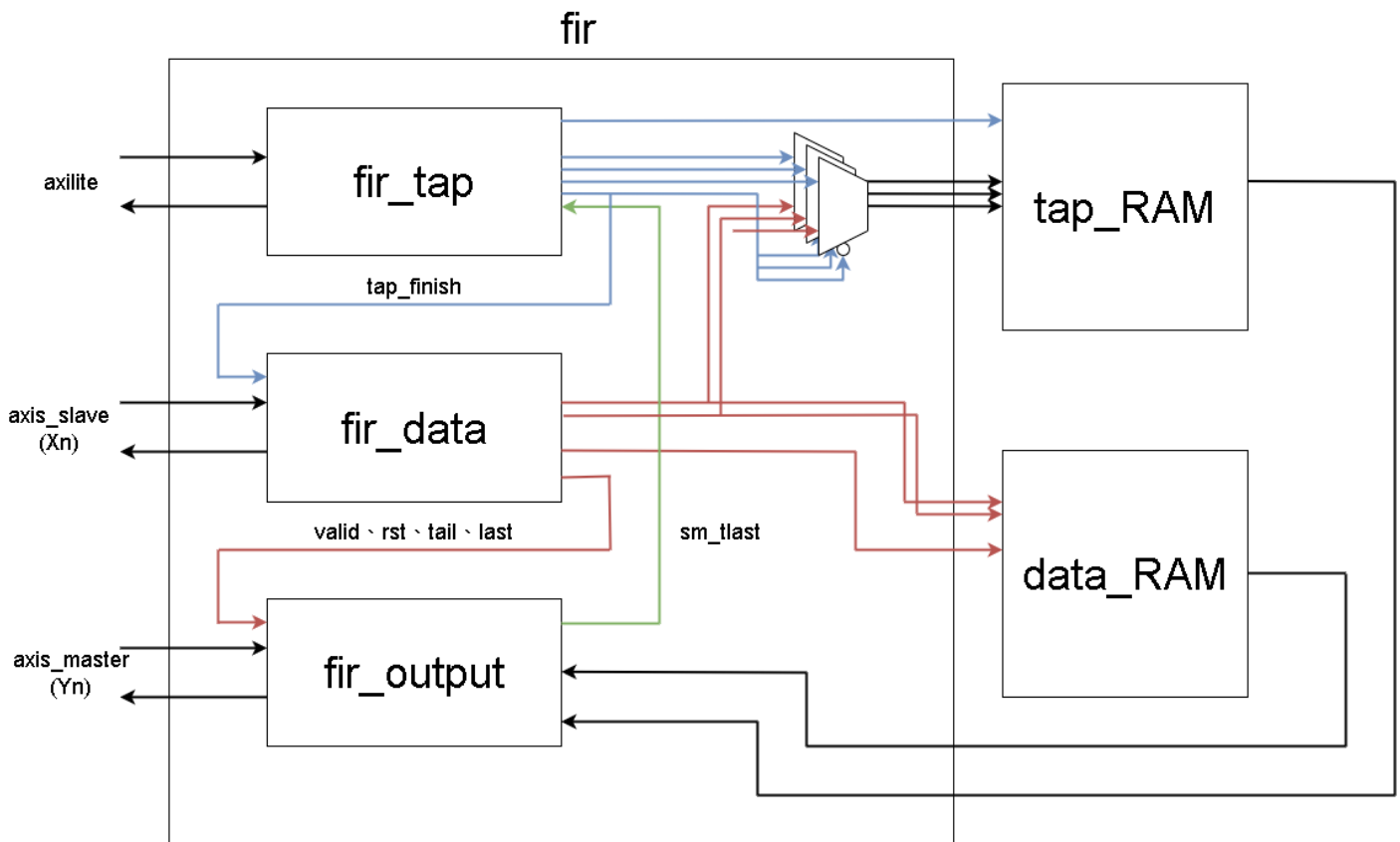


111061621 蔡以心

Block Diagram:

Datapath – dataflow:



Control signals:

tap_finish: **fir_tap** 的輸出，可以用來當 MUX 的控制訊號，**tap_finish** = 0，選擇 **fir_tap**，**tap_finish** = 1，選擇 **fir_data**。以及 **fir_data** 的啟動訊號，當 **tap_finish** = 1，從 **fir_data** 才會從 s0 跳到 s1。

valid、rst、tail、last: **fir_data** 的輸出，用來給 **fir_output** 計算 **tap_Do**、**data_Do** 所需的訊號。

sm_tlast: **fir_out** 的輸出，傳給 testbench 和 **fir_tap**，會讓 **fir_tap** 產生依序 **ap_done**、**ap_idle**。

How to receive tap parameters and place into SRAM:

testbench 在 write channel 使用 Axi-lite 傳送 awvalid、awaddr、wvalid、wdata，fir_tap 收到後有使用暫存器來儲存資料並回傳 awready、wready，收到 wready 後 testbench 才會繼續傳送下一筆資料。

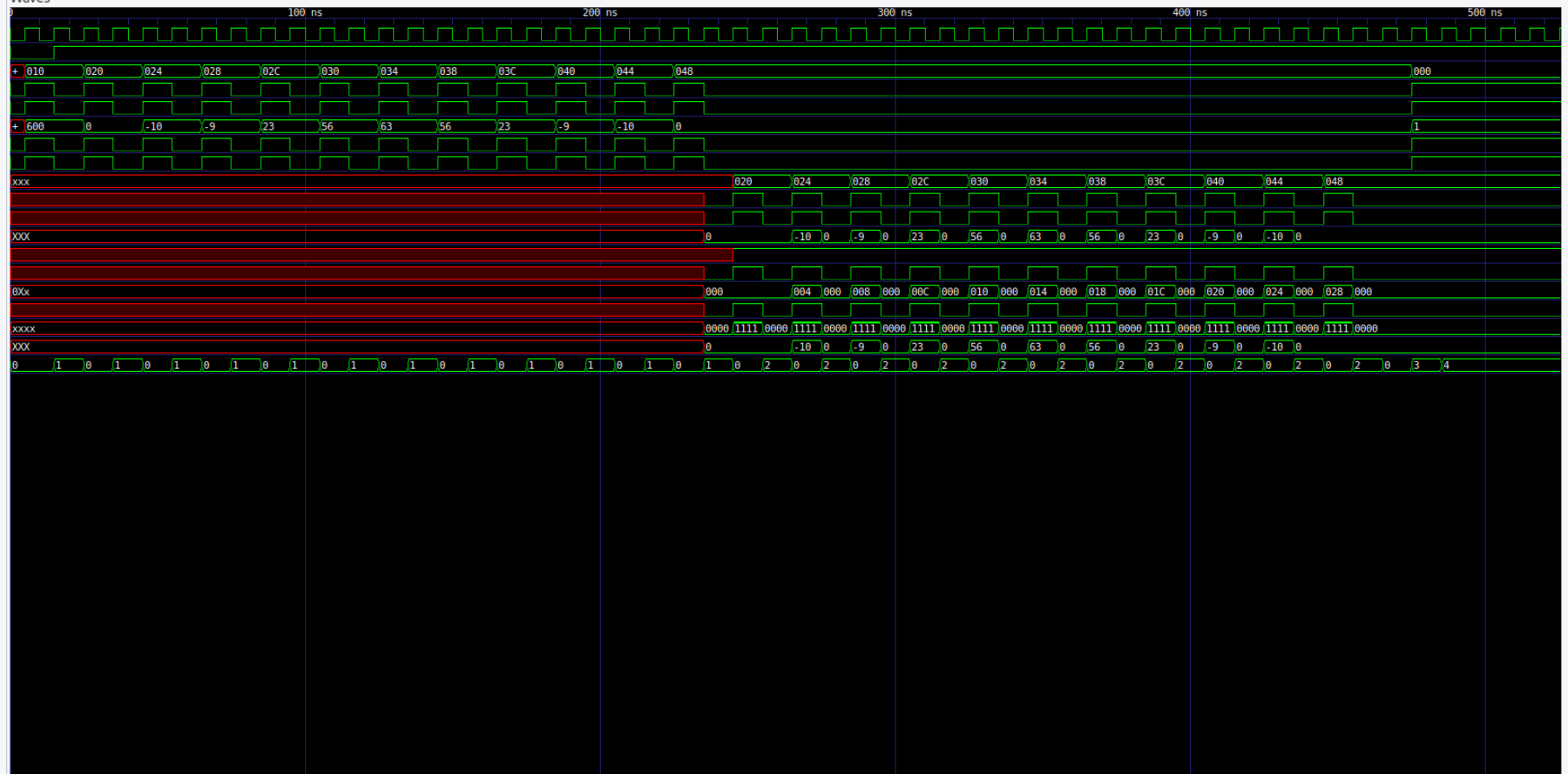
testbench 在 read channel 使用 Axi-lite 傳送 rready、arvalid、araddr，fir_tap 收到後把儲存在暫存器的資料輸出並回傳 arready、rvalid、rdata，收到 rvalid 後 testbench 才會繼續讀取下一筆資料。

寫資料的順序是先寫入 data-length，再寫入 11 個 tap parameter 給 fir_tap，當 testbench 寫完資料後會把 11 個 tap parameter 從 fir 讀取出來做確認，fir_tap 在讀取的同時會輸出 tapwe、tapen、tap_Di、tapaddr 給寫入 tap_RAM，確認完 11 個 tap parameter 時 tap_RAM 也完成寫入 11 個 tap parameter。

SST Signals

Time
axis_clk
axis_rst_n
awaddr[11:0]
awready
awvalid
wdata[31:0]
wready
wvalid
araddr[11:0]
arready
arvalid
rdata[31:0]
rready
rvalid
tapaddr[11:0]
tapen
tapwe[3:0]
tap_Di[31:0]
curr_state[2:0]

Waves




How to receive data-in and place into SRAM:

testbench 使用 Axi-stream 寫入 ss_tvalid、ss_tdata 給 fir_data，fir_data 收到後有使用 10 個 32bit 的暫存器來儲存輸入並回傳 ss_tready，testbench 收到 ss_tready 後才會繼續傳送下一筆資料。

當 tap_finish = 1，fir_data 的 fsm 才會啟動。依序將 fir_data 裡面的 mem[9]開始寫到 data_RAM 的 0x28，mem[8]寫到 data_RAM 的 0x24，直到 mem[0]寫到 data_RAM 的 0x04 後，將 fir_data 裡面的暫存器位移一位，開始讀取 ss_tdata 並儲存到 mem[0]，然後將 mem[0]寫到 data_RAM 的 0x00，這樣就完成 data_RAM 的寫入。

File Edit Search Time Markers View Help

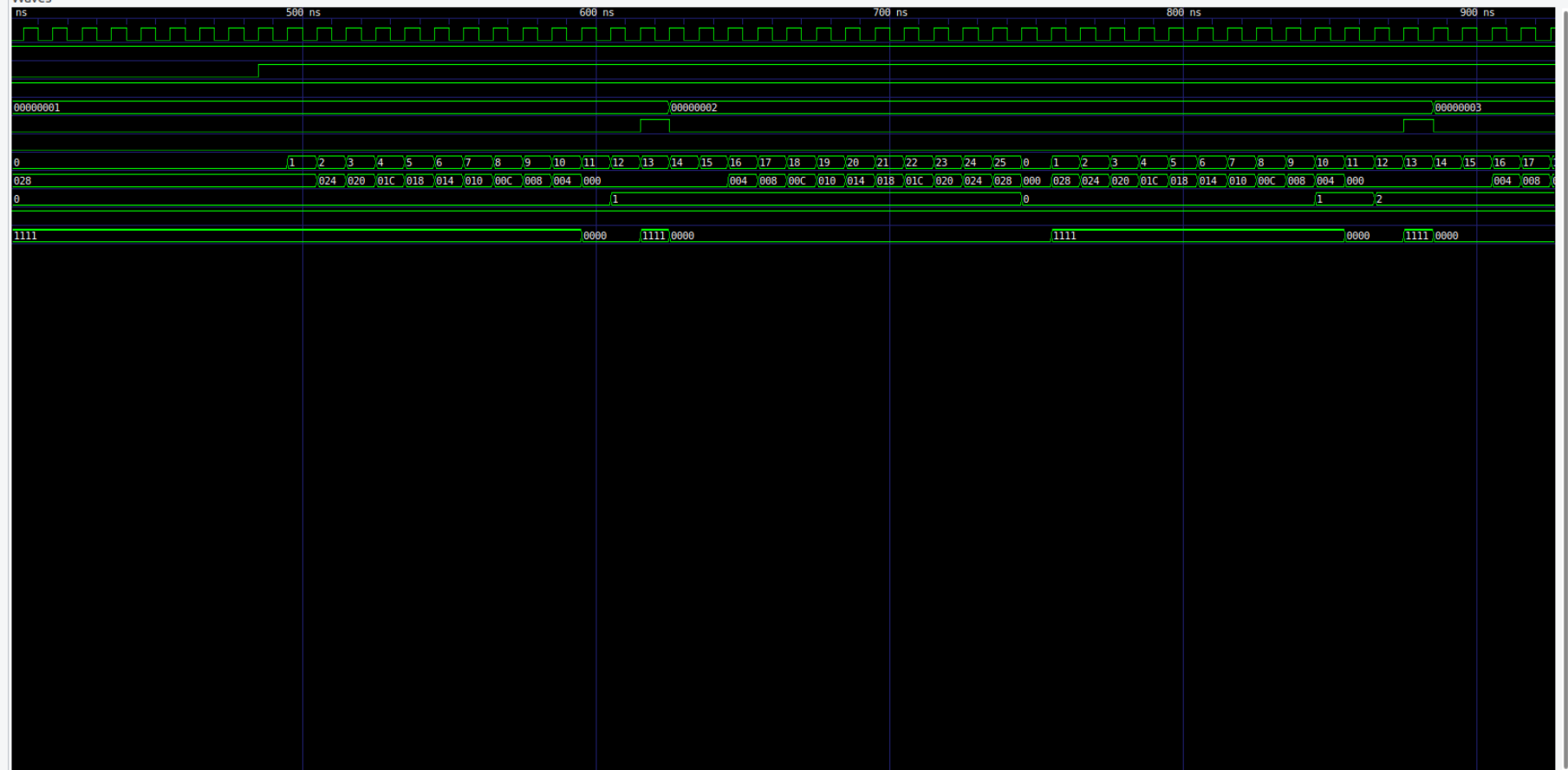
 From: 0 sec To: 156545 ns Marker: - | Cursor: 355800 ps

SST Signals

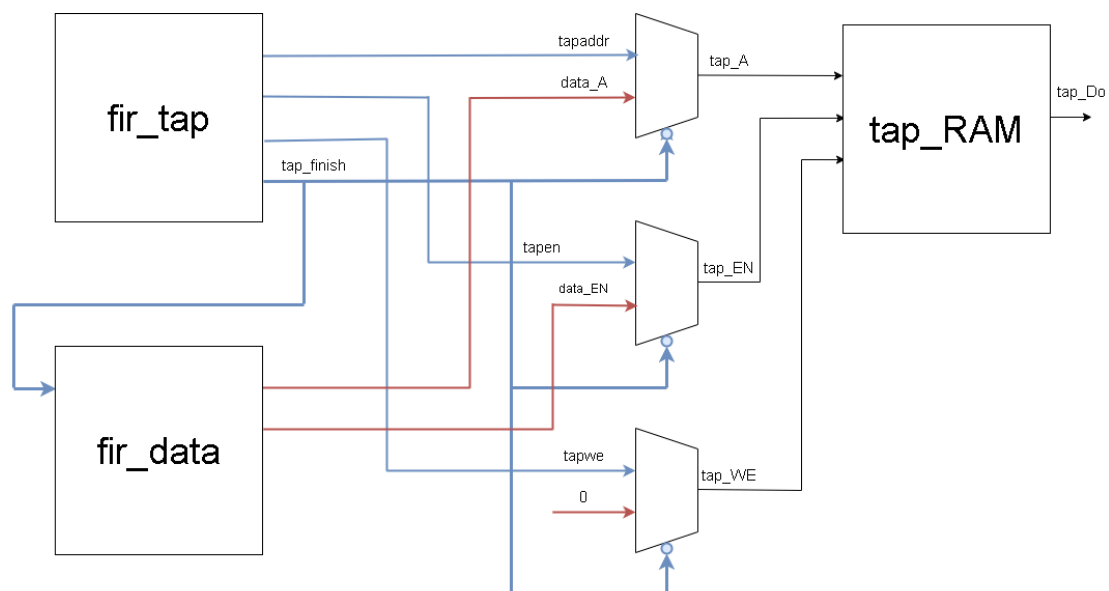
Time

- axis_clk
- axis_rst_n
- tap_finish
- ss_tvalid
- ss_tdata[31:0]
- ss_tready
- ss_tlast
- curr_state[4:0]
- data_A[11:0]
- data_Di[31:0]
- data_EN
- data_WE[3:0]

Waves



How to access shiftram and tapRAM to do computation:



當 **fir_data** 把 **data[0]**到 **data[10]**都寫入 **data_RAM** 時，會開始輸出 **data_WE**、**data_EN**、**data_A**，開始從 **data_RAM** 讀取 **data[0]**，並使用 MUX 和 **tap_finish** 作為 select 讓 **data_A** 跟 **data_WE** 一起控制 **tap_RAM** 讀取 **tap[0]**，可以讓 **tap_RAM**、**data_RAM** 同時一起輸出，方便 **fir_output** 去做控制。

File Edit Search Time Markers View Help

 From: 0 sec To: 156545 ns Marker: - | Cursor: 457900 ps

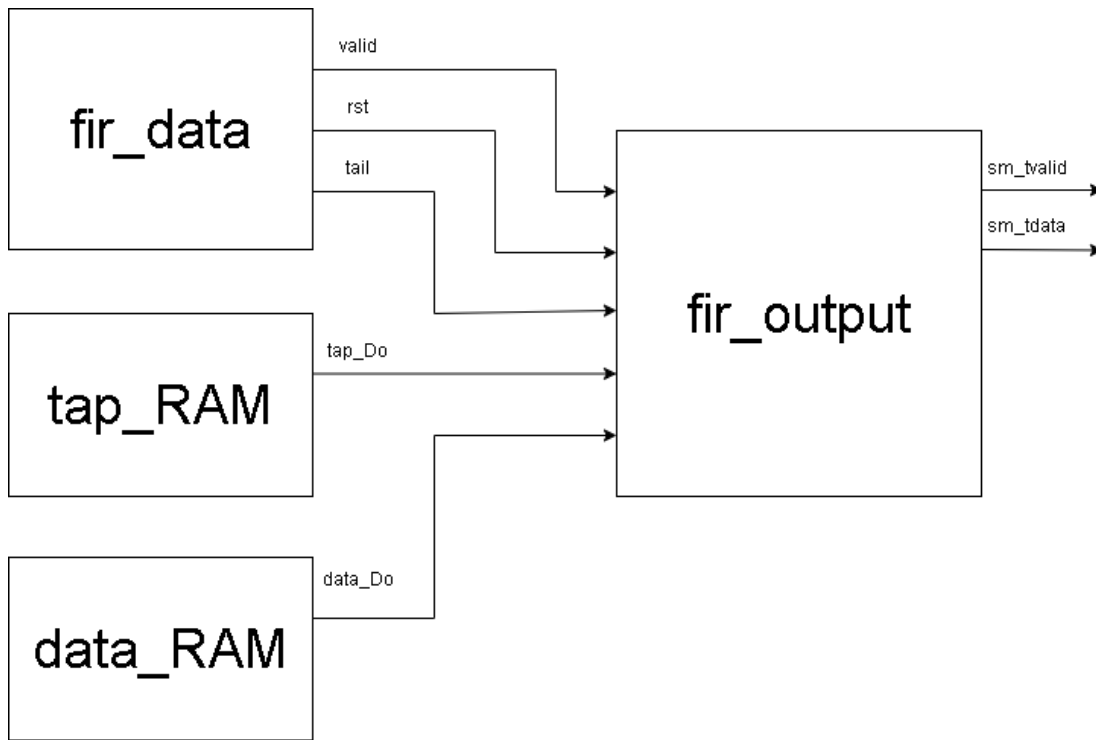
SST Signals

Time

- axis_clk
- axis_rst_n
- data_Di[31:0]
- data_A[11:0]
- data_EN
- data_WE[3:0]
- curr_state[4:0]
- tap_finish
- tapaddr[11:0]
- tapen
- tapwe[3:0]
- tap_A[11:0]
- tap_EN
- tap_WE[3:0]

Waves






rst: curr_state = s14 的輸出，只會拉起 1 個 cycle 且在 valid 的前 1 個 cycle 產生，讓 fir_output 把輸出歸零，每次計算時都會先歸零，避免產生錯誤。

valid: curr_state = s15~s25 的輸出，會拉起 11 個 cycle，讓 fir_output 接收到正確的 tap_Do 跟 data_Do。

tail: curr_state = s25 的輸出，只會跟最後一筆資料的 valid 同時拉起 1 個 cycle，讓 fir_output 產生 sm_valid，在接收每次計算的最後一筆資料會收到，讓 testbench 能收到正確的結果。

File Edit Search Time Markers View Help

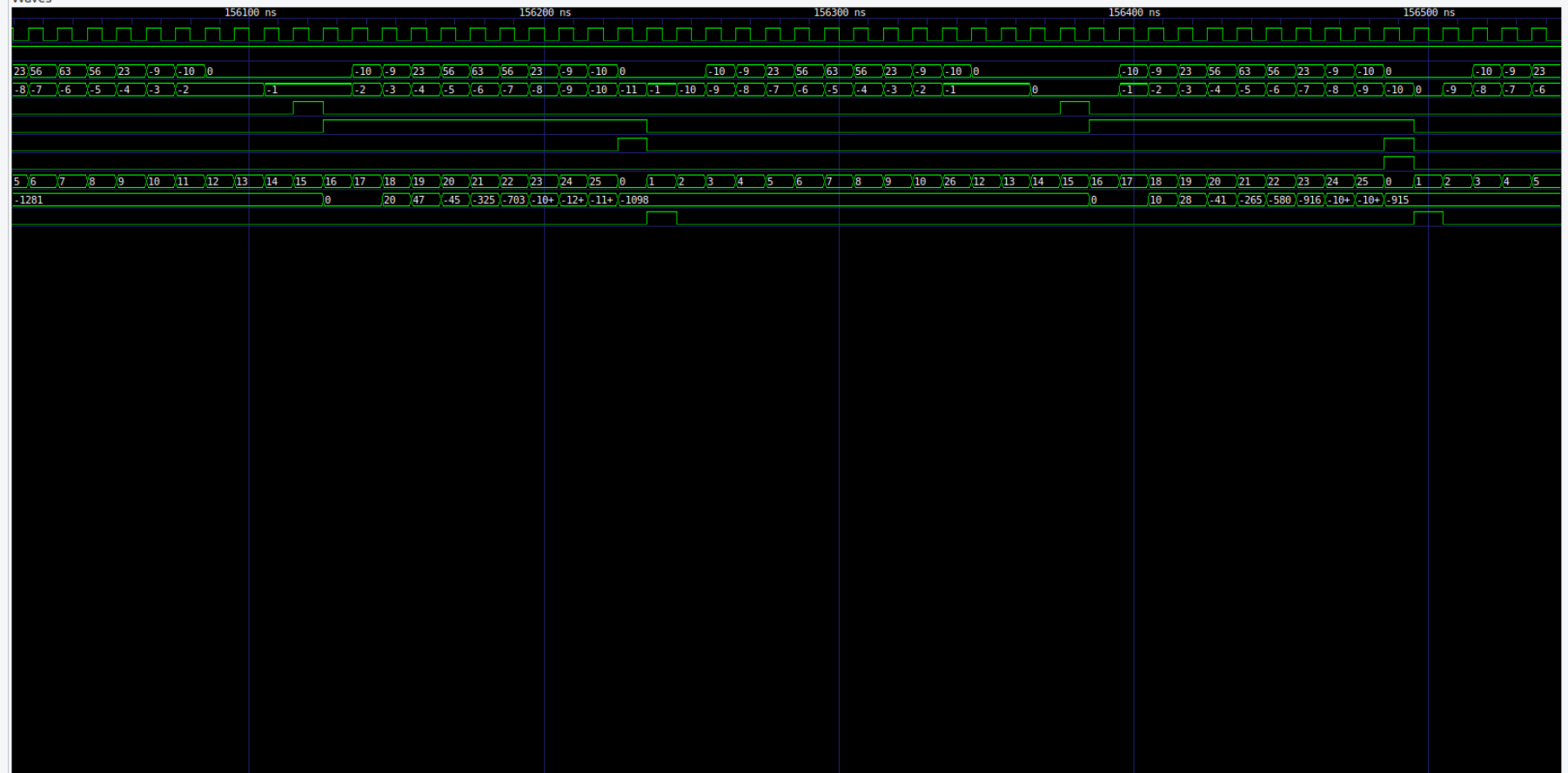
 From: 0 sec To: 156545 ns Marker: - | Cursor: 156124200 ps

SST Signals

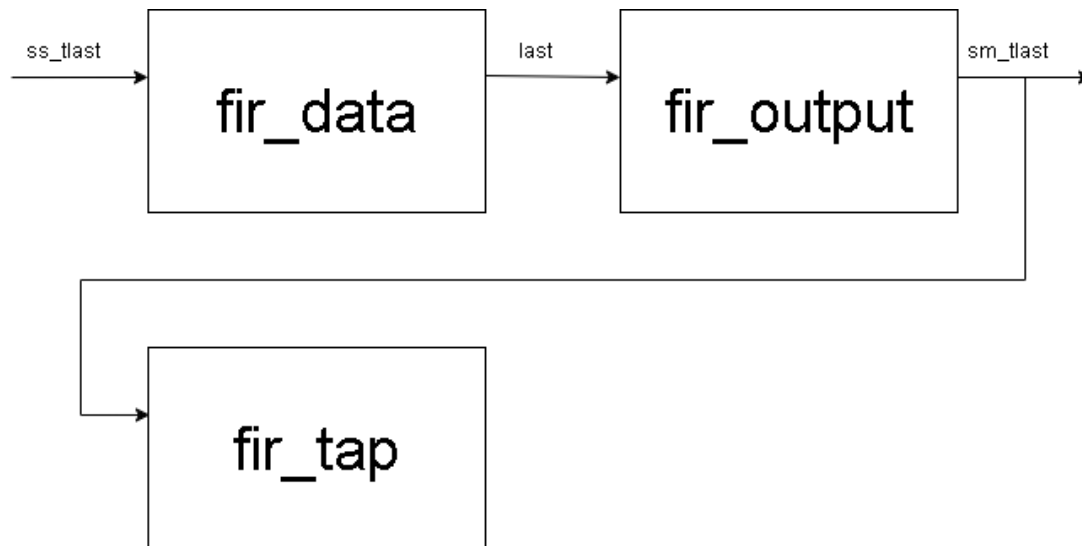
Time

- axis_clk
- axis_rst_n
- tap_Do[31:0]
- data_Do[31:0]
- rst
- valid
- tail
- last
- curr_state[4:0]
- sm_tdata[31:0]
- sm_tvalid

Waves




How ap_done& ap_idle is generated:



當 fir_data 接收到 ss_tlast = 1 時且 current_state = s10 時，則 next_state = s26，s26 的功能是 load 最後一筆 data 並會拉起 1 個 1bit 暫存器 datalast，然後跳回 s12，繼續運作，直到 s25 會拉起 1 個 1bit 暫存器 datatail，last 為 datalast 跟 datatail 做 and 運算的輸出，將 last 輸入到 fir_output。當 fir_output 收到 last = 1 會輸出 sm_tlast 給 Testbench 跟 fir_tap。當 fir_tap 收到 sm_tlast = 1 時，fir_tap 的 current_state 會從 s4 跳到 s5，s5 會將 Register[0x00]寫入 32'h2，代表 ap_done，下一個 cycle 跳到 s6，s6 會將 Register[0x00]寫入 32'h4，代表 ap_idle。

File Edit Search Time Markers View Help

 From: 0 sec To: 156545 ns Marker: - | Cursor: 156544300 ps

SST Signals

Time

- axis_clk
- axis_rst_n
- ss_tlast
- datalast
- dataail
- last
- curr_state[4:0]
- sm_tlast
- araddr[11:0]
- rdata[31:0]
- curr_state[2:0]

Waves



Resource usage: including FF, LUT, BRAM:

Open

fir_utilization_s

~/lab3_vivado/lab3_viva

28 1. Slice Logic

29 -----

30

31 +-----+-----+-----+-----+-----+-----+

32 | Site Type | Used | Fixed | Prohibited | Available | Util% |

33 +-----+-----+-----+-----+-----+-----+

34 | Slice LUTs* | 1835 | 0 | 0 | 53200 | 3.45 |

35 | LUT as Logic | 1835 | 0 | 0 | 53200 | 3.45 |

36 | LUT as Memory | 0 | 0 | 0 | 17400 | 0.00 |

37 | Slice Registers | 2860 | 0 | 0 | 106400 | 2.69 |

38 | Register as Flip Flop | 2860 | 0 | 0 | 106400 | 2.69 |

39 | Register as Latch | 0 | 0 | 0 | 106400 | 0.00 |

40 | F7 Muxes | 589 | 0 | 0 | 26600 | 2.21 |

41 | F8 Muxes | 256 | 0 | 0 | 13300 | 1.92 |

42 +-----+-----+-----+-----+-----+-----+

43 * Warning! The Final LUT count, after physical optimizations and full implementation, is typically low

44

45

46 1.1 Summary of Registers by Type

47 -----

48

49 +-----+-----+-----+-----+-----+

50 | Total | Clock Enable | Synchronous | Asynchronous |

51 +-----+-----+-----+-----+-----+

52 | 0 | - | - | - |

53 | 0 | - | - | Set |

54 | 0 | - | - | Reset |

55 | 0 | - | Set | - |

56 | 0 | - | Reset | - |

57 | 0 | Yes | - | - |

58 | 1 | Yes | - | Set |

59 | 392 | Yes | - | Reset |

60 | 0 | Yes | Set | - |

61 | 2467 | Yes | Reset | - |

62 +-----+-----+-----+-----+-----+

63

64

65 2. Memory

66 -----

67

68 +-----+-----+-----+-----+-----+

69 | Site Type | Used | Fixed | Prohibited | Available | Util% |

70 +-----+-----+-----+-----+-----+

71 | Block RAM Tile | 0 | 0 | 0 | 140 | 0.00 |

72 | RAMB36/FIFO* | 0 | 0 | 0 | 140 | 0.00 |

73 | RAMB18 | 0 | 0 | 0 | 280 | 0.00 |

74 +-----+-----+-----+-----+-----+

75 * Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one f

accommodate a RAMB18E1

Open			fir_vds
~/lab3_vivado/lab3_viva			
173 Start RTL Component Statistics			
174 -----			
175 Detailed RTL Component Info :			
176 +---Registers :			
177	32 Bit	Registers :=	85
178	12 Bit	Registers :=	1
179	4 Bit	Registers :=	1
180	1 Bit	Registers :=	9
181 +---Multipliers :			
182	32x32	Multipliers :=	1
183 +---Muxes :			
184	27 Input	32 Bit	Muxes := 1
185	2 Input	32 Bit	Muxes := 2
186	7 Input	32 Bit	Muxes := 1
187	75 Input	32 Bit	Muxes := 1
188	27 Input	27 Bit	Muxes := 1
189	2 Input	27 Bit	Muxes := 28
190	27 Input	12 Bit	Muxes := 1
191	2 Input	12 Bit	Muxes := 1
192	2 Input	6 Bit	Muxes := 1
193	27 Input	4 Bit	Muxes := 1
194	2 Input	4 Bit	Muxes := 1
195	7 Input	3 Bit	Muxes := 1
196	2 Input	3 Bit	Muxes := 3
197	27 Input	1 Bit	Muxes := 8
198	2 Input	1 Bit	Muxes := 8
199	3 Input	1 Bit	Muxes := 74
200	7 Input	1 Bit	Muxes := 3
201 -----			
202 Finished RTL Component Statistics			
203 -----			
204			
205 Start Part Resource Summary			
206 -----			
207 Part Resources:			
208 DSPs: 220 (col length:60)			
209 BRAMs: 280 (col length: RAMB18 60 RAMB36 30)			
210 -----			
211 Finished Part Resource Summary			
212 -----			
213			
214 Start Cross Boundary and Area Optimization			
215 -----			
216 WARNING: [Synth 8-7080] Parallel synthesis criteria is not met			
217 DSP Report: Generating DSP o1/outputdata1, operation Mode is: A*B.			
218 DSP Report: operator o1/outputdata1 is absorbed into DSP o1/outputdata1.			
219 DSP Report: operator o1/outputdata1 is absorbed into DSP o1/outputdata1.			
220 DSP Report: Generating DSP o1/outputdata1, operation Mode is: (PCIN>>17)+A*B.			
221 DSP Report: operator o1/outputdata1 is absorbed into DSP o1/outputdata1			

Timing Report:

Try to synthesize the design with maximum frequency & Report timing on longest path, slack

Open

timing_report.txt

~/lab3_vivado

503

504 | Timing Details

505 | -----

506

507

508

509

510 From Clock: axis_clk

511 To Clock: axis_clk

512

513 Setup : 0 Failing Endpoints, Worst Slack 0.001ns, Total Violation 0.000ns

514 Hold : 0 Failing Endpoints, Worst Slack 0.137ns, Total Violation 0.000ns

515 PW : 0 Failing Endpoints, Worst Slack 1.420ns, Total Violation 0.000ns

516

517

518

519 Max Delay Paths

520

521 Slack (MET) : 0.001ns (required time - arrival time)

522 Source: t1/mem_reg[55][0]/C

523 (rising edge-triggered cell FDRE clocked by axis_clk {rise@0.000ns fall@1.920ns period=3.840ns})

524 Destination: t1/mem_reg[0][0]_0/D

525 (rising edge-triggered cell FDRE clocked by axis_clk {rise@0.000ns fall@1.920ns period=3.840ns})

526 Path Group: axis_clk

527 Path Type: Setup (Max at Slow Process Corner)

528 Requirement: 3.840ns (axis_clk rise@3.840ns - axis_clk rise@0.000ns)

529 Data Path Delay: 3.703ns (Logic 1.561ns (42.155%) route 2.142ns (57.845%))

530 Logic Levels: 5 (LUT5=1 LUT6=2 MUXF7=1 MUXF8=1)

531 Clock Path Skew: -0.145ns (DCD - SCD + CPR)

532 Destination Clock Delay (DCD): 2.128ns = (5.968 - 3.840)

533 Source Clock Delay (SCD): 2.456ns

534 Clock Pessimism Removal (CPR): 0.184ns

535 Clock Uncertainty: 0.035ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE

536 Total System Jitter (TSJ): 0.071ns

537 Total Input Jitter (TIJ): 0.000ns

538 Discrete Jitter (DJ): 0.000ns

539 Phase Error (PE): 0.000ns

540

541 Location Delay type Incr(ns) Path(ns) Netlist Resource(s)

542 -----

543 (clock axis_clk rise edge)

544 0.000 0.000 r

545 0.000 0.000 r axis_clk (IN)

546 net (fo=0) 0.000 0.000 axis_clk

547 r axis_clk_IBUF_inst/I

548 IBUF (Prop_ibuf_I_0) 0.972 0.972 r axis_clk_IBUF_inst/O

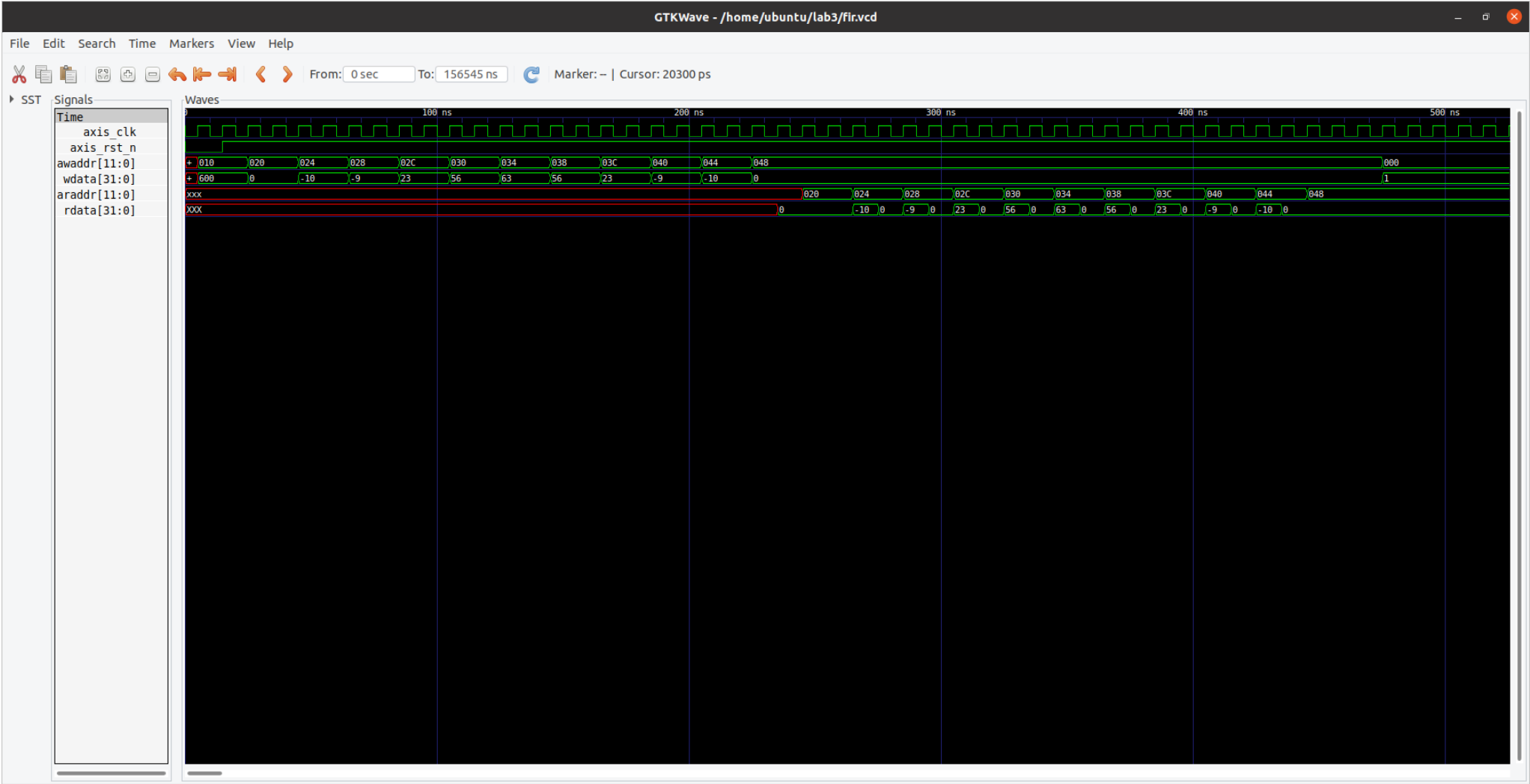
549 net (fo=1, unplaced) 0.800 1.771 axis_clk_IBUF

550 r axis_clk_IBUF_BUFG_inst/I

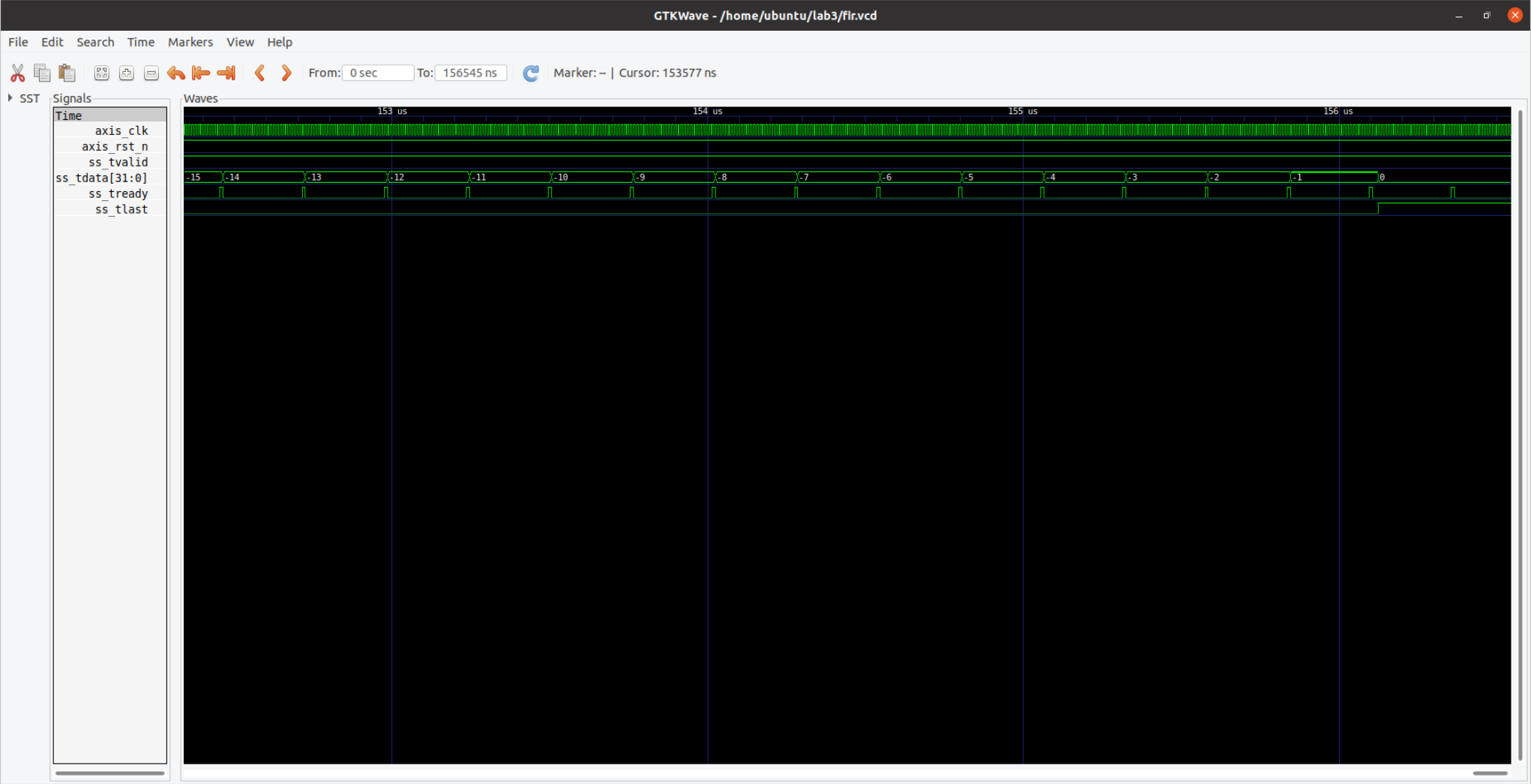
clock period=3.84ns

timing_report.txt					
~/lab3_vivado					
547	-----				
548	IBUF (Prop_ibuf_I_0)	0.972	0.972	r	axis_clk_IBUF_inst/I
549	net (fo=1, unplaced)	0.800	1.771	r	axis_clk_IBUF_inst/O
550				r	axis_clk_IBUF
551	BUFG (Prop_bufg_I_0)	0.101	1.872	r	axis_clk_IBUF_BUFG_inst/I
552	net (fo=2860, unplaced)	0.584	2.456	r	axis_clk_IBUF_BUFG_inst/O
553	FDRE			r	t1/mem_reg[55][0]/C
554	-----				
555	FDRE (Prop_fdre_C_Q)	0.478	2.934	r	t1/mem_reg[55][0]/Q
556	net (fo=2, unplaced)	0.976	3.910	r	t1/mem_reg_n_0[55][0]
557				r	t1/mem[0][0]_0_i_20/I0
558	LUT6 (Prop_lut6_I0_0)	0.295	4.205	r	t1/mem[0][0]_0_i_20/O
559	net (fo=1, unplaced)	0.000	4.205	r	t1/mem[0][0]_0_i_20_n_0
560				r	t1/mem_reg[0][0]_0_i_11/I1
561	MUXF7 (Prop_muxf7_I1_0)	0.247	4.452	r	t1/mem_reg[0][0]_0_i_11/O
562	net (fo=1, unplaced)	0.000	4.452	r	t1/mem_reg[0][0]_0_i_11_n_0
563				r	t1/mem_reg[0][0]_0_i_7/I0
564	MUXF8 (Prop_muxf8_I0_0)	0.098	4.550	r	t1/mem_reg[0][0]_0_i_7/O
565	net (fo=1, unplaced)	0.717	5.267	r	t1/mem_reg[0][0]_0_i_7_n_0
566				r	t1/mem[0][0]_0_i_3/I0
567	LUT6 (Prop_lut6_I0_0)	0.319	5.586	r	t1/mem[0][0]_0_i_3/O
568	net (fo=1, unplaced)	0.449	6.035	r	t1/mem[0][0]_0_i_3_n_0
569				r	t1/mem[0][0]_0_i_1/I3
570	LUT5 (Prop_lut5_I3_0)	0.124	6.159	r	t1/mem[0][0]_0_i_1/O
571	net (fo=75, unplaced)	0.000	6.159	r	t1/mem[0][0]_0_i_1_n_0
572	FDRE			r	t1/mem_reg[0][0]_0/D
573	-----				
574	(clock axis_clk rise edge)				
575		3.840	3.840	r	
576		0.000	3.840	r	axis_clk (IN)
577	net (fo=0)	0.000	3.840	r	axis_clk
578				r	axis_clk_IBUF_inst/I
579	IBUF (Prop_ibuf_I_0)	0.838	4.678	r	axis_clk_IBUF_inst/O
580	net (fo=1, unplaced)	0.760	5.438	r	axis_clk_IBUF
581				r	axis_clk_IBUF_BUFG_inst/I
582	BUFG (Prop_bufg_I_0)	0.091	5.529	r	axis_clk_IBUF_BUFG_inst/O
583	net (fo=2860, unplaced)	0.439	5.968	r	t1/CLK
584	FDRE			r	t1/mem_reg[0][0]_0/C
585	clock pessimism	0.184	6.151		
586	clock uncertainty	-0.035	6.116		
587	FDRE (Setup_fdre_C_D)	0.044	6.160		t1/mem_reg[0][0]_0
588	-----				
589	required time		6.160		
590	arrival time		-6.159		
591	-----				
592	slack		0.001		
593	-----				
594	-----				

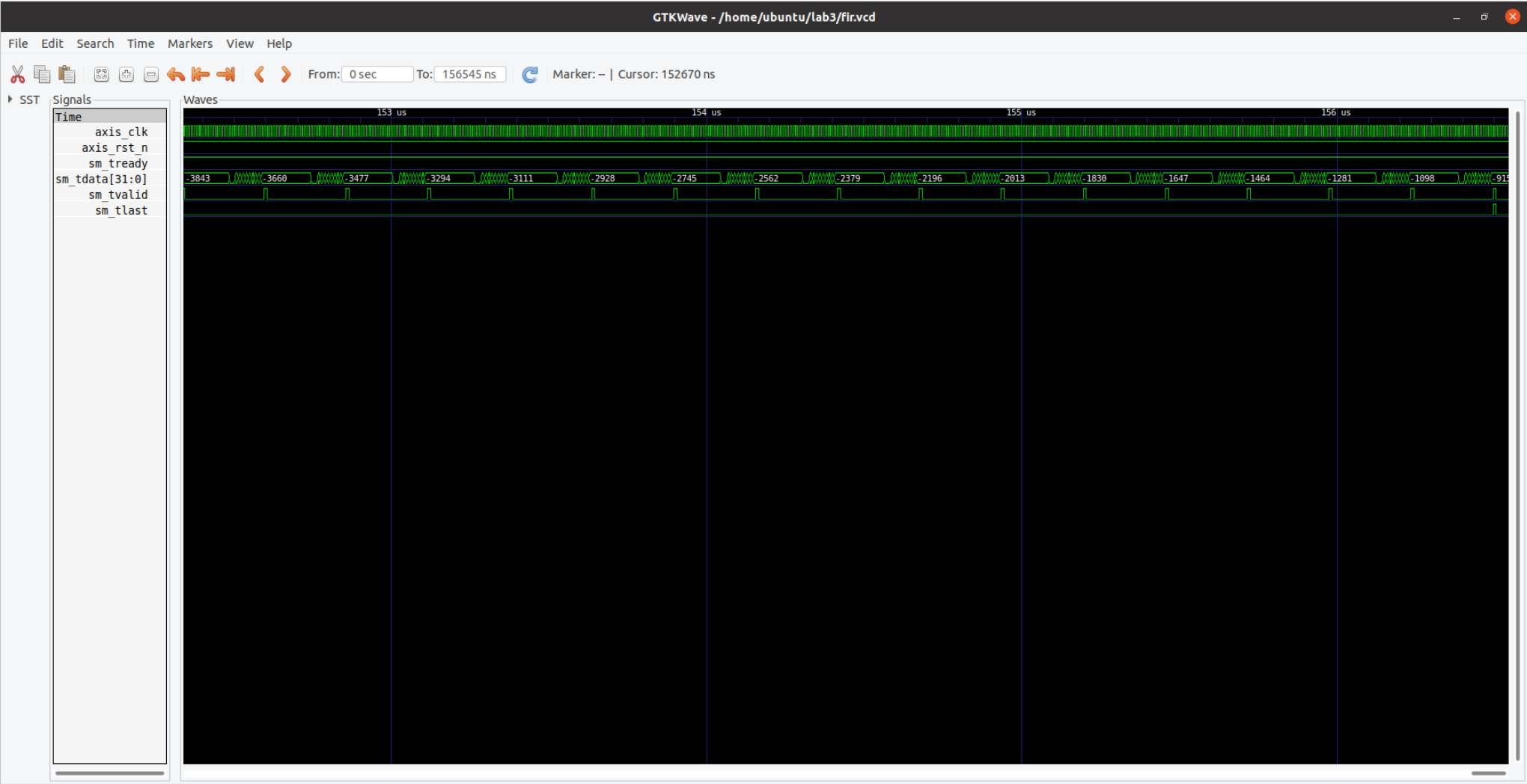
Coefficient program, and read back



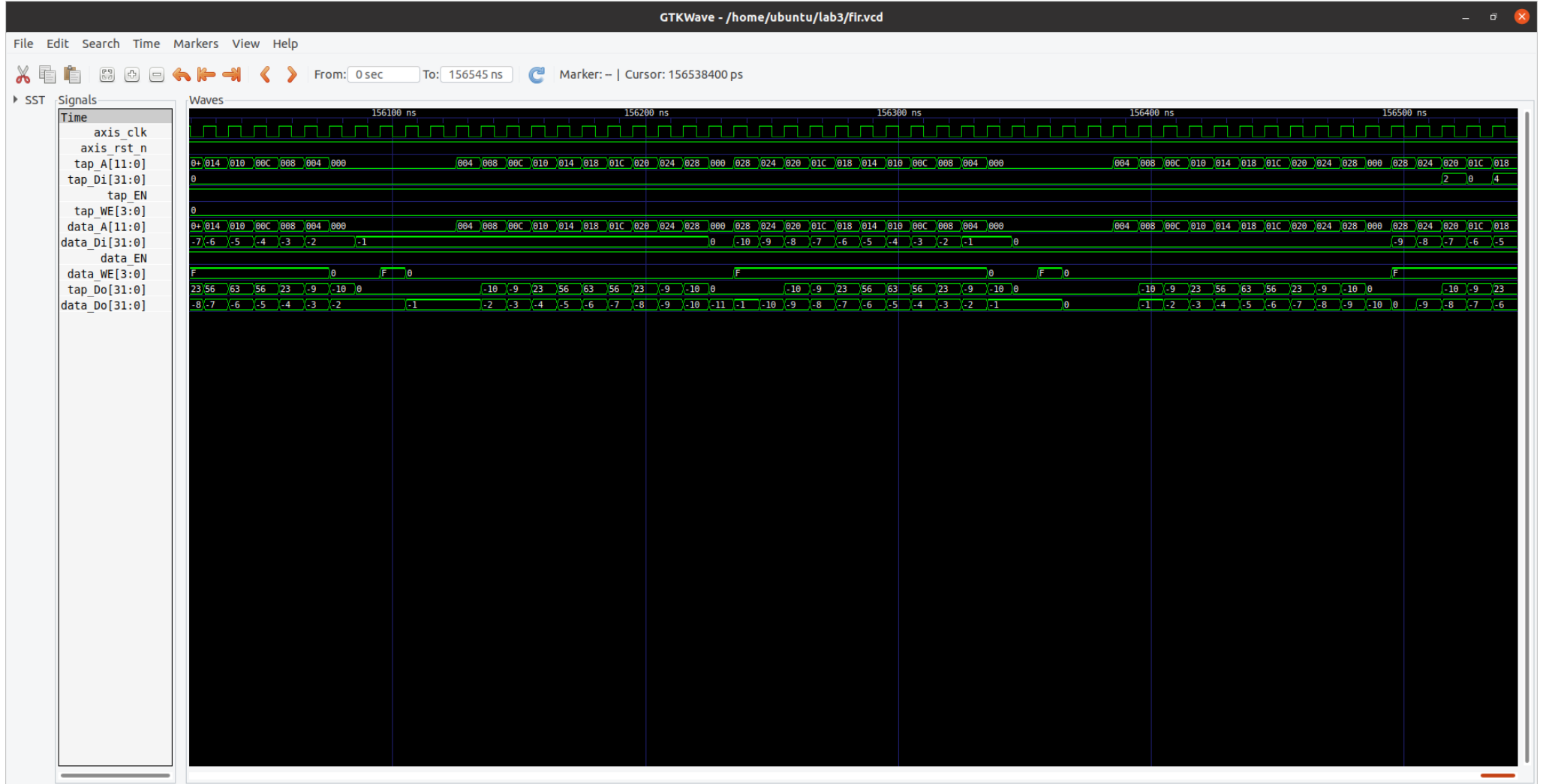
Data-in stream-in



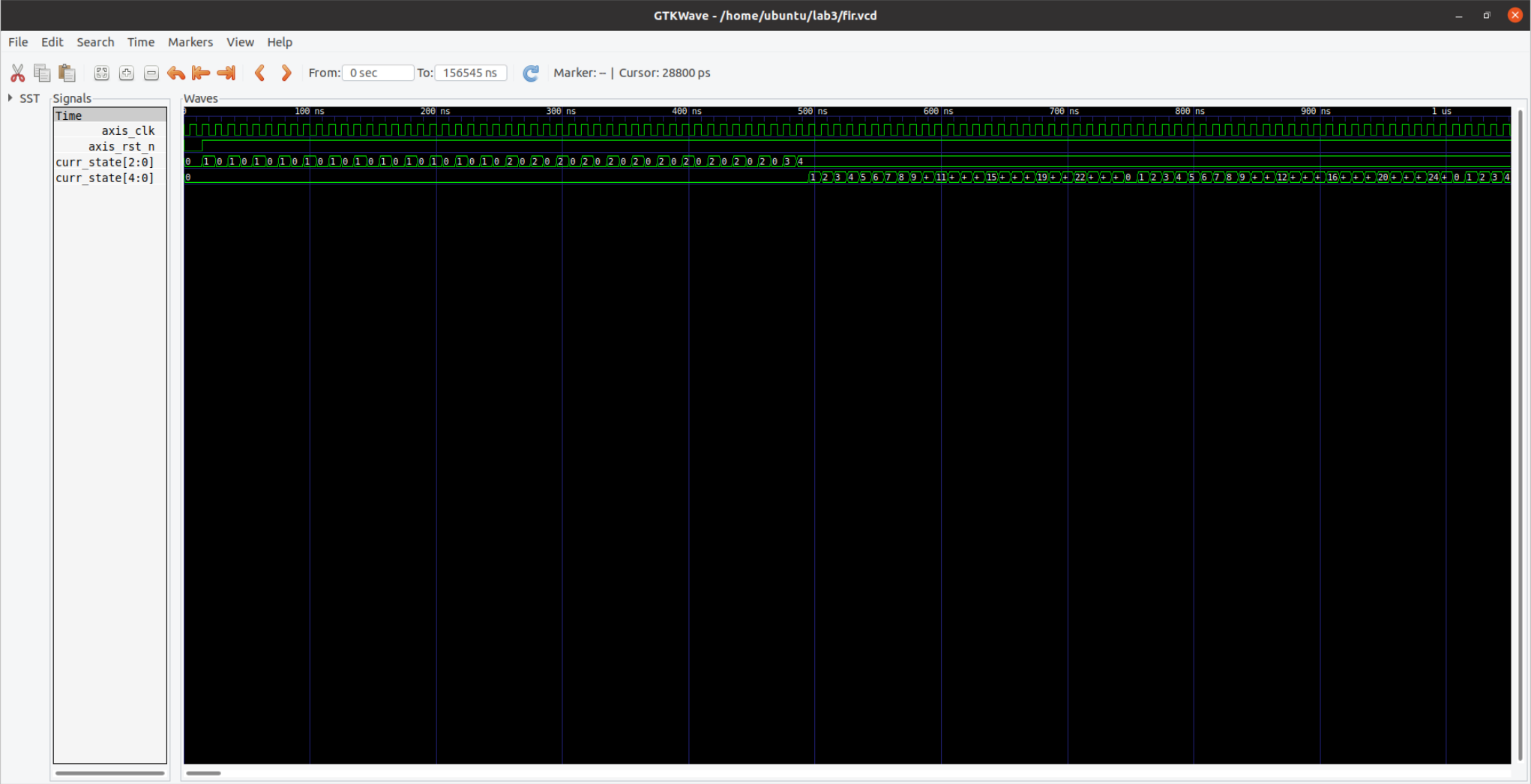
Data-out stream-out



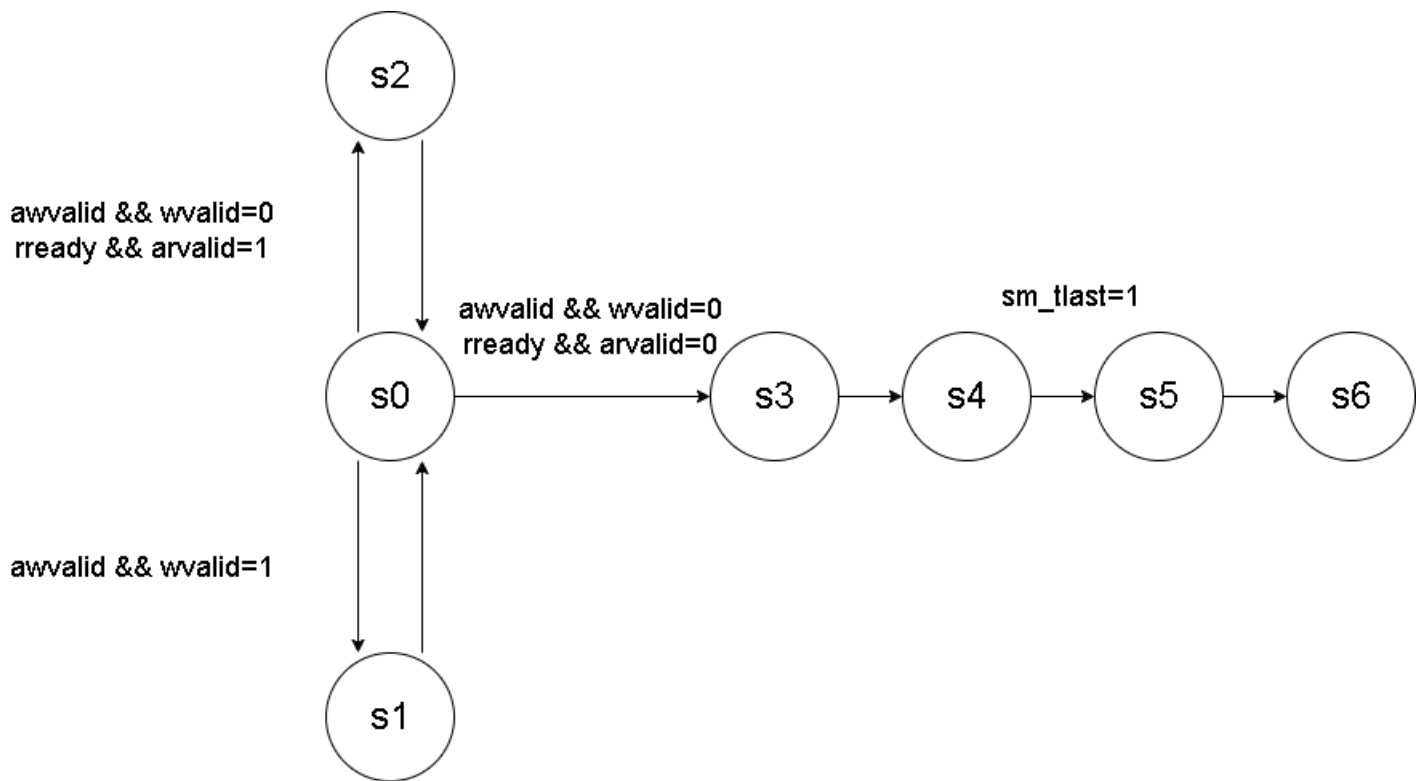
RAM access control



FSM



fir_tap:



s0:等待狀態，根據 `awvalid`、`wvalid`，`rready`、`arvalid` 決定下一個狀態。

s1:testbench 寫入 `fir_tap`，進行 write，下一個狀態會跳回 s0。

s2:testbench 讀取 `fir_tap`，進行 read 並輸出 `we`，同時寫入 `tap_RAM`，下一個狀態會跳回 s0。

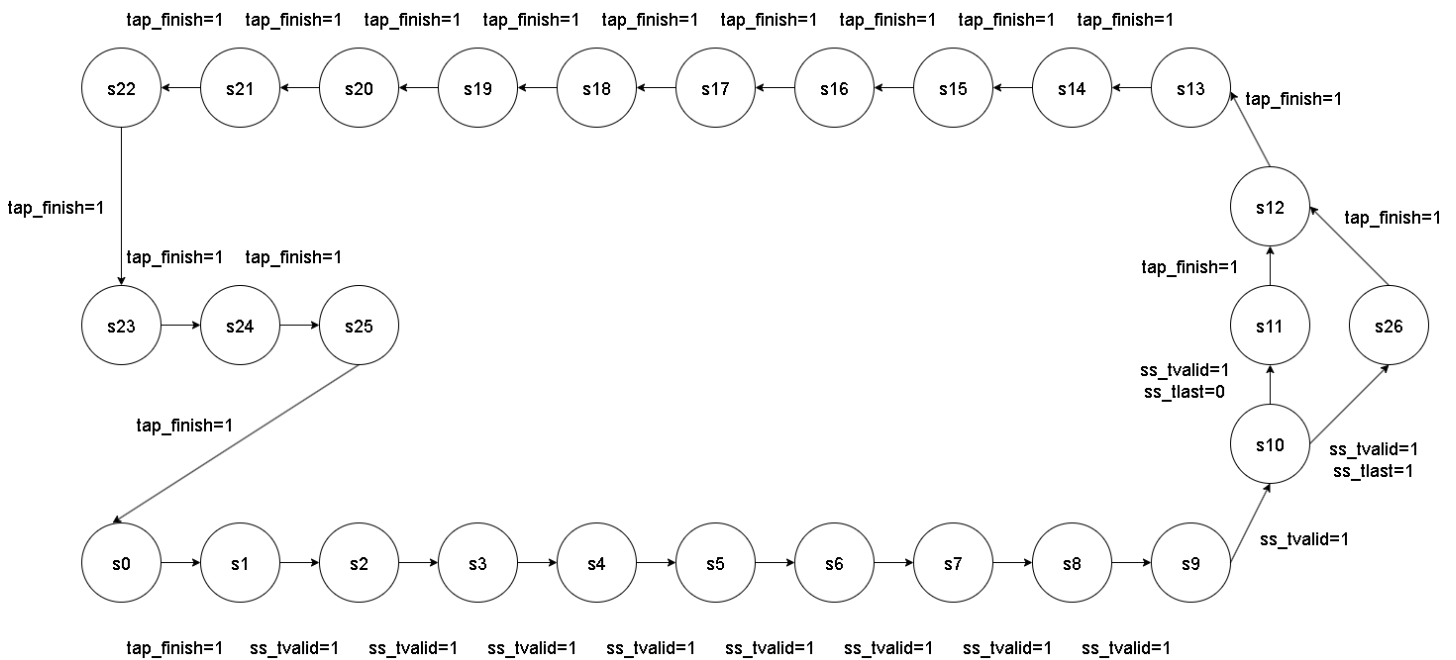
s3:輸出 `tap_finish`，下一個狀態會跳到 s4。

s4:將內部暫存器位址 0x00 的值歸零，即 `ap_start reset`，當 `fir_tap` 接收到 `sm_tlast` 下一個狀態會跳到 s5。

s5:將內部暫存器位址 0x00 的值改為 32'h2，即代表 `ap_done`，一個狀態會跳到 s6。

s6:將內部暫存器位址 0x00 的值改為 32'h4，即代表 `ap_idle`。

fir_data:



fir_data 裡面有 10 個 32bit 的暫存器 mem。

s0: write mem[9]，fir_data 寫入 data_RAM[10]，接收到 tap_finish 才會跳到下一個狀態 s1。

s1: write mem[8]，fir_data 寫入 data_RAM[9]，當 ss_tvalid=1，下一個狀態會跳到 s2。

s2: write mem[7]，fir_data 寫入 data_RAM[8]，當 ss_tvalid=1，下一個狀態會跳到 s3。

s3: write mem[6]，fir_data 寫入 data_RAM[7]，當 ss_tvalid=1，下一個狀態會跳到 s4。

s4: write mem[5]，fir_data 寫入 data_RAM[6]，當 ss_tvalid=1，下一個狀態會跳到 s5。

s5: write mem[4]，fir_data 寫入 data_RAM[5]，當 ss_tvalid=1，下一個狀態會跳到 s6。

s6: write mem[3]，fir_data 寫入 data_RAM[4]，當 ss_tvalid=1，下一個狀態會跳到 s7。

s7: write mem[2]，fir_data 寫入 data_RAM[3]，當 ss_tvalid=1，下一個狀態會跳到 s8。

s8: write mem[1]，fir_data 寫入 data_RAM[2]，當 ss_tvalid=1，下一個狀態會跳到 s9。

s9: write mem[0]，fir_data 寫入 data_RAM[1]，當 ss_tvalid=1，下一個狀態會跳到 s10。

s10:將暫存器進行位移，當 ss_tvalid=1、sstlast=0，下一個狀態會跳到 s11，

當 ss_tvalid=1、sstlast=1，下一個狀態會跳到 s26。

s11: load ss_tdata，fir_data 讀取 testbench 給的輸入，當 ss_tvalid=1，下一個狀態會跳到 s12。

s12: write mem[0]，fir_data 寫入 data_RAM[0]，當 tap_finish=1，下一個狀態會跳到 s13。

s13: wait，當 tap_finish=1，下一個狀態會跳到 s14。

s14: read mem[0] , fir_data 讀取 data_RAM[0] , 當 tap_finish=1 , 下一個狀態會跳到 s15 。
s15: read mem[1] , fir_data 讀取 data_RAM[1] , 當 tap_finish=1 , 下一個狀態會跳到 s16 。
s16: read mem[2] , fir_data 讀取 data_RAM[2] , 當 tap_finish=1 , 下一個狀態會跳到 s17 。
s17: read mem[3] , fir_data 讀取 data_RAM[3] , 當 tap_finish=1 , 下一個狀態會跳到 s18 。
s18: read mem[4] , fir_data 讀取 data_RAM[4] , 當 tap_finish=1 , 下一個狀態會跳到 s19 。
s19: read mem[5] , fir_data 讀取 data_RAM[5] , 當 tap_finish=1 , 下一個狀態會跳到 s20 。
s20: read mem[6] , fir_data 讀取 data_RAM[6] , 當 tap_finish=1 , 下一個狀態會跳到 s21 。
s21: read mem[7] , fir_data 讀取 data_RAM[7] , 當 tap_finish=1 , 下一個狀態會跳到 s22 。
s22: read mem[8] , fir_data 讀取 data_RAM[8] , 當 tap_finish=1 , 下一個狀態會跳到 s23 。
s23: read mem[9] , fir_data 讀取 data_RAM[9] , 當 tap_finish=1 , 下一個狀態會跳到 s24 。
s24: read mem[10] , fir_data 讀取 data_RAM[10] , 當 tap_finish=1 , 下一個狀態會跳到 s25 。
s25: wait , 當 tap_finish=1 , 下一個狀態會跳到 s25 。
s26: last load ss_tdata , fir_data 讀取最後一筆 testbench 給的輸入 , 當 tap_finish=1 , 下一個狀態會跳到 s12 。

github:

<https://github.com/yihsintsai1003/lab3>