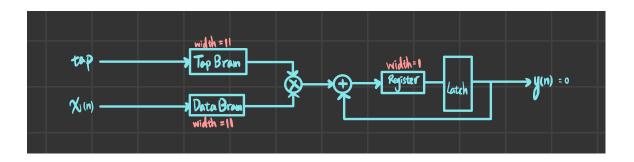
lab3_report

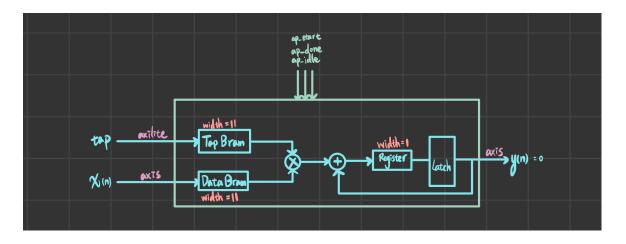
https://github.com/Josephood7/SoC-lab3.git

▼ Block Diagram

• Data path - dataflow



Control Signals



▼ Operation Description

- 1. Load tap coefficient into tap block ram through AXI4-Lite
- 2. load input data into data block ram through AXI4-Stream
- 3. When both tap and data ram are full, start calculation using 1 multiplier and 1 adder
- 4. When sm_tready && sm_tvalid, output the result, FIR operation is done
- 5. FIR enter idle mode

lab3_report

How to receive data-in and tap parameters and place into SRAM
Output arready then wait for arvalid to tell where to put the upcoming tap coefficients. Output the rvalid to wait for rready to handshake for tap.

When the FIR engine output the ss_tready signal and the data source input a ss_tvalid signal, a handshake is formed. The datas steam into the data RAM.

How to access shiftram and tapRAM to do computation
While ap_start is 1.

If the state of FIR is BUSY, we can put the output of the shiftram and the tapRAM directly into our FIR kernel.

How ap_done is generated

If (sm tvalid && sm tready && sm tlast) → ap done = 1

Which means that while the FIR engine is in an output mode, the ap_done is generated once the last data is transmitted.

▼ Resource Usage

- Flip-flop → 71
- Lookup Table → 186
- Bram → 2
- Adder → 1 for FIR, 2 for tap& data counters
- Multiplier → 1

▼ Timing Report

Timing on longest path, slack



▼ Simulation Waveform

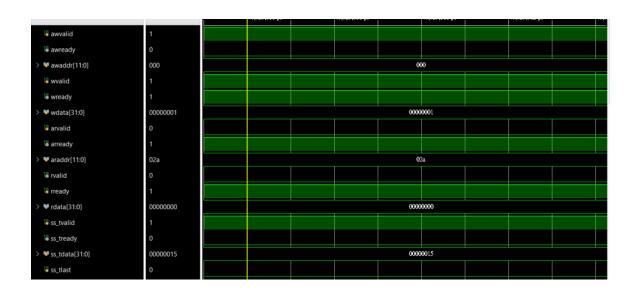
Coefficient program, and read back

lab3_report 2

/fir_tb/coef[0:10][31:0]	00000000,fffffff6,f	00000000 дишин дишиг д0000017,0000038,00000036,00000017 дишиг дишин д0000000
> 💆 [0][31:0]	00000000	0000000
> 🕨 [1][31:0]	ffffff6	mm6
> 🕨 [2][31:0]	ffffff7	91111177
> 🕨 [3][31:0]	00000017	00000017
> 💆 [4][31:0]	00000038	00000038
> 🕨 [5][31:0]	0000003f	000003f
> 💆 [6][31:0]	00000038	00000038
> 🕨 [7][31:0]	00000017	00000017
> 🕨 [8][31:0]	11111117	mmr7
> 🕨 [9][31:0]	11111116	mm6
> 🕨 [10][31:0]	00000000	0000000
/fir_tb/sm_tlast	1	
<pre>axis_rst_n</pre>	1	
N Ave. 14/512-01		0

/fir_tb/sm_tlast	1										
axis_rst_n	1										
> W tap_WE[3:0]	0	0									
16 tap_EN	1										
> ♥ tap_Di[31:0]	00000000	0000000									
> ♥ tap_A[11:0]	00c	00e									
> W tap_Do[31:0]	fffffff7	311117									
> V data_WE[3:0]	0				()					
la data_EN	1										
> V data_Di[31:0]	00000016	00000016									
> V data_A[11:0]	014	014									
> V data_Do[31:0]	00000018				0000	0018					

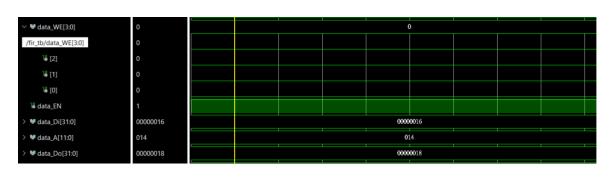
Data-in stream-in



Data-out stream-out

Name	Value	47,729,8 0 6 ps		47,729,808 ps		47,729,810 ps		47,729,812 ps	ı [⁴⁷
14 wvalid	1								
18 wready	1								
▶ ₩ wdata[31:0]	00000001	0000001							
1 arvalid	0								
arready	1								
▶ ▼ araddr[11:0]	02a	02a							
le rvalid	0								
₩ rready	1								
♥ rdata[31:0]	00000000				0000	0000			
↓ ss_tvalid	1								
ss_tready	0								
▶ ™ ss_tdata[31:0]	00000015				0000	0015			
₩ ss_tlast	0								
sm_tvalid	0								
1 sm_tready	1								
→ sm_tdata[31:0]	00001404	00001404							
sm_tlast	0								

RAM access control



∕ ▼ tap_wε(s.u)	V					,				
¹ 6 tap_EN	1									
> 🕶 tap_Di[31:0]	00000000				0000	0000				
> W tap_A[11:0]	00c				04)c				
> ₩ tap_Do[31:0]	ffffff7	mm7								
> ₩ data_WE[3:0]	0	0								
le data_EN	1									
> ♥ data_Di[31:0]	00000016	00000016								
> V data_A[11:0]	014	014								
> V data_Do[31:0]	00000018	00000018								

FSM

¹⊌ sm_tvalid	0									
↓ sm_tready	1									
> V sm_tdata[31:0]	00001404	00001404								
■ sm_tlast	0									
axis_clk	1									
<pre>axis_rst_n</pre>	1									
M . WE(2.0)	0				·					