cal_data 為計算累加 MAC 值的 counter。fir_mul_cal 為一個計算現在做到第幾筆 FIR 的 counter。

Describe operation

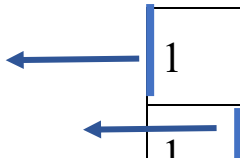
- How to receive tap parameters and place into BRAM

在 AXI – write phase 中，每當 input signal awready 拉起來時，對應拉起 awvalid，同樣的 wready 來時也拉起 wvalid，wready && wvalid handshake 時代表可以將 wdata 寫進 BRAM，每次 handshake 都會將 awaddr(要寫進 BRAM 的地址 – 32)，要寫的那 T 也要將 tap_WE, tap_EN 對應拉起來

- How to receive data-in and place into BRAM

每當 axi-stream 中 ss_tvalid 拉起來的時候，把新進的 ss_tdata 寫進 data BRAM 中，寫的地址為 write pointer 指向的地方，write pointer 每次會加 4，若 write pointer=40，那下次就要指到 0。

Data BRAM address:	0	4	8	12	16	20	24	28	32	36	40
	1	0	0	0	0	0	0	0	0	0	0
	1	2	0	0	0	0	0	0	0	0	0

 write_pointer

1	2	3	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---

⋮

1	2	3	4	5	6	7	8	9	10	11
12	2	3	3	5	6	7	8	9	10	11
12	13	3	3	5	6	7	8	9	10	11

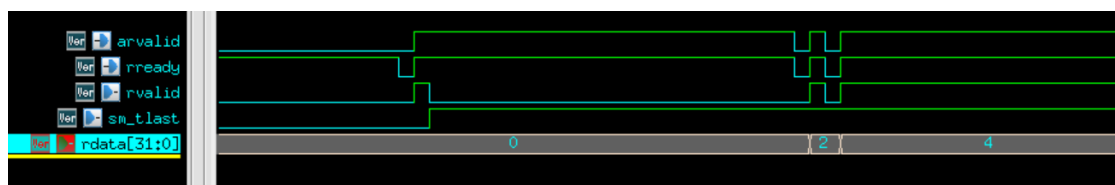
Tap BRAM address: 0 4 8 12 16 20 24 28 32 36 40

0	-10	-9	23	56	63	56	23	-9	-10	0
---	-----	----	----	----	----	----	----	----	-----	---

- How to access shiftram and tapRAM to do computation

每當要計算 11 筆 data 和 tap 係數相乘的 FIR 時拉起 fir_en，tap bram 每次都是從 0 開始讀取 tap_data，而 data BRAM 的方向則和 tap_BRAM 的讀取方向相反，(上面有讀取方向和起點的示意圖)，而且每次的起始位置是 write_pointer。

- How idle/ap_done/ap_idle is generated.



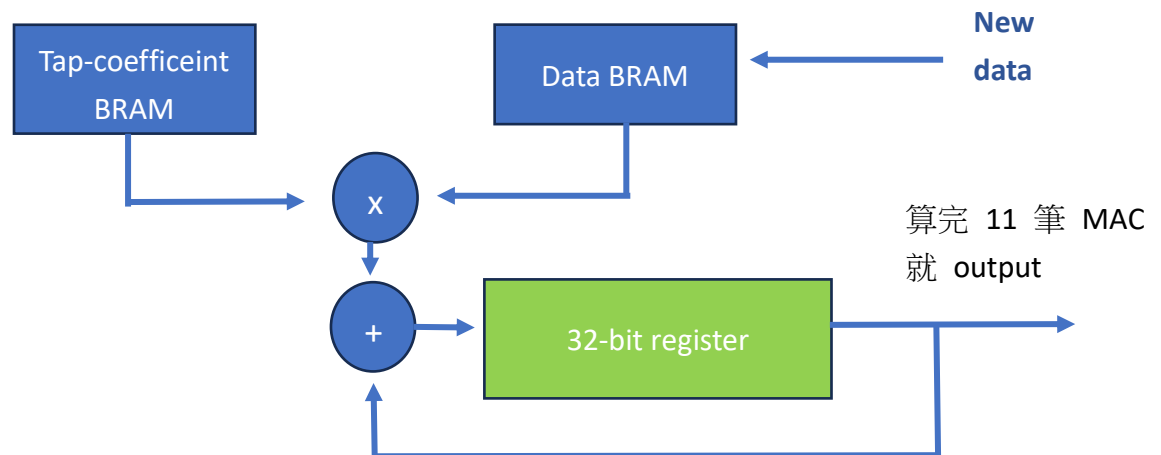
Ap_done 的檢查要等到，sm_tlast 從 design 拉起來的時候才會開始，拉起 rvalid 讓 tb 可以用 config read check 來對答案，而答案依照 dle/ap_done/ap_idle 的順

序為 0,2,4 來設定 rdata

Resource usage: including FF, LUT, BRAM

Fir.v 內只有用到一個 32 bit 的 register 作為 MAC

運算值得暫存，沒有 instantiate LUT 和 BRAM。



Timing Report

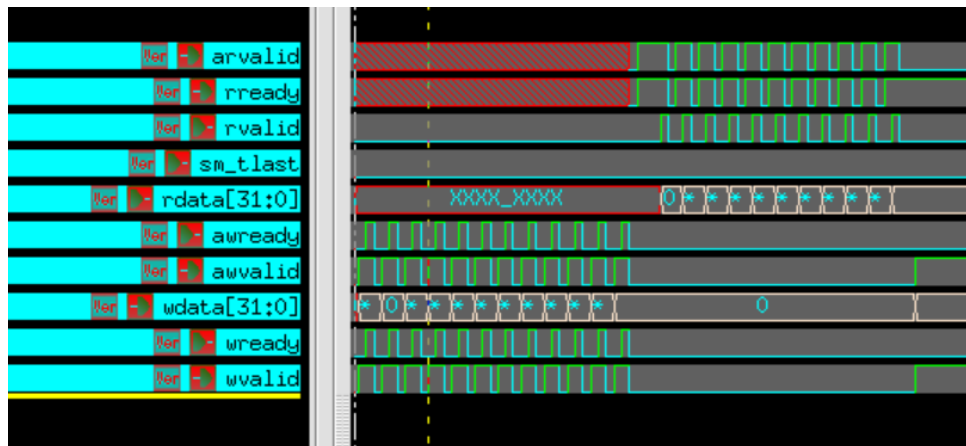
- Parse 過整個 timing report 沒有負的 slack，也沒有 latch

```
12. checking latch_loops (0)
-----
There are 0 combinational latch loops in the design through latch input

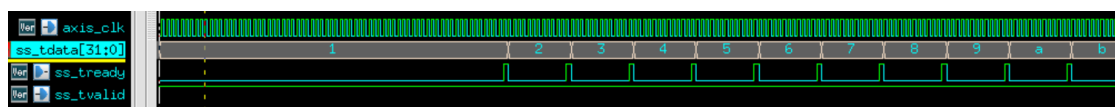
-----
| Design Timing Summary
|-----
| WNS(ns)    TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints  WHS(ns)  THS(ns)  THS Failing Endpoints  THS Total Endpoints  WPNS(ns)
| TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
|-----
| NA         inf      0.000      NA         0          905      inf      0.000      0          905      NA
```

Simulation Waveform, show

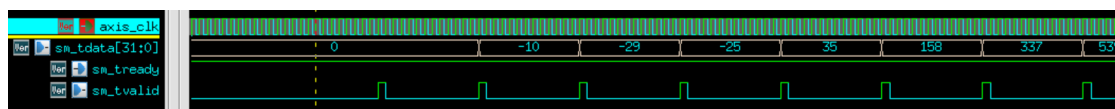
- Coefficient program, and read back



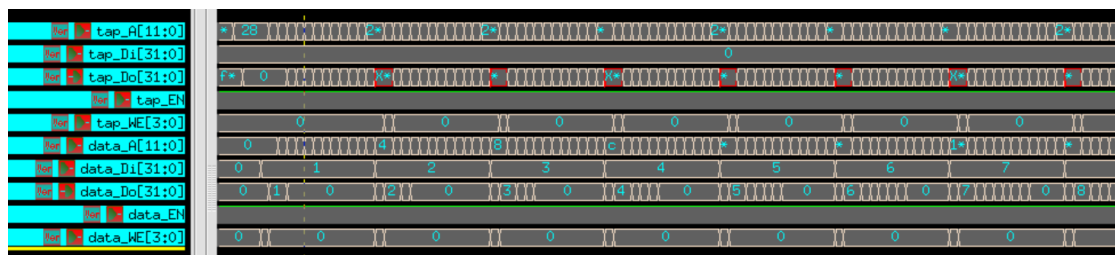
- Data-in stream-in



- Data-out stream-out



- RAM access control



- 沒有用 FSM, 全部用 counter, 用 FSM, datapath 會更清楚