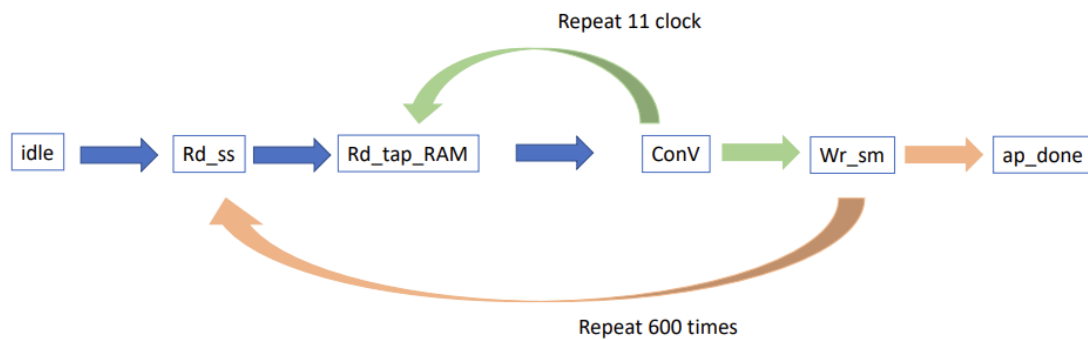


Lab3 Report

M11202115 林翰祥

- Block Diagram



- Rd_ss:
Data_WE 設為 1，將 ss_data 讀的值寫入 data_RAM
- Rd_tap_RAM:
讀取 tap_RAM 的值，在 wvalid==1 時，將 wdata 寫入 tap_Di
- ConV:

- $y[t] = \sum (h[i] * x[t - i])$

將剛讀取的 tap Di 和 data Di 做相乘後，加上原來的值

```
245     if(cur_state==ConV)begin
246         temp<=data_Do*tap_Do+temp;
247         conv cnt<=conv cnt+1;
```

總共會花 11 個 clock 完成 ConV

- Wr_sm:
將 sm_tvalid 設為 1，將 ConVstate 算出的值寫入 sm_tdata

- Describe operation

- How to receive data-in and tap parameters and place into SRAM

當 testbench 拉起 awvalid 時，將 awready 和 wready 設為 1，

接著將 tap_WE 設為 1，開始將 awaddr 寫入 tap_A，wdata 寫入 tap_Di。

```
//=====
//aw&w
always@(posedge axis_clk,negedge axis_rst_n)begin
  if(~axis_rst_n)begin
    awready<=0;
    wready<=0;
  end
  else begin
    if(awvalid)begin
      awready<=1;
      wready<=1;
    end
    else begin
      awready<=0;
      wready<=0;
    end
  end
end
end
//=====

169 //=====
170 //coef write to tap_ram
171 always@(posedge axis_clk,negedge axis_rst_n)begin
172   if(!axis_rst_n)begin
173     tap_WE<=0;
174     tap_EN<=0;
175     tap_Di<=0;
176     tap_A<=0;
177     tap_cnt<=0;
178   end
179   else begin
180     if(cur_state==ConV)begin
181       tap_WE<=4'b0000;
182       tap_EN<=1;
183       tap_A<=cnt;
184     end
185     else if(awvalid&awready&tap_cnt<12)begin
186       tap_WE<=4'b1111;
187       tap_EN<=1;
188       tap_A<={5'd0,awaddr[6],awaddr[4:0]};
189       tap_cnt<=tap_cnt+1;
190     end
191     else if(arvalid&arready)begin
192       tap_EN<=1;
193       tap_A<={5'd0,araddr[6],araddr[4:0]};
194     end
195     else begin
196       tap_WE<=4'b0000;
197       tap_EN<=1;
198       tap_Di<=tap_Di;
199       tap_A<=tap_A;
200     end
201   end
202   if(wvalid&wready)
203     tap_Di<=wdata;
204   else
205     tap_Di<=tap_Di;
206 end
207 end
208 //=====
```

- How to access shiftram and tapRAM to do computation

當狀態機到 ConV 時，總共會花 11 個 clock 來完成 ConV，

每個 clock 都會去讀取 data_RAM 和 tap_RAM，每個 address

對應的 data 做相乘，算完後狀態機跳到 wr_sm，將算完的

temp 輸出到 sm_tdata，將 sm_tvalid 拉到 1。

$$y[t] = \sum (h[i] * x[t - i])$$

```

239 always@(posedge axis_clk,negedge axis_rst_n)begin
240     if(!axis_rst_n)begin
241         cnt<=0;
242         temp<=0;
243         conv_cnt<=0;
244     end
245     else begin
246         if(cur_state==ConV)begin
247             temp<=data_Do*tap_Do+temp;
248             conv_cnt<=conv_cnt+1;
249             if(cnt==40)
250                 cnt<=cnt;
251             else
252                 cnt<=cnt+4;
253         end

226 //=====
227 //sm
228 always@(posedge axis_clk,negedge axis_rst_n)begin
229     if(!axis_rst_n)
230         sm_tvalid<=0;
231     else begin
232         if(cur_state==wr_sm)
233             sm_tvalid<=1;
234         else
235             sm_tvalid<=0;
236     end
237 end

```

- How ap_done is generatedResource usage

每 write 進 sm 一個值，p_cnt 就會加 1，當 sm 輸出 600 個值

也就是 p_cnt==599 時，將 sm_tlast 設為 1。

當 sm_tlast 為 1 時，狀態機進入 ap_done，將 ardata<=00

rdata<=32'h02，ap_done 後進入 ap_idle，將 ardata<=00

rdata<=32'h04，節素 testbench。

```

281 always@(posedge axis_clk, negedge axis_rst_n) begin
282     if(!axis_rst_n)
283         sm_tlast<=0;
284     else begin
285         if(cur_state==wr_sm&p_cnt==599)
286             sm_tlast<=1;
287         else
288             sm_tlast<=sm_tlast;
289     end
290 end
291
292 always@(posedge axis_clk, negedge axis_rst_n) begin
293     if(!axis_rst_n)
294         p_cnt<=0;
295     else begin
296         if(cur_state==wr_sm)
297             p_cnt<=p_cnt+1;
298         else
299             p_cnt<=p_cnt;
300     end
301 end

```

- How does filter read correct address from tap_RAM and data_RAM?

每做完一個 ConV，下一次的 data_RAM 就會去讀入

ss_data，讀進去的位址會直接替換上次 address+1，此設計會

利用一個 head 的 pointer 來去指向 data_RAM，透過這個

pointer 來去指向接下來的 11 clocks data_RAM address 要從哪

裡開始讀，

ConV 的第一個 clock 會讀 tap_RAM 的 address=0，

data_RAM 的 address 就會等於 head_pointer，透過這個

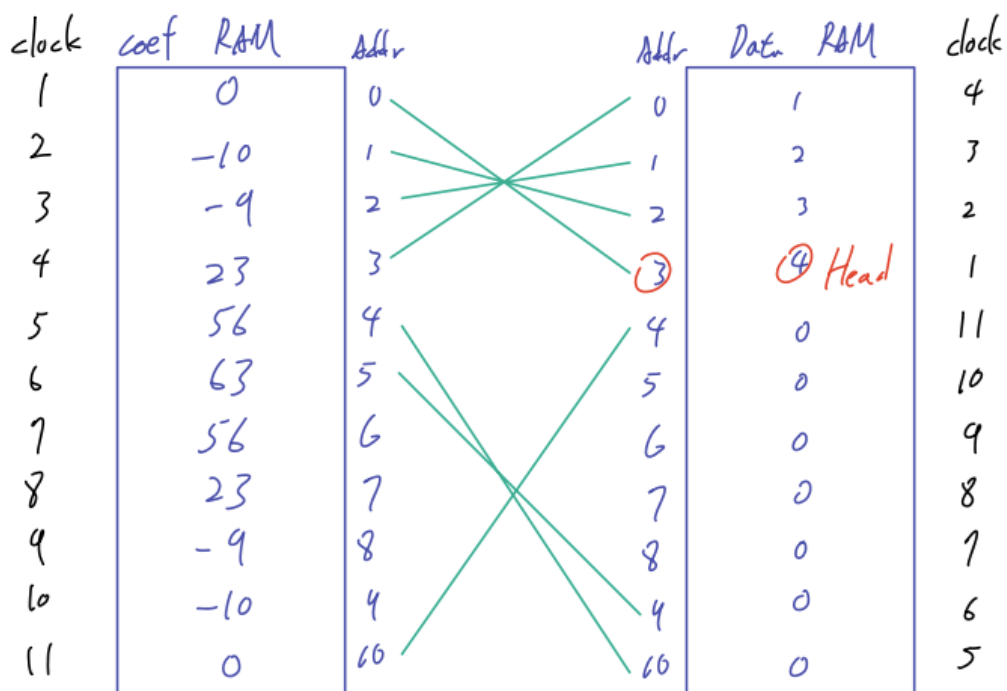
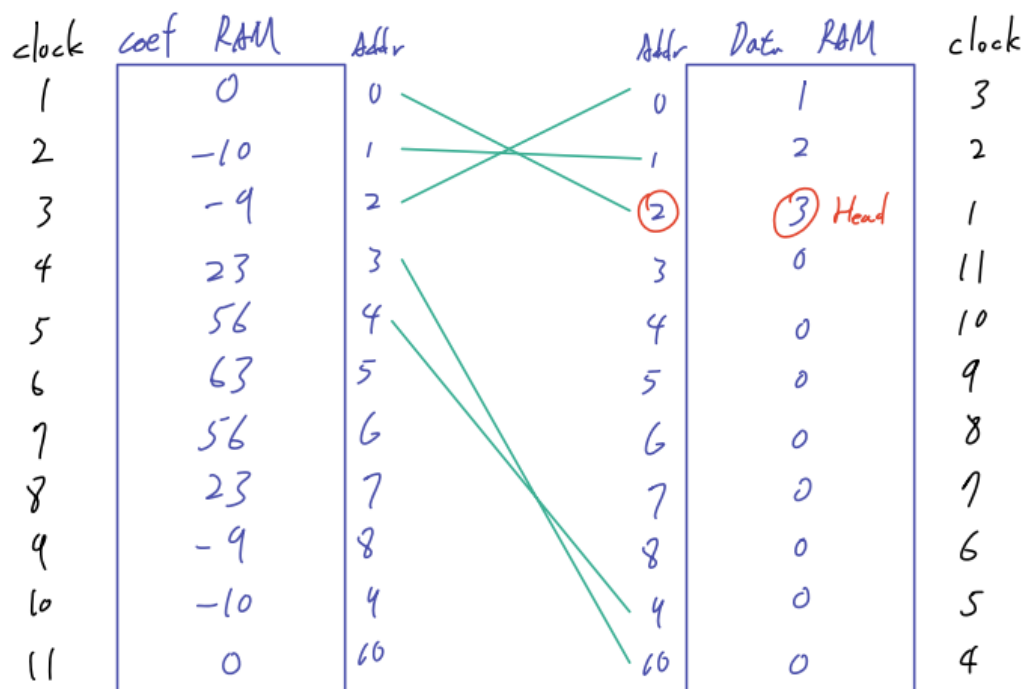
pointer 就可以不用 shift register 就完成 data_RAM 的寫入。

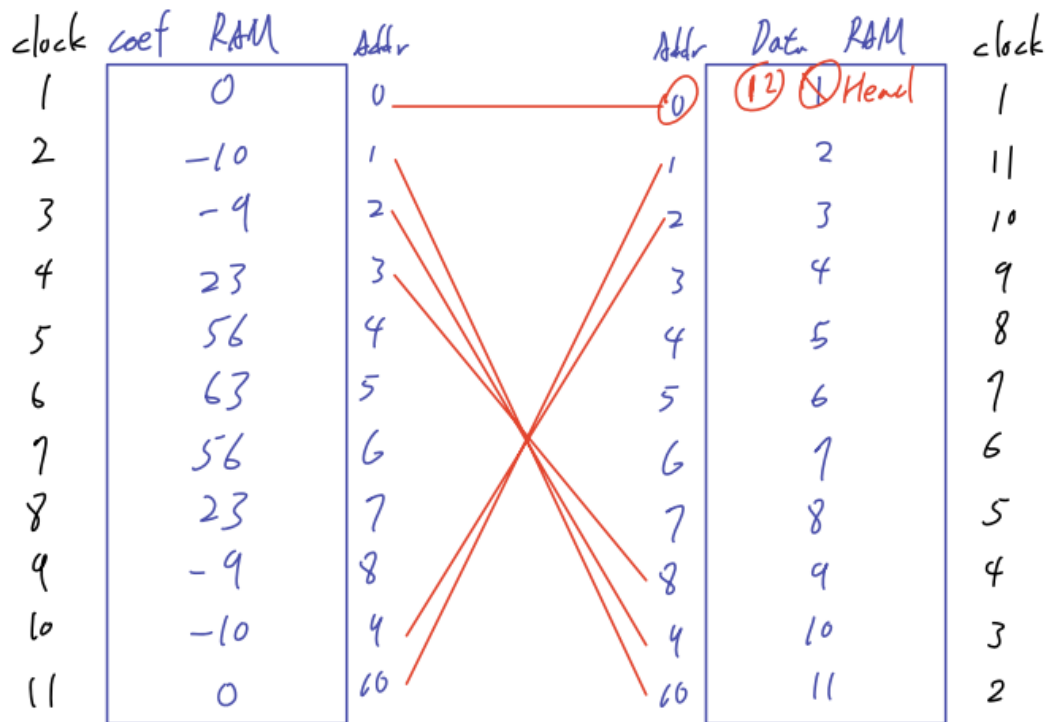
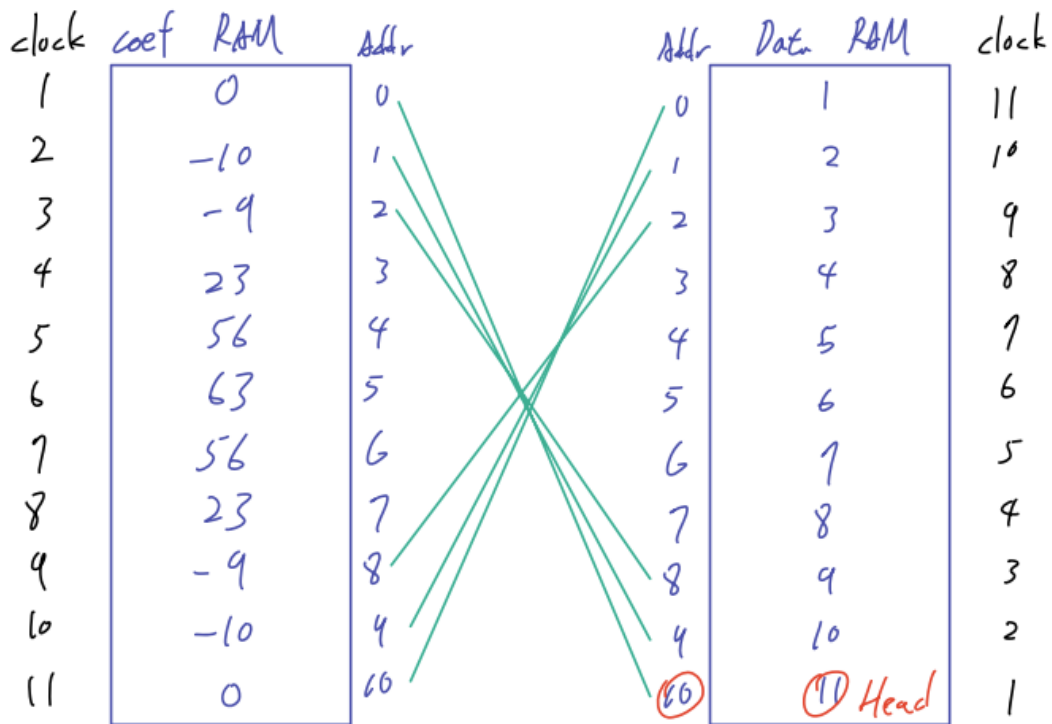
下圖為是讀取後座 ConV 的示意圖

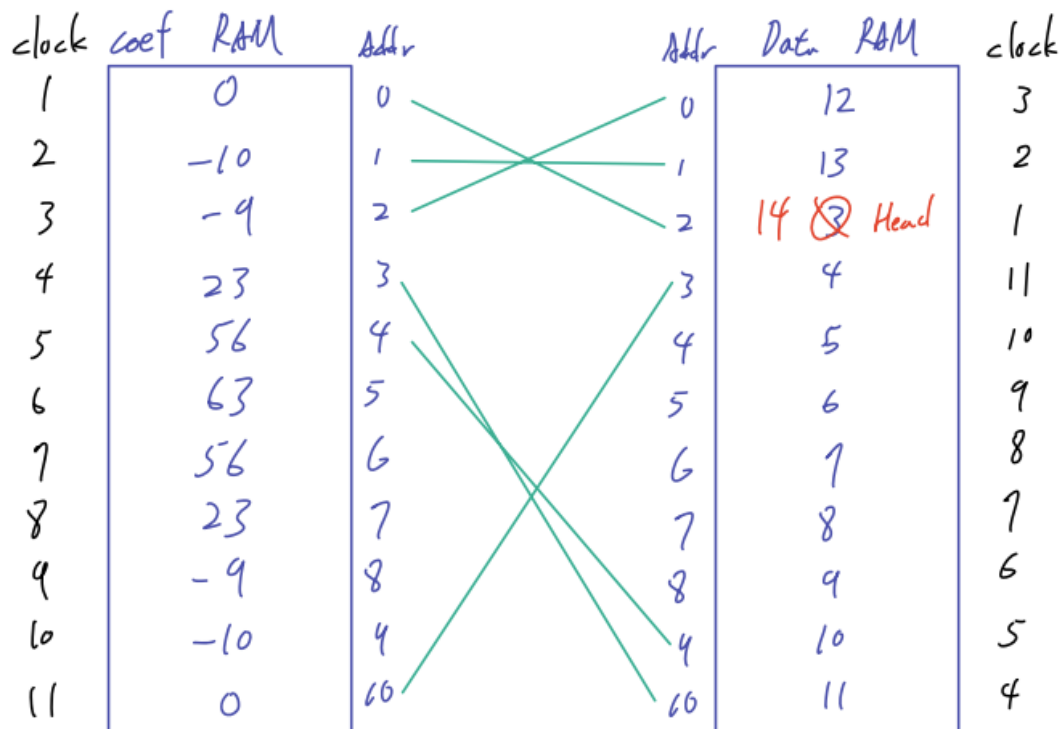
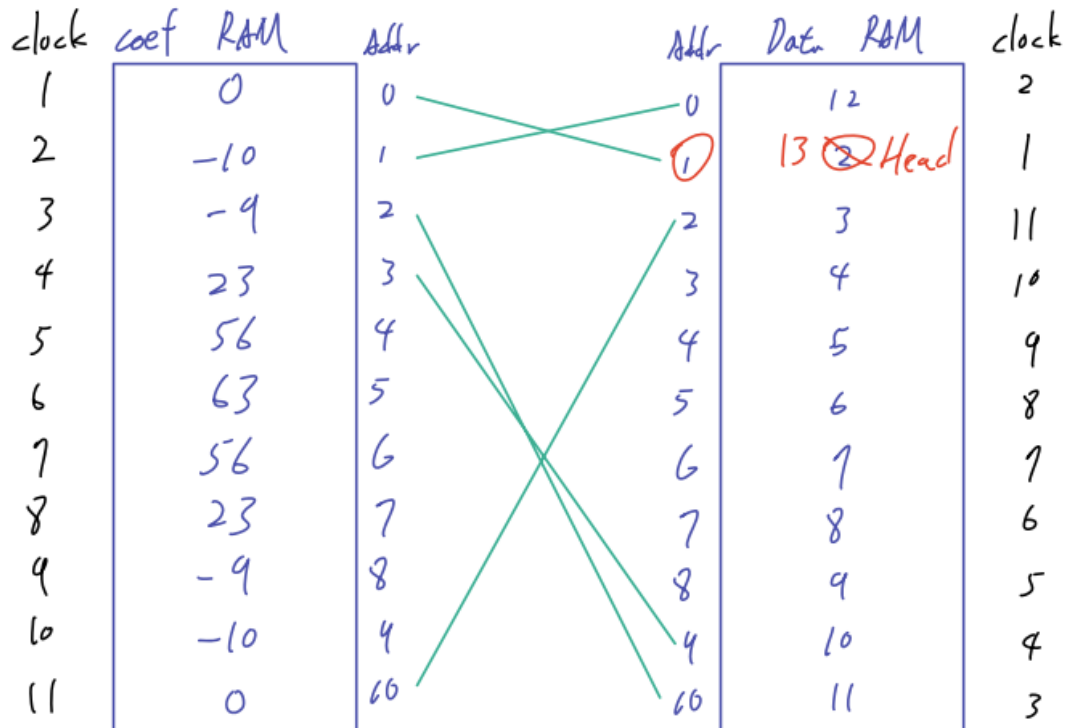
clock 1	data RAM [Head]	\times	coef[0]
clock 2	data RAM [Head-1]	\times	coef[1]
⋮	⋮		⋮
⋮	⋮		⋮
⋮	⋮		⋮
⋮	⋮		⋮
clock 11	data RAM [Head-10]	\times	coef[10]

clock	coef	RAM	Addr	Addr	Data	RAM	clock
1	0	0	0	0	(1) Head		1
2	-10	1	1	1	0		11
3	-9	2	2	2	0		10
4	23	3	3	3	0		9
5	56	4	4	4	0		8
6	63	5	5	5	0		7
7	56	6	6	6	0		6
8	23	7	7	7	0		5
9	-9	8	8	8	0		4
10	-10	4	4	4	0		3
11	0	16	16	16	0		2

clock	coef	RAM	Addr	Addr	Data	RAM	clock
1	0		0	0	1		2
2	-10		1	①	② Head		1
3	-9		2	2	0		11
4	23		3	3	0		10
5	56		4	4	0		9
6	63		5	5	0		8
7	56		6	6	0		7
8	23		7	7	0		6
9	-9		8	8	0		5
10	-10		9	9	0		4
11	0		10	10	0		3







- Syn utilization

Tcl Console	Messages	Log	Reports	Design Runs	Utilization	Timing
Hierarchy						
Hierarchy						
Summary						
<ul style="list-style-type: none"> ✓ Slice Logic <ul style="list-style-type: none"> ✓ Slice LUTs (1%) <ul style="list-style-type: none"> LUT as Logic (1%) 						
Name	1	Slice LUTs (53200)	Slice Registers (106400)	DSPs (220)	Bonded IOB (125)	BUFGCTRL (32)
N fir		233	212	3	321	1

- Timing Report

Period = 4.5

SYNTHESIZED DESIGN * - synth_1 | xc7z020clg400-1

Sources Netlist x ? _ □ ×

Project Summary x Device x fir.tb.v x bram11.v x fir_constrs.xdc x fir.v x Timing Constraints x

▼ Clocks (1)

Create Clock (1)

Create Generated Clock (0)

Rename Auto-Derived Clock (0)

Set Clock Latency (0)

Set Clock Uncertainty (0)

Set Clock Groups (0)

Set Clock Sense (0)

Set Input Jitter (0)

Set System Jitter (0)

Position Clock Name Period (ns) Rise At (ns) Fall At (ns) Add Clock Source Objects Source File

1 axis_clk 4.500 0.000 2.250 ☐ [get_ports axis_clk] fir_constrs.xdc

Double click to create a Create Clock constraint

All Constraints

Position Command Scoped Cell

1 create_clock -period 4.500 -name axis_clk -waveform {0.000 2.250} [get_ports axis_clk]

Apply Cancel

Tcl Console Messages Log Reports Design Runs Timing x

Design Timing Summary

General Information

Timer Settings

Design Timing Summary

Clock Summary (1)

Methodology Summary

Check Timing (296)

Intra-Clock Paths

Inter-Clock Paths

Other Path Groups

Timing Summary - timing_1

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.262 ns	Worst Hold Slack (WHS): 0.142 ns	Worst Pulse Width Slack (WPWS): 1.750 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: 378	Total Number of Endpoints: 378	Total Number of Endpoints: 213

All user specified timing constraints are met.

Period=5

Sources Netlist x ? _ □ ×

Project Summary x Device x fir.tb.v x bram11.v x fir_constrs.xdc x fir.v x Timing Constraints x

▼ Clocks (1)

Create Clock (1)

Create Generated Clock (0)

Rename Auto-Derived Clock (0)

Set Clock Latency (0)

Set Clock Uncertainty (0)

Set Clock Groups (0)

Set Clock Sense (0)

Set Input Jitter (0)

Set System Jitter (0)

Position Clock Name Period (ns) Rise At (ns) Fall At (ns) Add Clock Source Objects Source File

1 axis_clk 4.000 0.000 2.000 ☐ [get_ports axis_clk] fir_constrs.xdc

Double click to create a Create Clock constraint

All Constraints

Position Command Scoped Cell

1 create_clock -period 4.000 -name axis_clk -waveform {0.000 2.000} [get_ports axis_clk]

Apply Cancel

Tcl Console Messages Log Reports Design Runs Timing x

Design Timing Summary

General Information

Timer Settings

Design Timing Summary

Clock Summary (1)

Methodology Summary

Check Timing (296)

Intra-Clock Paths

Inter-Clock Paths

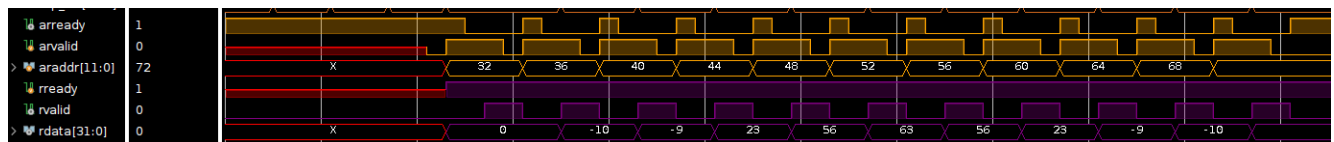
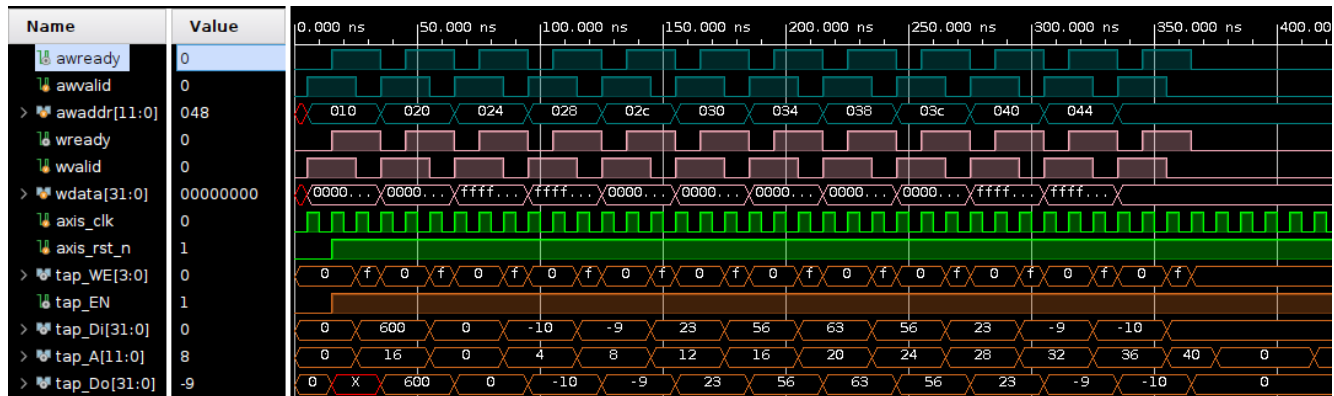
Other Path Groups

Timing Summary - timing_1

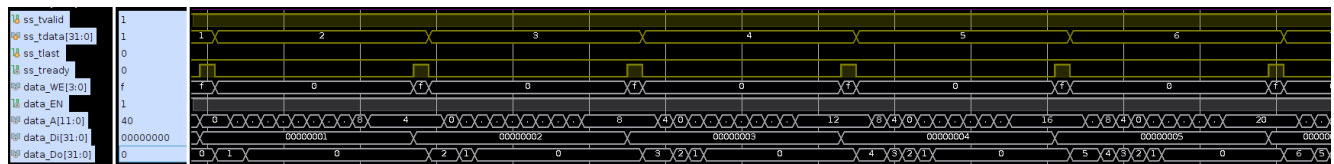
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): -0.238 ns	Worst Hold Slack (WHS): 0.142 ns	Worst Pulse Width Slack (WPWS): 1.500 ns
Total Negative Slack (TNS): -1.959 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 9	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 378	Total Number of Endpoints: 378	Total Number of Endpoints: 213

Timing constraints are not met.

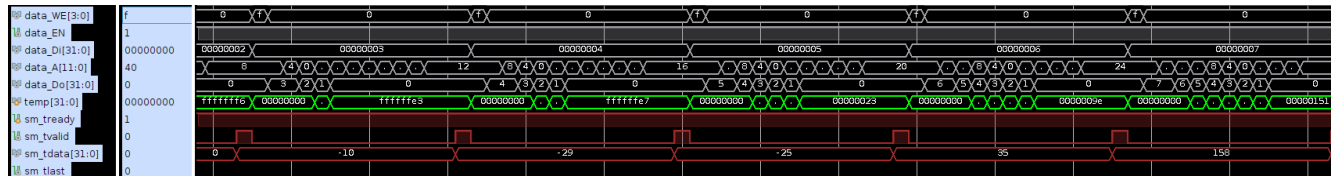
- Simulation Waveform
 - Coefficient program, and read back



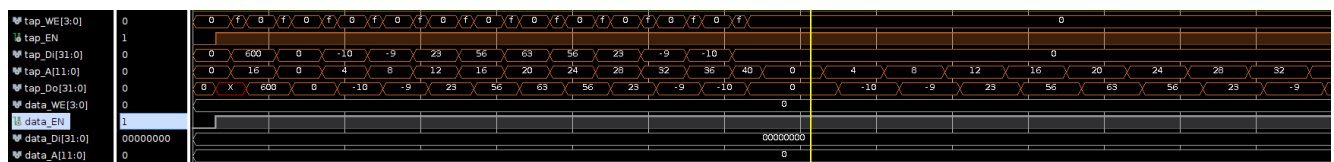
- Data-in stream-in

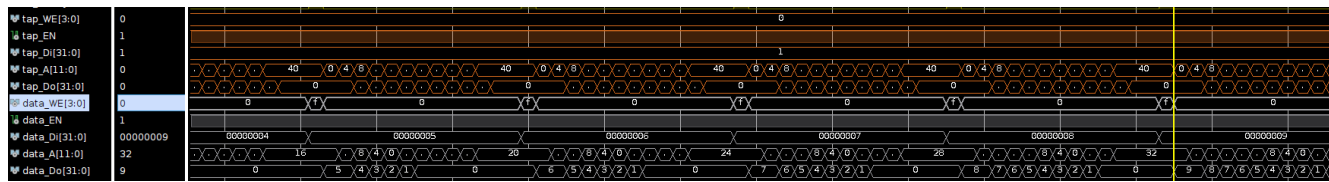


- Data-out stream-out



- RAM access control





• FSM

