Single Cycle Processor

Project 2

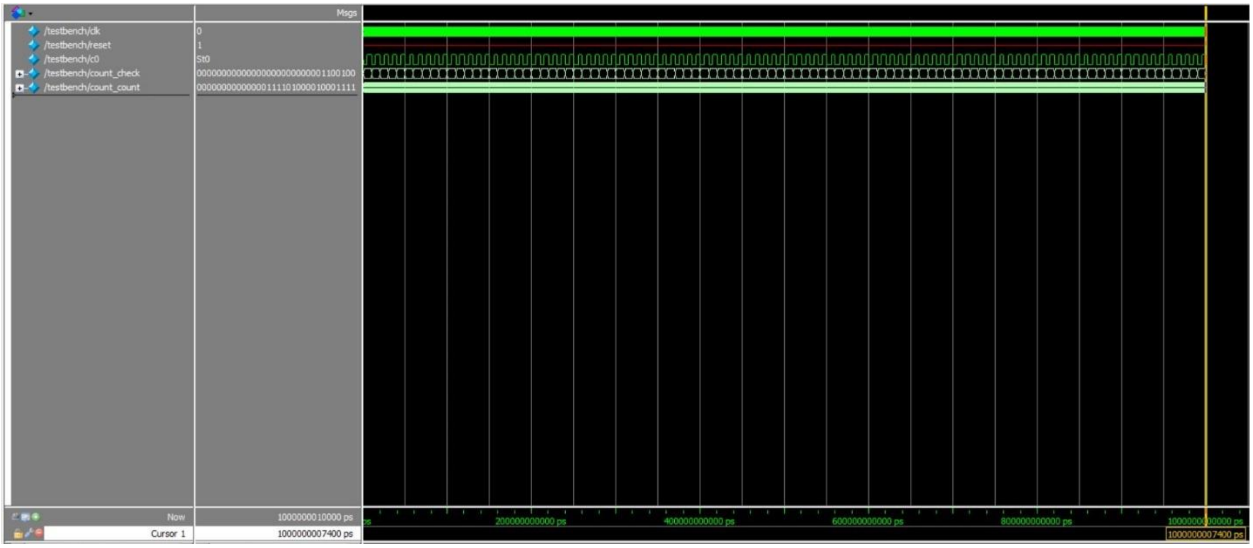
Chenkai Shao

Jeongsoo Kim

Waveforms & Testbenches

Single Modules

ClockDivider.v



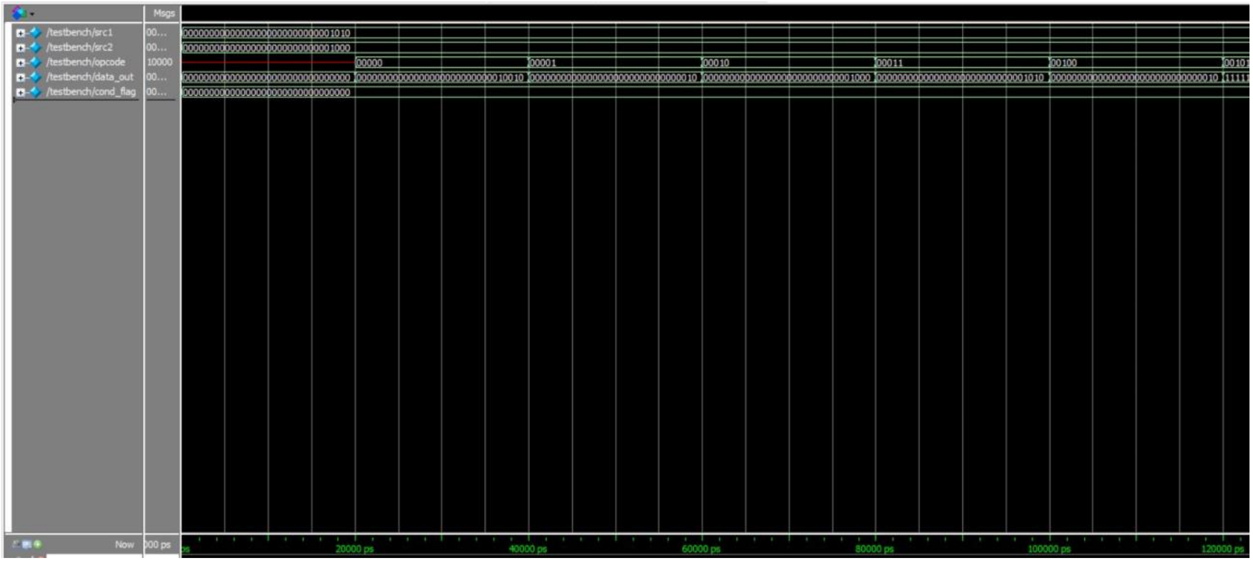
```
/testbench/clk            0
/testbench/reset          1
/testbench/c0             St0
/testbench/count_check    00000000000000000000000001100100
/testbench/count_count    00000000000000011110100001000111 1
Now                       1000000010000 ps
Cursor 1                  1000000007400 ps
```

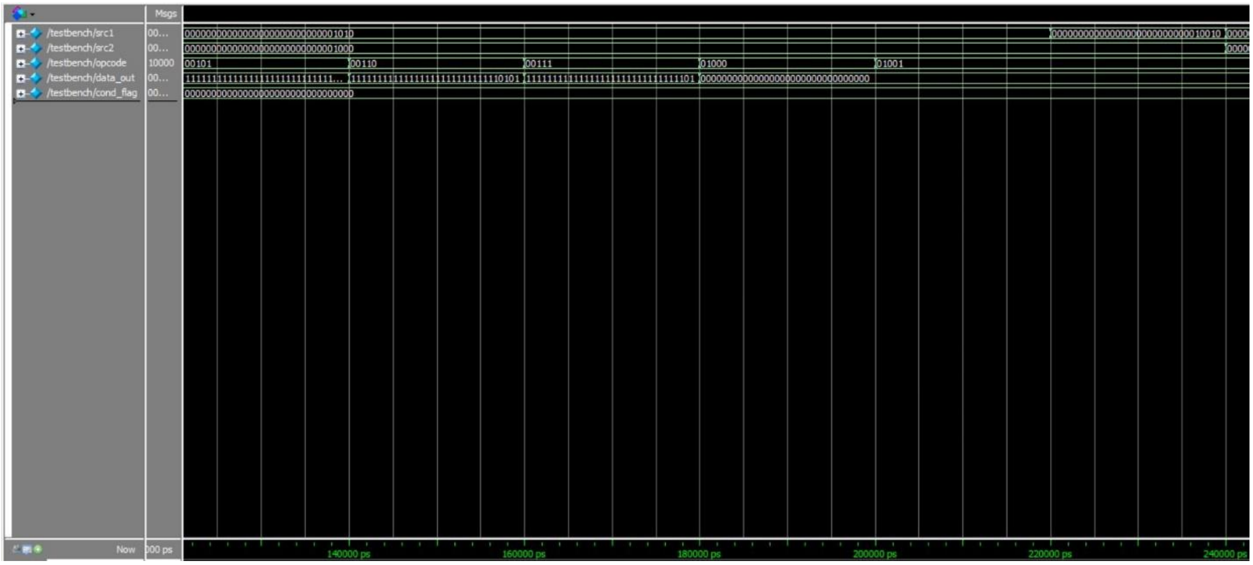
Description

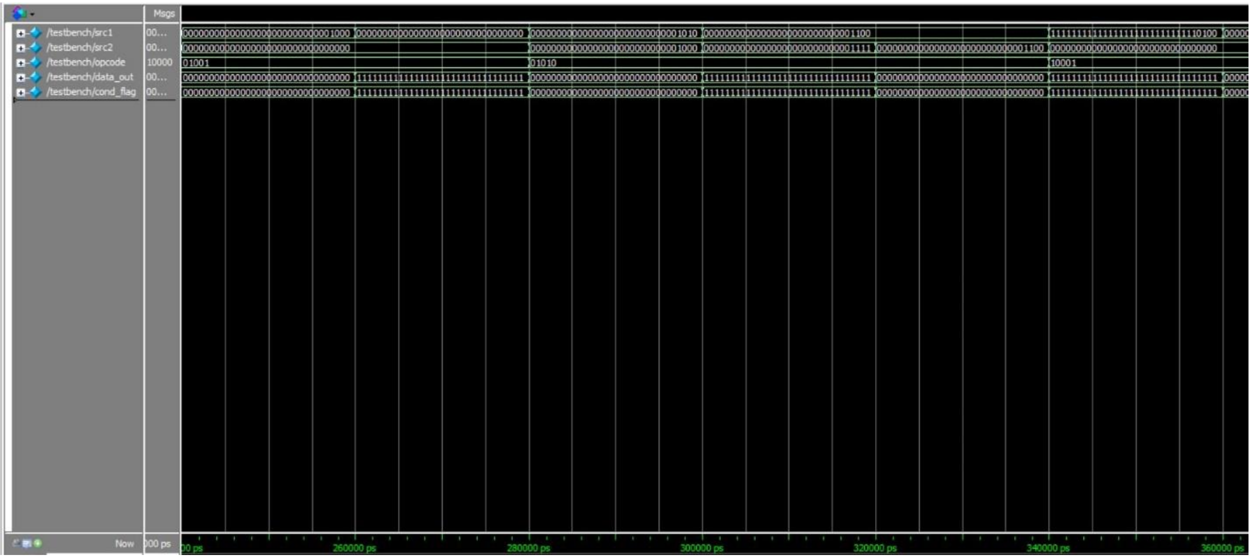
Since, it is a little hard to check what happens based on the image of the waveform of the ClockDivider. We add a brief description to explain.

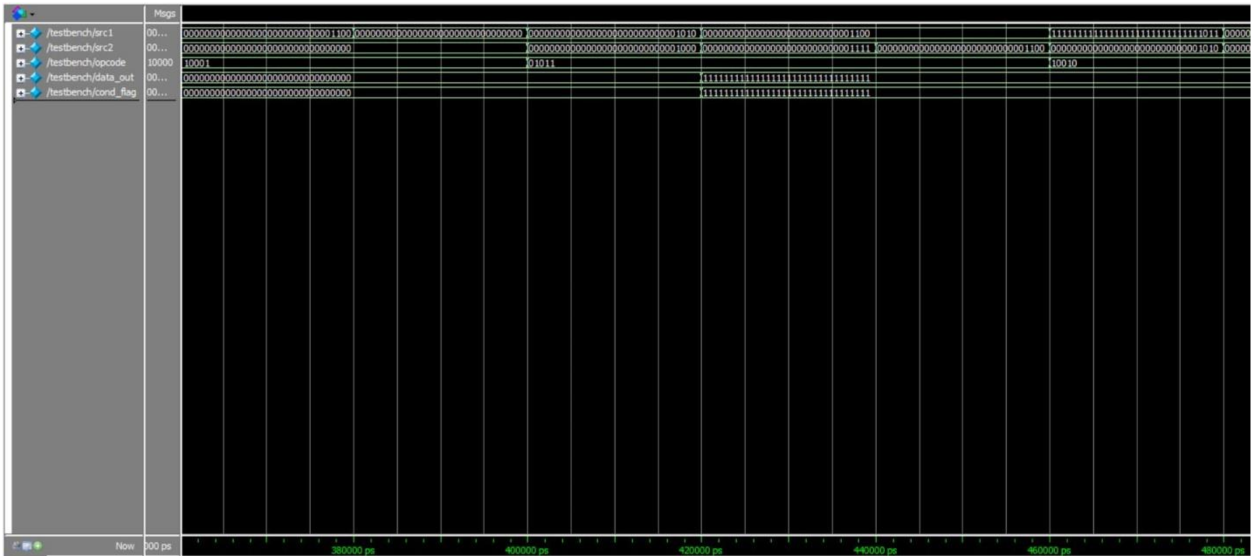
In the testbench of ClockDivider, we first needed to generate 50 MHz clock to test it. For this reason, we use `timescale 1ns/100ps and set that the value of clk is consistently changed at 10 ns interval for 1 second. [1000000000 ns == 1 second] In other word, the clk' value will be changed 100000000 times, and there are total 50000000 positive edges and another 50000000 negative edeges. This is exactly the same what Clock_50 generates. [1 MHz = 10^6 Hz] In the verilog ClockDivider code, it will return 1 clock cycle output for every 250000 times of the clock cycle input from our testbench. This should theoretically work fine, I constantly observed that I missed 1 cycle while I was testing this module. After running the test several more times, I assume that the reason why that happened was changing the clock output of the ClockDivider and changing the value of clk in the testbench were not performed at exact same time. [But expected performing at the same time] So, the time difference between those two tasks eventually affects on the last cycle that was not completed. Therefore, we changed the counting times for the output clock of our ClockDivider to 24448 from 250000. After that, we can achieve 100 clock cycle from 50 MHz clock. The count_check is the variable that holds the number of output cycles, and its value is 100 in the waveform image as it shows.

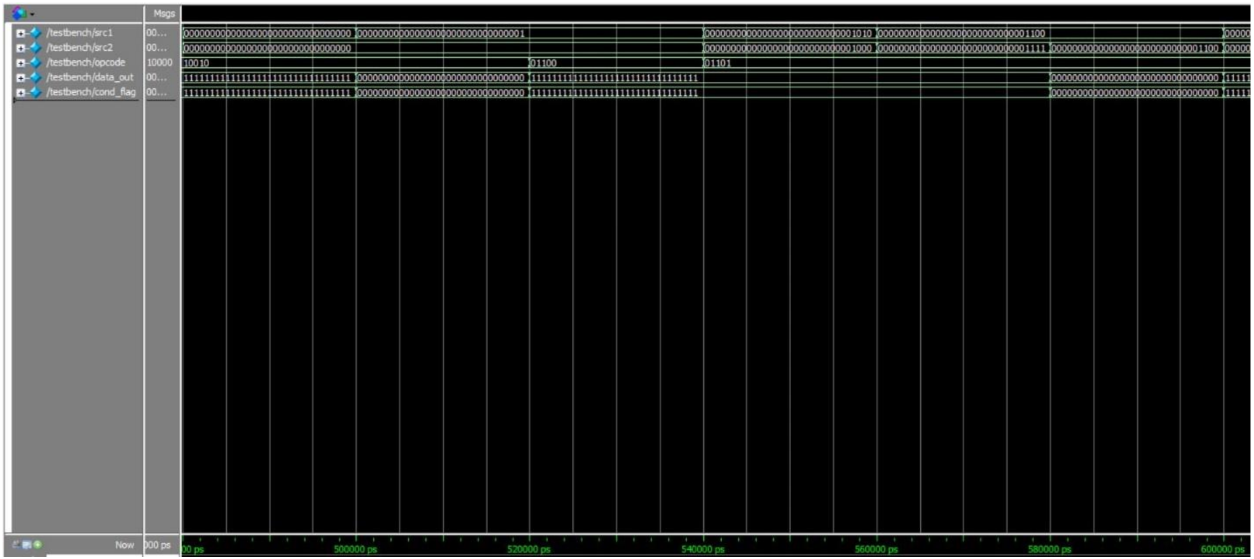
ALU.v

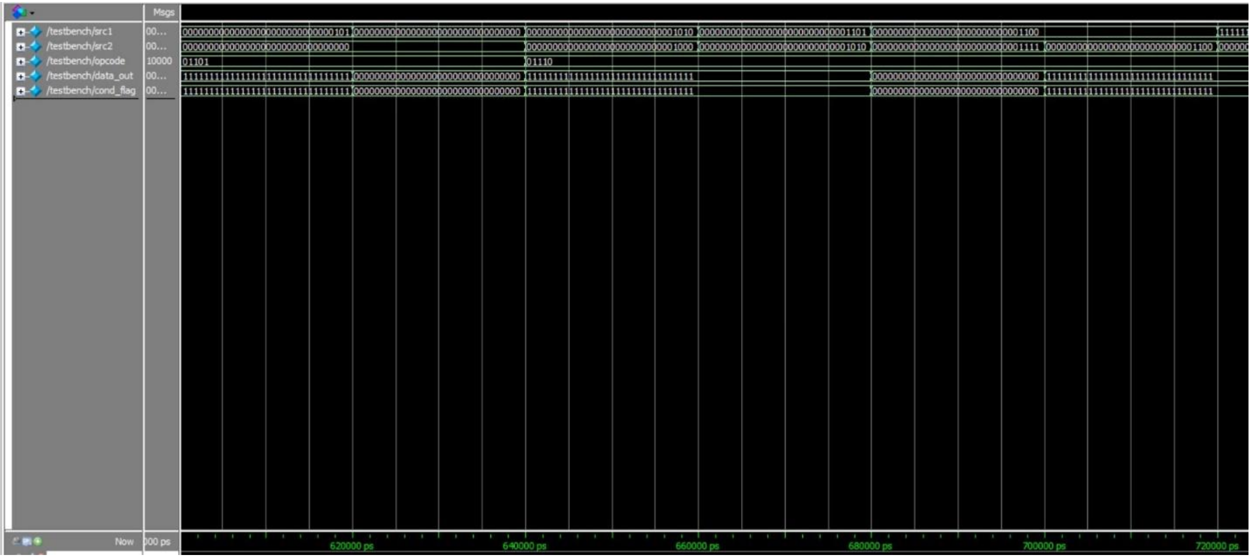

Msgs
/testbench/src1
/testbench/src2
/testbench/opcode
/testbench/data_out
/testbench/cond_flag
00...
00...
10000
00...
00...
00000000000000000000000000001010
00000000000000000000000000001000
00000
00001
00010
00011
00100
00101
00000000000000000000000000000000
00000000000000000000000000010010
00000000000000000000000000000010
00000000000000000000000000001000
00000000000000000000000000001010
00000000000000000000000000000010
11111
00000000000000000000000000000000
Now
000 ps
ps
20000 ps
40000 ps
60000 ps
80000 ps
100000 ps
120000 ps

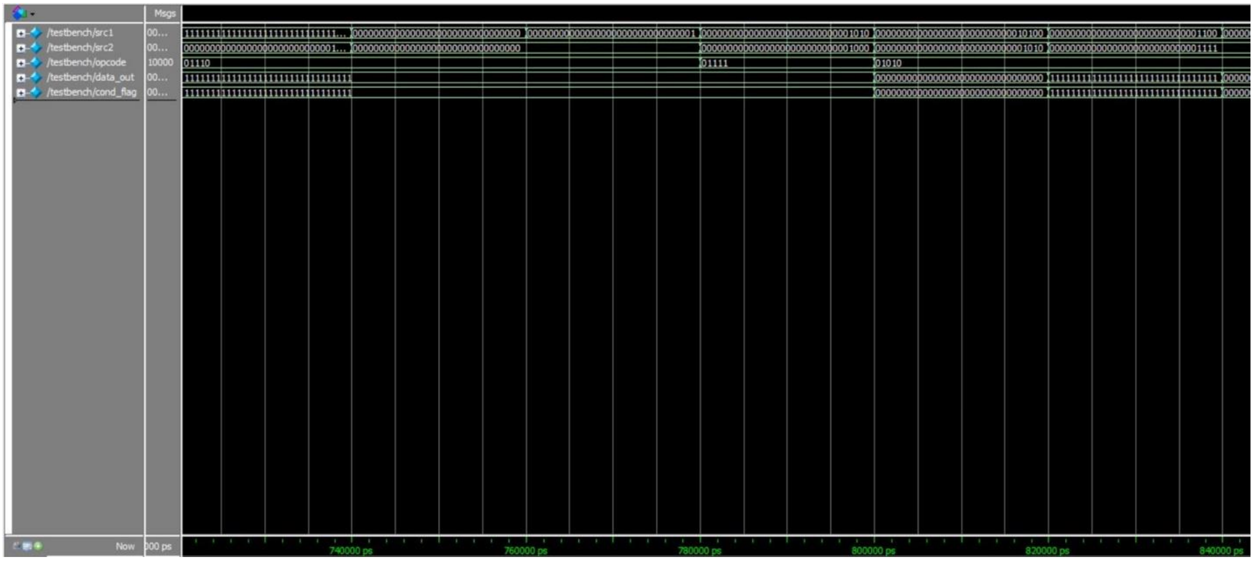

Msgs
/testbench/src1
/testbench/src2
/testbench/opcode
/testbench/data_out
/testbench/cond_flag
00...
00...
10000
00...
00...
00000000000000000000000000001010
00000000000000000000000000010010
00000
00000000000000000000000000001000
00000
00101
00110
00111
01000
01001
111111111111111111111111111111...
11111111111111111111111111110101
11111111111111111111111111111101
00000000000000000000000000000000
00000000000000000000000000000000
Now
000 ps
140000 ps
160000 ps
180000 ps
200000 ps
220000 ps
240000 ps

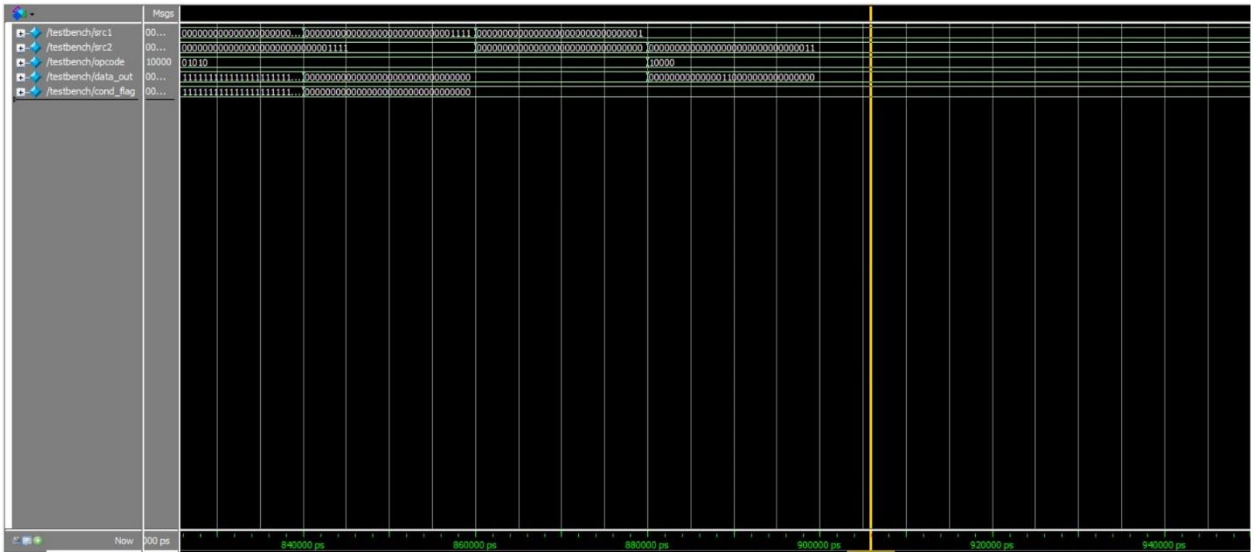

Msgs
/testbench/src1
/testbench/src2
/testbench/opcode
/testbench/data_out
/testbench/cond_flag
00...
00...
10000
00...
00...
00000000000000000000000000001000
00000000000000000000000000000000
00000000000000000000000000001010
00000000000000000000000000001100
11111111111111111111111111110100
00000
00000000000000000000000000000000
00000000000000000000000000001000
00000000000000000000000000001111
00000000000000000000000000001100
00000000000000000000000000000000
01001
01010
10001
00000000000000000000000000000000
11111111111111111111111111111111
00000000000000000000000000000000
11111111111111111111111111111111
00000000000000000000000000000000
11111111111111111111111111111111
00000
00000000000000000000000000000000
11111111111111111111111111111111
00000000000000000000000000000000
11111111111111111111111111111111
00000000000000000000000000000000
11111111111111111111111111111111
00000
Now
000 ps
00 ps
260000 ps
280000 ps
300000 ps
320000 ps
340000 ps
360000 ps


Msgs
/testbench/src1 00... 00000000000000000000000000001100 00000000000000000000000000000000 00000000000000000000000000001010 00000000000000000000000000001100 11111111111111111111111111111011 00000
/testbench/src2 00... 00000000000000000000000000000000 00000000000000000000000000001000 00000000000000000000000000001111 00000000000000000000000000001100 00000000000000000000000000001010 00000
/testbench/opcode 10000 10001 01011 10010
/testbench/data_out 00... 00000000000000000000000000000000 11111111111111111111111111111111
/testbench/cond_flag 00... 00000000000000000000000000000000 11111111111111111111111111111111
Now 00 ps
380000 ps
400000 ps
420000 ps
440000 ps
460000 ps
480000 ps

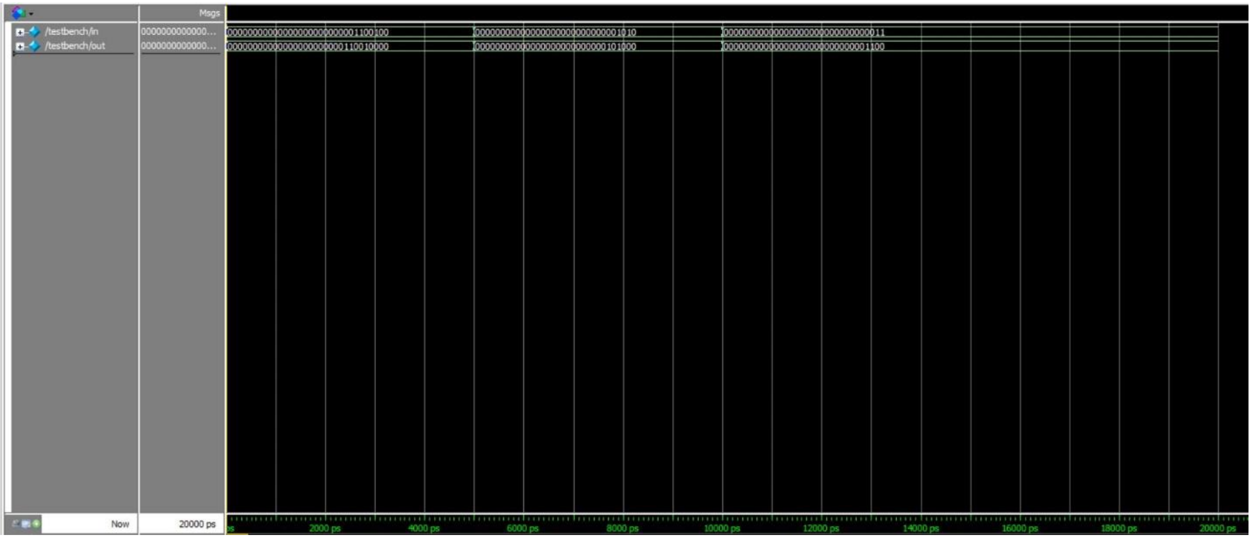

Msgs
/testbench/src1 00... 00000000000000000000000000000000 00000000000000000000000000000001 00000000000000000000000000001010 00000000000000000000000000001100 00000
/testbench/src2 00... 00000000000000000000000000000000 00000000000000000000000000001000 00000000000000000000000000001111 00000000000000000000000000001100 00000
/testbench/opcode 10000 10010 01100 01101
/testbench/data_out 00... 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111 00000000000000000000000000000000 11111
/testbench/cond_flag 00... 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111 00000000000000000000000000000000 11111
Now 00 ps
00 ps
500000 ps
520000 ps
540000 ps
560000 ps
580000 ps
600000 ps


Msgs
/testbench/src1 00... 00000000000000000000000000000101 00000000000000000000000000000000 00000000000000000000000000001010 00000000000000000000000000001101 00000000000000000000000000001100 11111
/testbench/src2 00... 00000000000000000000000000000000 00000000000000000000000000001000 00000000000000000000000000001010 00000000000000000000000000001111 00000000000000000000000000001100 00000
/testbench/opcode 10000 01101 01110
/testbench/data_out 00... 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111
/testbench/cond_flag 00... 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111
Now 00 ps
620000 ps
640000 ps
660000 ps
680000 ps
700000 ps
720000 ps

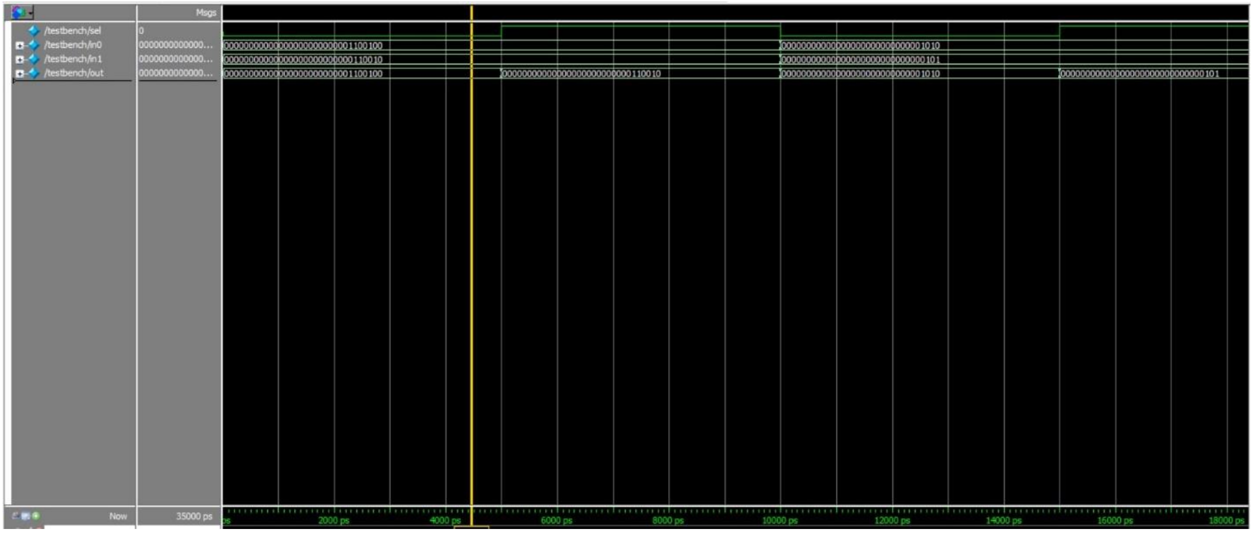

Msgs
/testbench/src1 00... 11111111111111111111111111111... 00000000000000000000000000000000 00000000000000000000000000000001 00000000000000000000000000001010 00000000000000000000000000010100 00000000000000000000000000001100 00000
/testbench/src2 00... 000000000000000000000000000001... 00000000000000000000000000000000 00000000000000000000000000001000 00000000000000000000000000001010 00000000000000000000000000001111
/testbench/opcode 10000 01110 01111 01010
/testbench/data_out 00... 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111 00000
/testbench/cond_flag 00... 11111111111111111111111111111111 00000000000000000000000000000000 11111111111111111111111111111111 00000
Now 000 ps
740000 ps
760000 ps
780000 ps
800000 ps
820000 ps
840000 ps

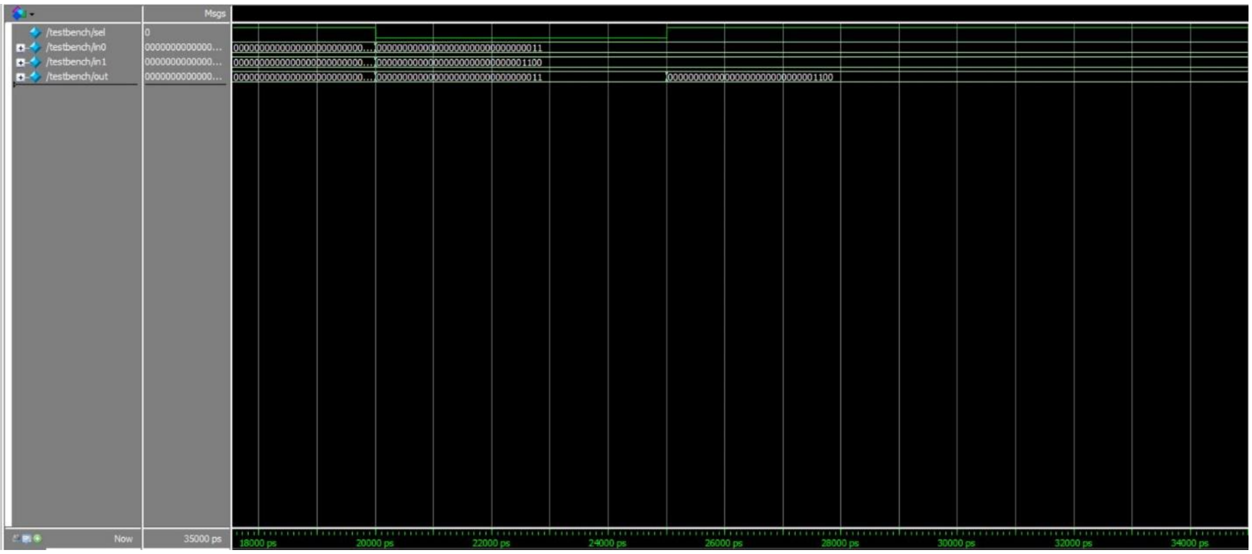

Msgs
/testbench/src1 00... 0000000000000000000000... 00000000000000000000000000001111 00000000000000000000000000000001
/testbench/src2 00... 00000000000000000000000000001111 00000000000000000000000000000000 00000000000000000000000000000011
/testbench/opcode 10000 01010 10000
/testbench/data_out 00... 1111111111111111111111... 00000000000000000000000000000000 00000000000000110000000000000000
/testbench/cond_flag 00... 1111111111111111111111... 00000000000000000000000000000000
Now 000 ps
840000 ps
860000 ps
880000 ps
900000 ps
920000 ps
940000 ps

Shifter.v

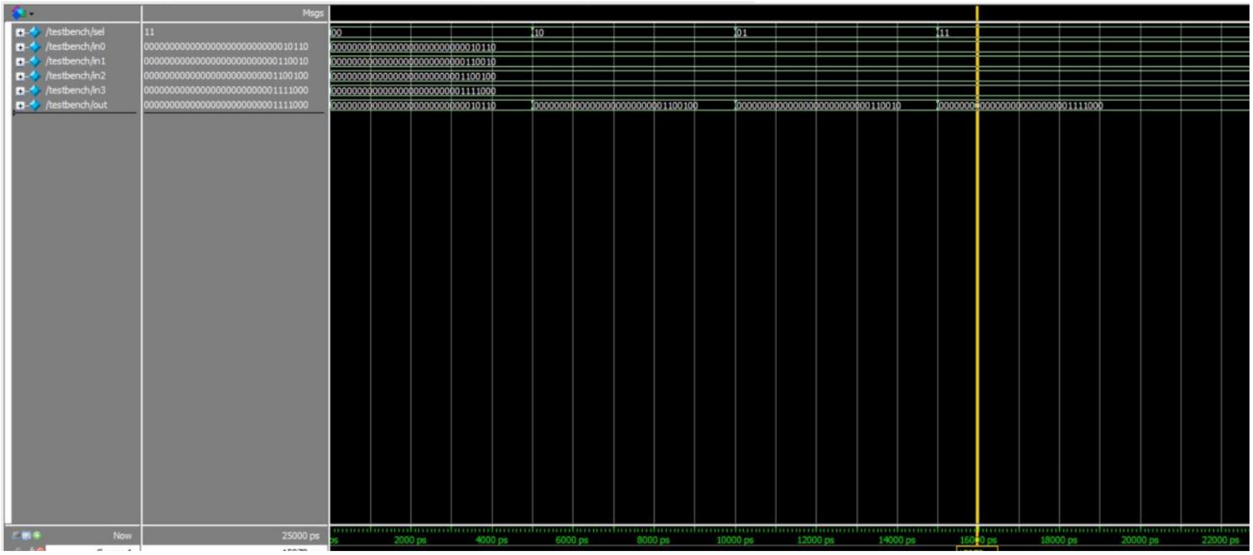

Msgs
/testbench/in
/testbench/out
0000000000000...
0000000000000...
00000000000000000000000001100100
00000000000000000000000000001010
00000000000000000000000000000011
00000000000000000000000110010000
00000000000000000000000000101000
00000000000000000000000000001100
Now
20000 ps
ps
2000 ps
4000 ps
6000 ps
8000 ps
10000 ps
12000 ps
14000 ps
16000 ps
18000 ps
20000 ps

Mux2x1.v


Msgs
/testbench/sel
/testbench/in0
/testbench/in1
/testbench/out
0
0000000000000...
0000000000000...
0000000000000...
00000000000000000000000001100100
00000000000000000000000000110010
00000000000000000000000001100100
00000000000000000000000000110010
00000000000000000000000000001010
00000000000000000000000000000101
00000000000000000000000000001010
00000000000000000000000000000101
Now
35000 ps
ps
2000 ps
4000 ps
6000 ps
8000 ps
10000 ps
12000 ps
14000 ps
16000 ps
18000 ps


Msgs
/testbench/sel
/testbench/in0
/testbench/in1
/testbench/out
0
0000000000000...
0000000000000...
0000000000000...
000000000000000000000000...
00000000000000000000000000000011
00000000000000000000000000001100
00000000000000000000000000000011
00000000000000000000000000001100
Now
35000 ps
18000 ps
20000 ps
22000 ps
24000 ps
26000 ps
28000 ps
30000 ps
32000 ps
34000 ps

Mux4x1.v

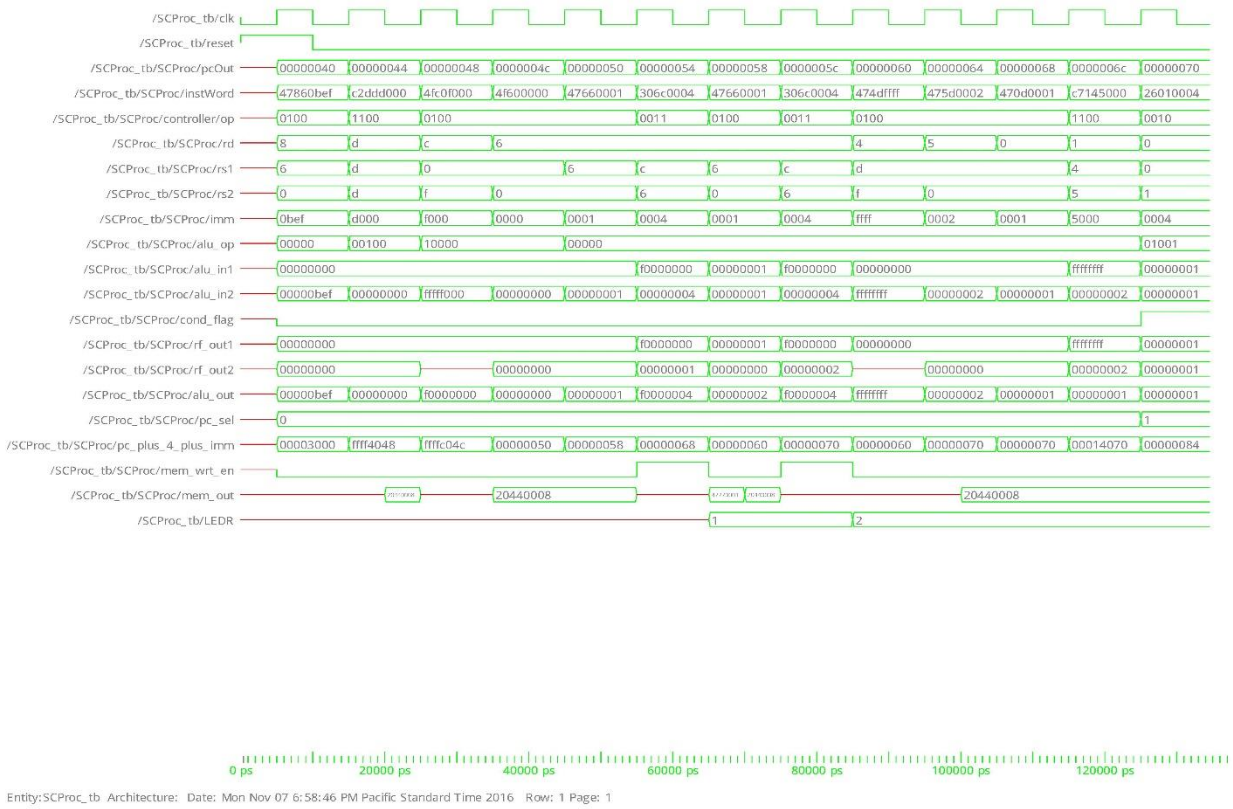


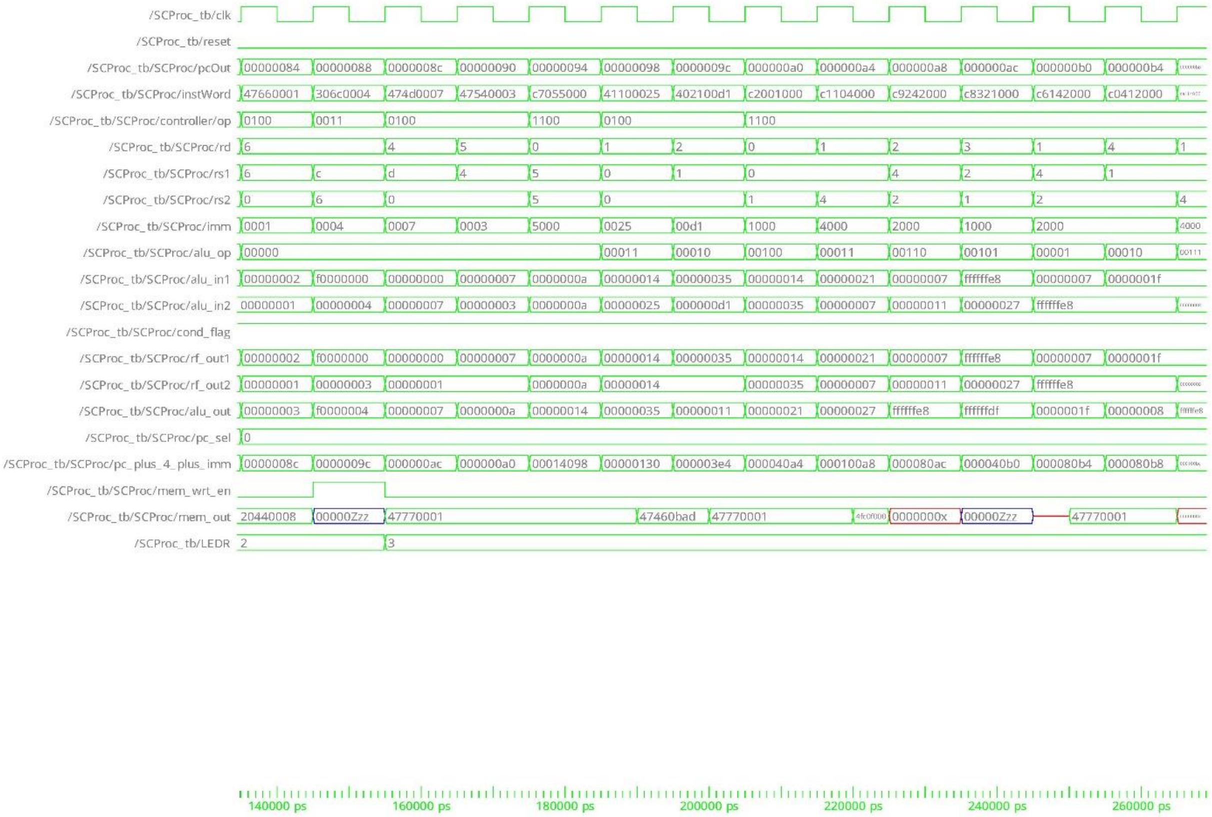
```
                   Msgs
/testbench/sel     11
/testbench/in0     00000000000000000000000000010110
/testbench/in1     00000000000000000000000000110010
/testbench/in2     00000000000000000000000001100100
/testbench/in3     00000000000000000000000001111000
/testbench/out     00000000000000000000000001111000

Now                25000 ps
```

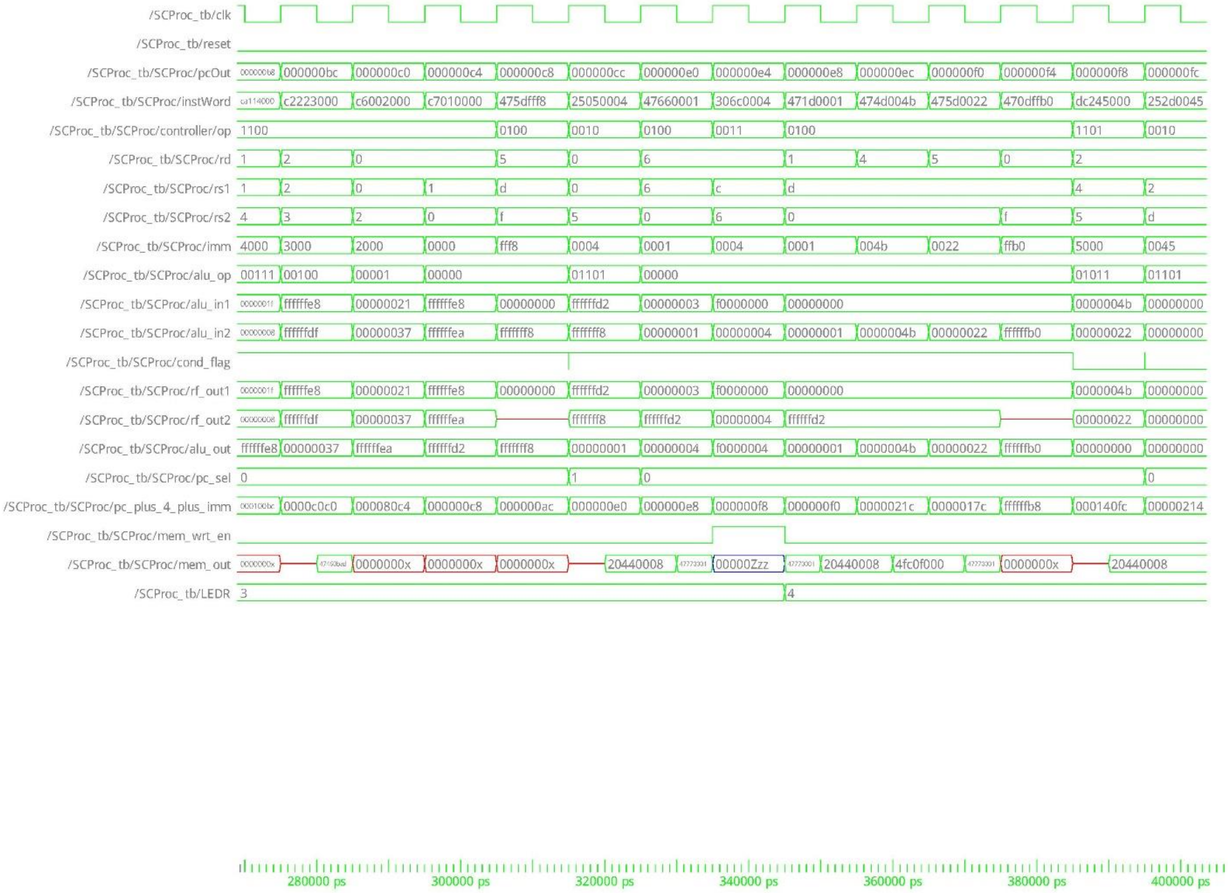
Single Cycle Processor Running Test2.mif

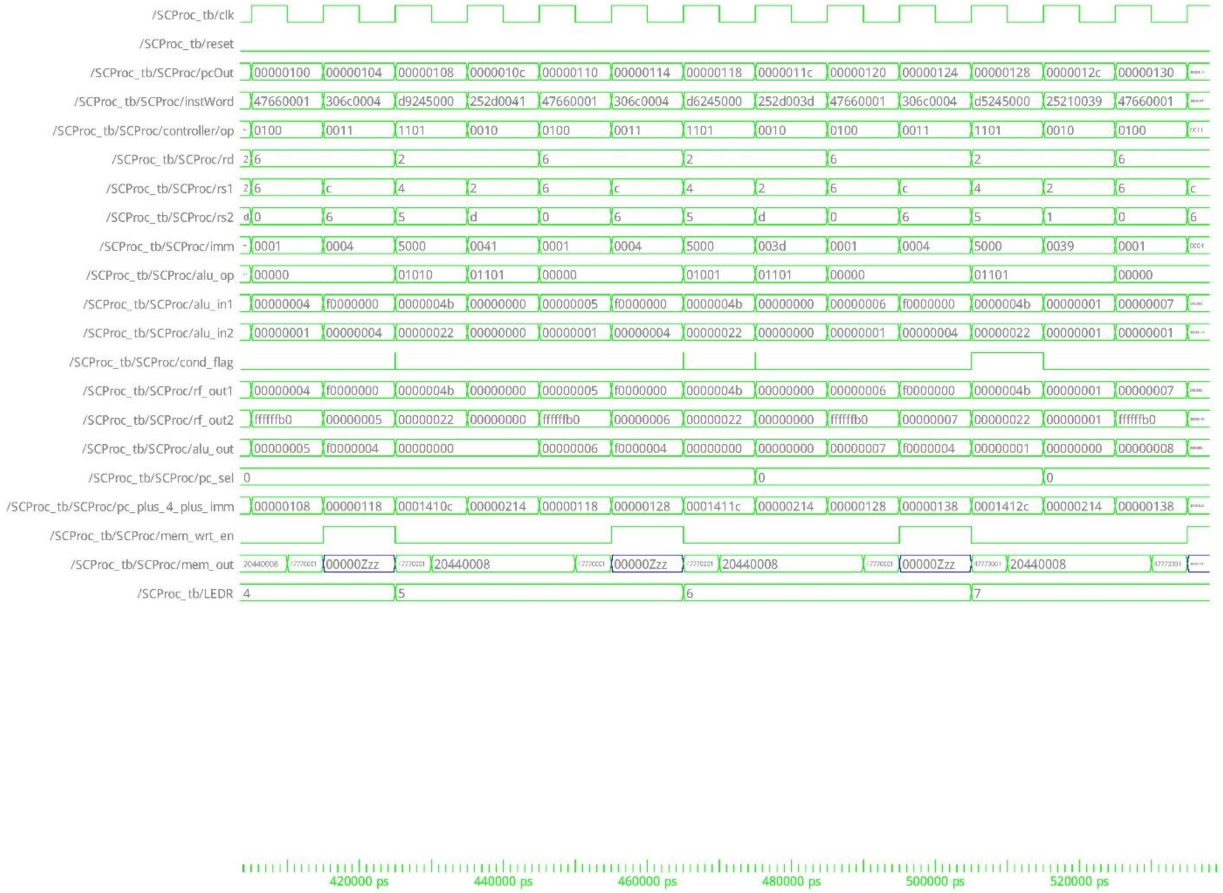
We can see that LEDR goes up to 24 as expected in the last line of the waveforms.

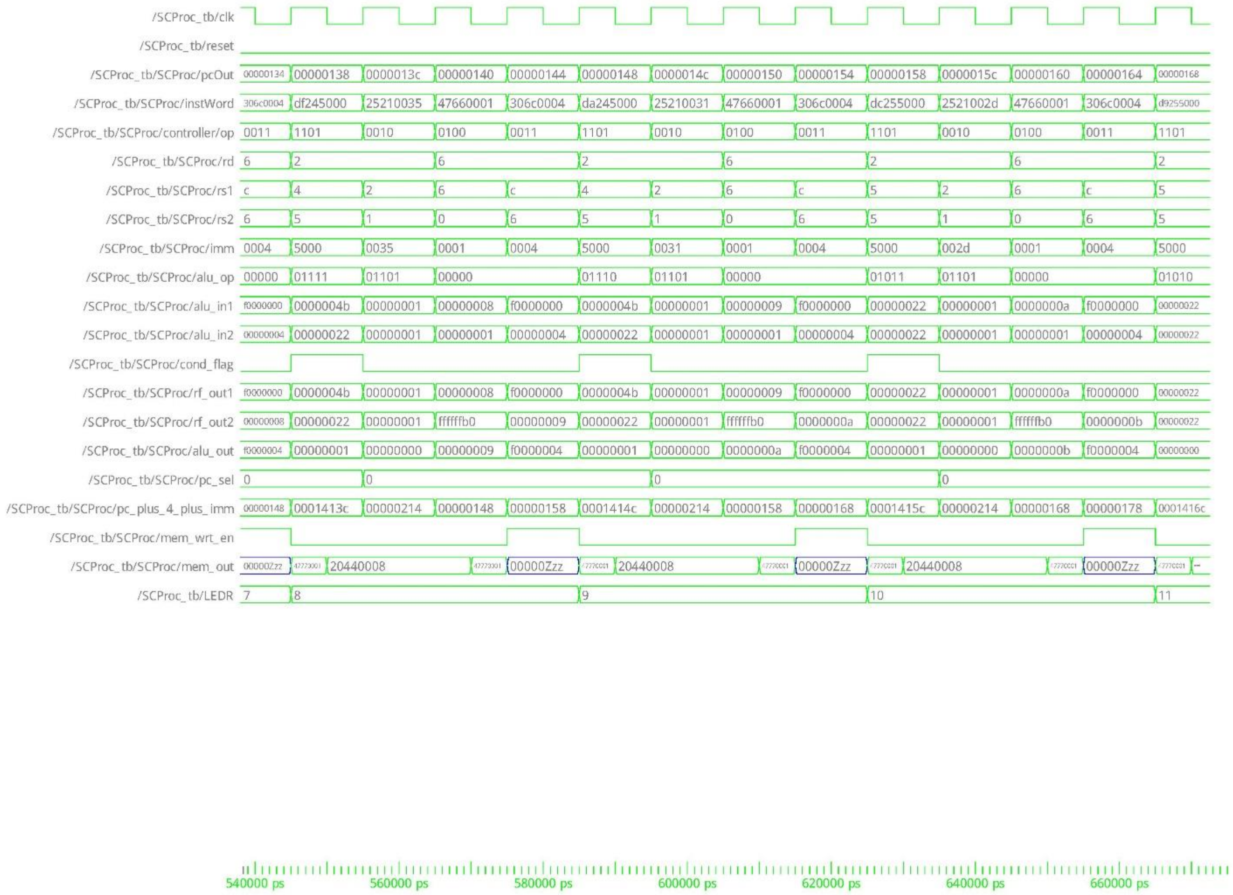


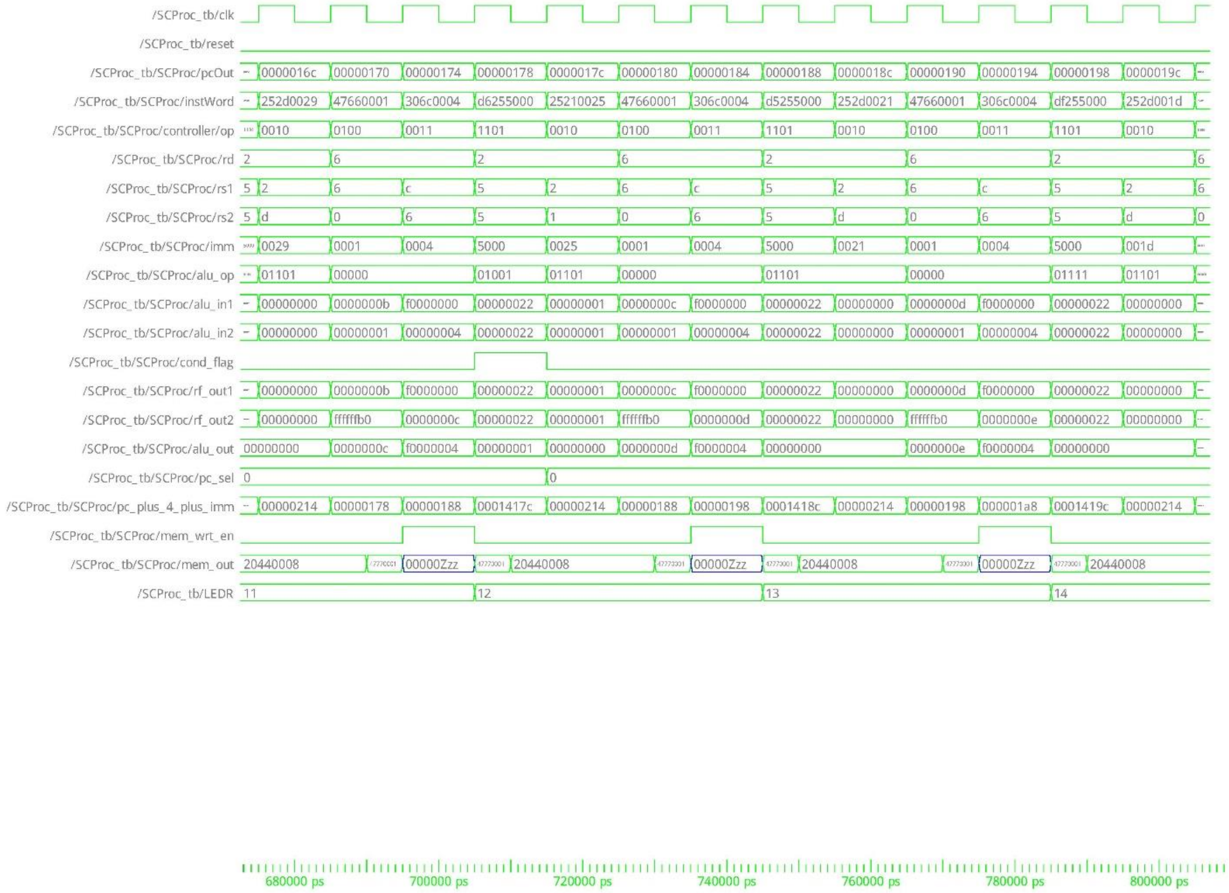


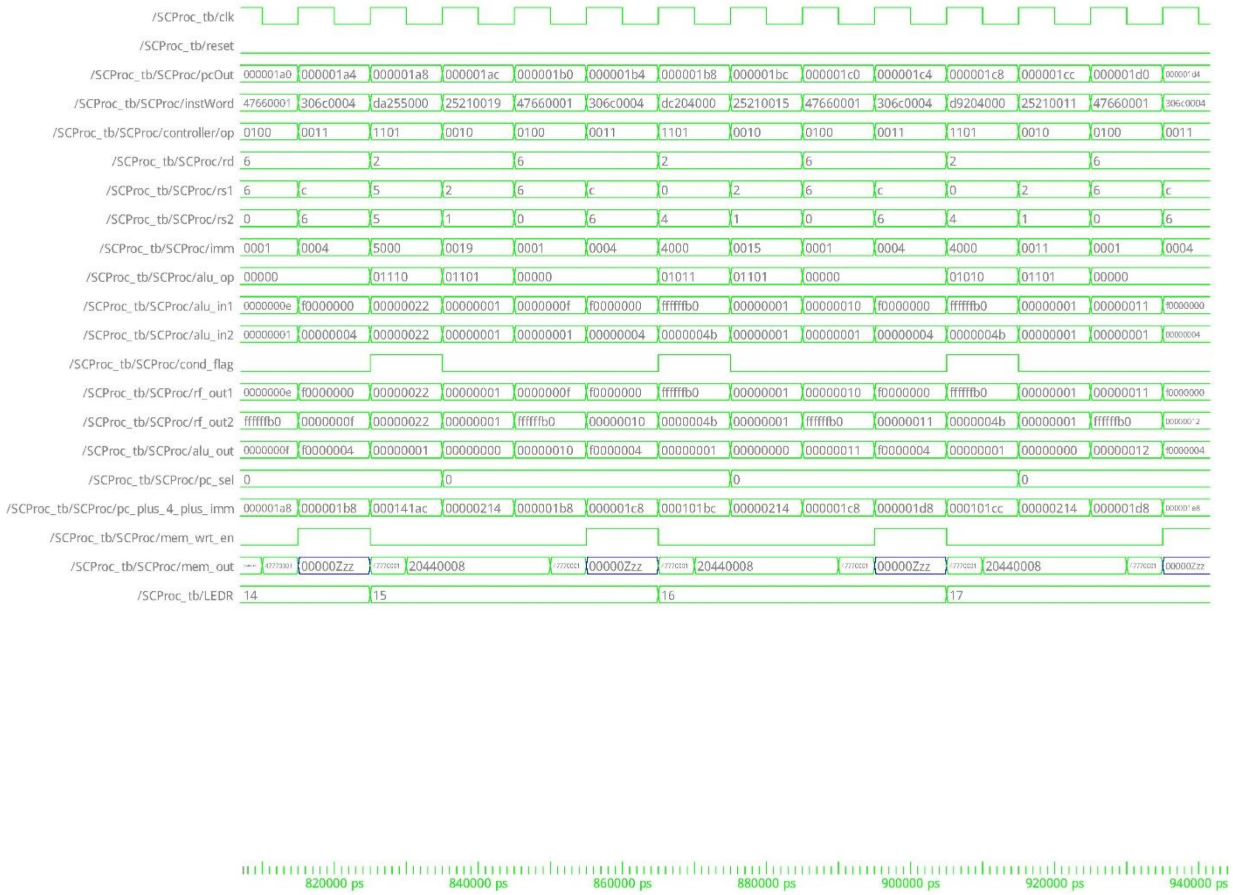
Entity:SCProc_tb Architecture: Date: Mon Nov 07 6:58:46 PM Pacific Standard Time 2016 Row: 2 Page: 2


/SCProc_tb/clk
/SCProc_tb/reset
/SCProc_tb/SCProc/pcOut 000000b8 000000bc 000000c0 000000c4 000000c8 000000cc 000000e0 000000e4 000000e8 000000ec 000000f0 000000f4 000000f8 000000fc
/SCProc_tb/SCProc/instWord ca114000 c2223000 c6002000 c7010000 475dfff8 25050004 47660001 306c0004 471d0001 474d004b 475d0022 470dffb0 dc245000 252d0045
/SCProc_tb/SCProc/controller/op 1100 0100 0010 0100 0011 0100 1101 0010
/SCProc_tb/SCProc/rd 1 2 0 5 0 6 1 4 5 0 2
/SCProc_tb/SCProc/rs1 1 2 0 1 d 0 6 c d 4 2
/SCProc_tb/SCProc/rs2 4 3 2 0 f 5 0 6 0 f 5 d
/SCProc_tb/SCProc/imm 4000 3000 2000 0000 fff8 0004 0001 0004 0001 004b 0022 ffb0 5000 0045
/SCProc_tb/SCProc/alu_op 00111 00100 00001 00000 01101 00000 01011 01101
/SCProc_tb/SCProc/alu_in1 0000001f ffffffe8 00000021 ffffffe8 00000000 ffffffd2 00000003 f0000000 00000000 0000004b 00000000
/SCProc_tb/SCProc/alu_in2 00000008 ffffffdf 00000037 ffffffea fffffff8 fffffff8 00000001 00000004 00000001 0000004b 00000022 ffffffb0 00000022 00000000
/SCProc_tb/SCProc/cond_flag
/SCProc_tb/SCProc/rf_out1 0000001f ffffffe8 00000021 ffffffe8 00000000 ffffffd2 00000003 f0000000 00000000 0000004b 00000000
/SCProc_tb/SCProc/rf_out2 00000008 ffffffdf 00000037 ffffffea fffffff8 ffffffd2 00000004 ffffffd2 00000022 00000000
/SCProc_tb/SCProc/alu_out ffffffe8 00000037 ffffffea ffffffd2 fffffff8 00000001 00000004 f0000004 00000001 0000004b 00000022 ffffffb0 00000000 00000000
/SCProc_tb/SCProc/pc_sel 0 1 0 0
/SCProc_tb/SCProc/pc_plus_4_plus_imm 000100bc 0000c0c0 000080c4 000000c8 000000ac 000000e0 000000e8 000000f8 000000f0 0000021c 0000017c ffffffb8 000140fc 00000214
/SCProc_tb/SCProc/mem_wrt_en
/SCProc_tb/SCProc/mem_out 0000000x 47150bad 0000000x 0000000x 0000000x 20440008 47773001 00000Zzz 47773001 20440008 4fc0f000 47773001 0000000x 20440008
/SCProc_tb/LEDR 3 4
280000 ps
300000 ps
320000 ps
340000 ps
360000 ps
380000 ps
400000 ps

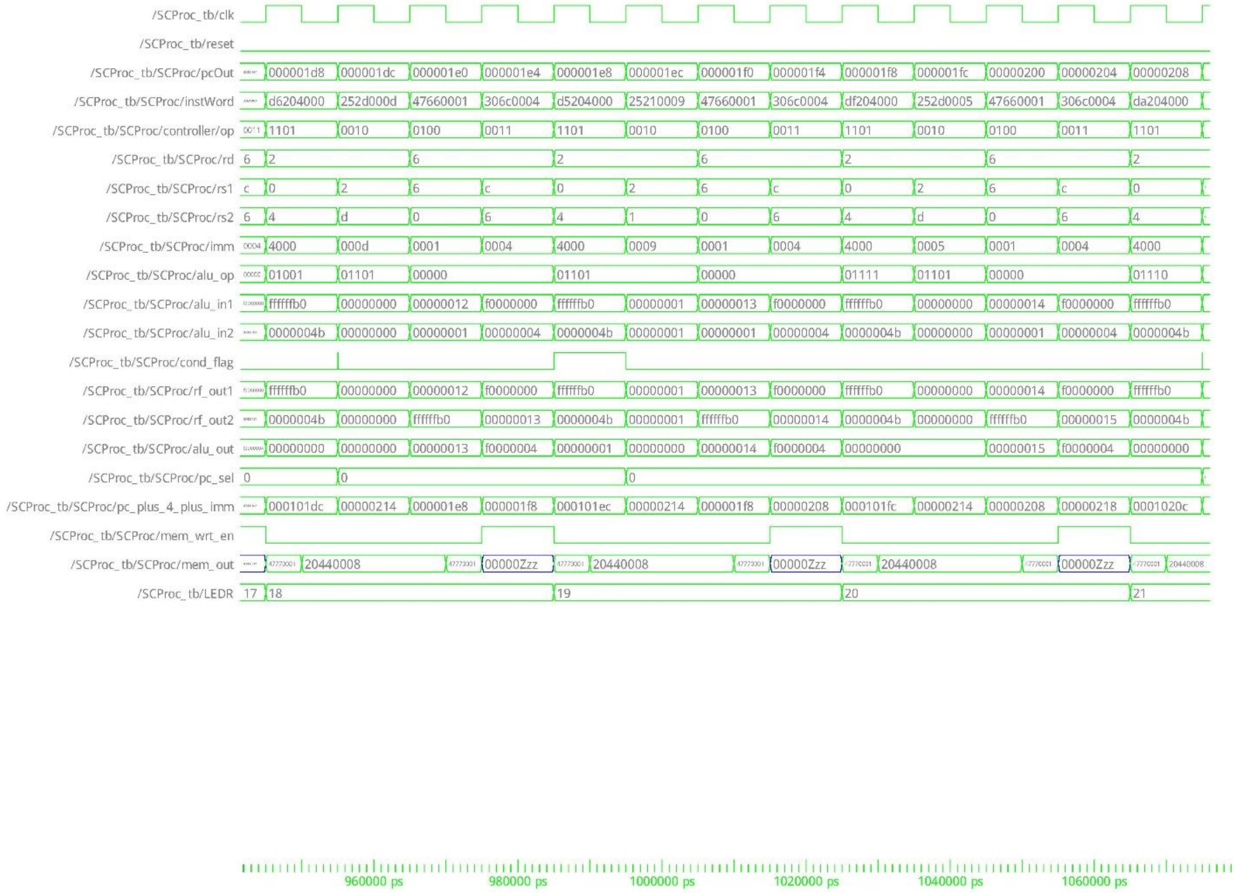

/SCProc_tb/clk
/SCProc_tb/reset
/SCProc_tb/SCProc/pcOut 00000100 00000104 00000108 0000010c 00000110 00000114 00000118 0000011c 00000120 00000124 00000128 0000012c 00000130
/SCProc_tb/SCProc/instWord 47660001 306c0004 d9245000 252d0041 47660001 306c0004 d6245000 252d003d 47660001 306c0004 d5245000 25210039 47660001
/SCProc_tb/SCProc/controller/op 0100 0011 1101 0010 0100 0011 1101 0010 0100 0011 1101 0010 0100
/SCProc_tb/SCProc/rd 2 6 2 6 2 6 2 6
/SCProc_tb/SCProc/rs1 2 6 c 4 2 6 c 4 2 6 c 4 2 6 c
/SCProc_tb/SCProc/rs2 d 0 6 5 d 0 6 5 d 0 6 5 1 0 6
/SCProc_tb/SCProc/imm 0001 0004 5000 0041 0001 0004 5000 003d 0001 0004 5000 0039 0001
/SCProc_tb/SCProc/alu_op 00000 01010 01101 00000 01001 01101 00000 01101 00000
/SCProc_tb/SCProc/alu_in1 00000004 f0000000 0000004b 00000000 00000005 f0000000 0000004b 00000000 00000006 f0000000 0000004b 00000001 00000007
/SCProc_tb/SCProc/alu_in2 00000001 00000004 00000022 00000000 00000001 00000004 00000022 00000000 00000001 00000004 00000022 00000001 00000001
/SCProc_tb/SCProc/cond_flag
/SCProc_tb/SCProc/rf_out1 00000004 f0000000 0000004b 00000000 00000005 f0000000 0000004b 00000000 00000006 f0000000 0000004b 00000001 00000007
/SCProc_tb/SCProc/rf_out2 ffffffb0 00000005 00000022 00000000 ffffffb0 00000006 00000022 00000000 ffffffb0 00000007 00000022 00000001 ffffffb0
/SCProc_tb/SCProc/alu_out 00000005 f0000004 00000000 00000006 f0000004 00000000 00000000 00000007 f0000004 00000001 00000000 00000008
/SCProc_tb/SCProc/pc_sel 0 0 0
/SCProc_tb/SCProc/pc_plus_4_plus_imm 00000108 00000118 0001410c 00000214 00000118 00000128 0001411c 00000214 00000128 00000138 0001412c 00000214 00000138
/SCProc_tb/SCProc/mem_wrt_en
/SCProc_tb/SCProc/mem_out 20440008 00000Zzz 20440008 00000Zzz 20440008 00000Zzz 20440008
/SCProc_tb/LEDR 4 5 6 7
420000 ps
440000 ps
460000 ps
480000 ps
500000 ps
520000 ps

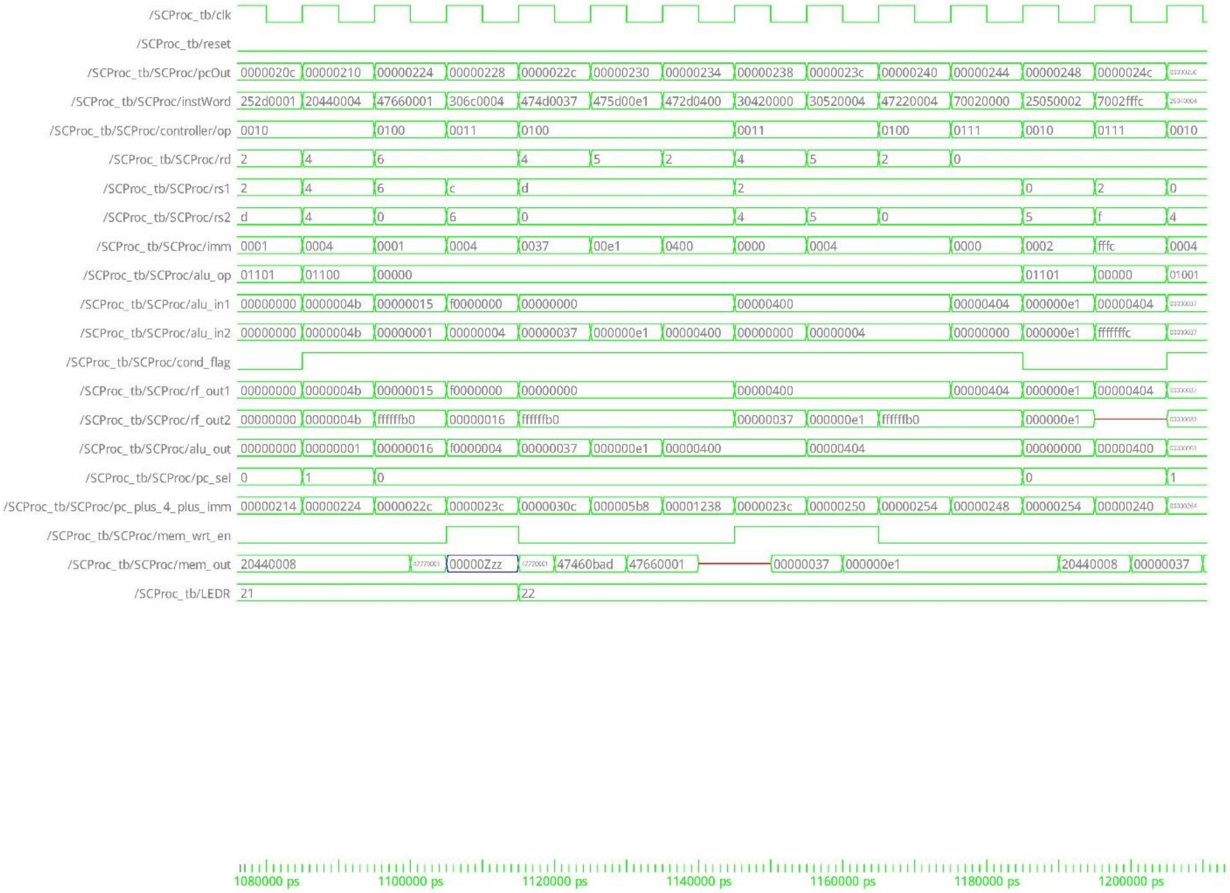

/SCProc_tb/clk
/SCProc_tb/reset
/SCProc_tb/SCProc/pcOut 00000134 00000138 0000013c 00000140 00000144 00000148 0000014c 00000150 00000154 00000158 0000015c 00000160 00000164 00000168
/SCProc_tb/SCProc/instWord 306c0004 df245000 25210035 47660001 306c0004 da245000 25210031 47660001 306c0004 dc255000 2521002d 47660001 306c0004 d9255000
/SCProc_tb/SCProc/controller/op 0011 1101 0010 0100 0011 1101 0010 0100 0011 1101 0010 0100 0011 1101
/SCProc_tb/SCProc/rd 6 2 6 2 6 2 6 2
/SCProc_tb/SCProc/rs1 c 4 2 6 c 4 2 6 c 5 2 6 c 5
/SCProc_tb/SCProc/rs2 6 5 1 0 6 5 1 0 6 5 1 0 6 5
/SCProc_tb/SCProc/imm 0004 5000 0035 0001 0004 5000 0031 0001 0004 5000 002d 0001 0004 5000
/SCProc_tb/SCProc/alu_op 00000 01111 01101 00000 01110 01101 00000 01011 01101 00000 01010
/SCProc_tb/SCProc/alu_in1 f0000000 0000004b 00000001 00000008 f0000000 0000004b 00000001 00000009 f0000000 00000022 00000001 0000000a f0000000 00000022
/SCProc_tb/SCProc/alu_in2 00000004 00000022 00000001 00000001 00000004 00000022 00000001 00000001 00000004 00000022 00000001 00000001 00000004 00000022
/SCProc_tb/SCProc/cond_flag
/SCProc_tb/SCProc/rf_out1 f0000000 0000004b 00000001 00000008 f0000000 0000004b 00000001 00000009 f0000000 00000022 00000001 0000000a f0000000 00000022
/SCProc_tb/SCProc/rf_out2 00000008 00000022 00000001 ffffffb0 00000009 00000022 00000001 ffffffb0 0000000a 00000022 00000001 ffffffb0 0000000b 00000022
/SCProc_tb/SCProc/alu_out f0000004 00000001 00000000 00000009 f0000004 00000001 00000000 0000000a f0000004 00000001 00000000 0000000b f0000004 00000000
/SCProc_tb/SCProc/pc_sel 0 0 0 0
/SCProc_tb/SCProc/pc_plus_4_plus_imm 00000148 0001413c 00000214 00000148 00000158 0001414c 00000214 00000158 00000168 0001415c 00000214 00000168 00000178 0001416c
/SCProc_tb/SCProc/mem_wrt_en
/SCProc_tb/SCProc/mem_out 00000Zzz 47773001 20440008 47773001 00000Zzz 4777c001 20440008 4777c001 00000Zzz 4777c001 20440008 4777c001 00000Zzz 4777c001 -
/SCProc_tb/LEDR 7 8 9 10 11
540000 ps
560000 ps
580000 ps
600000 ps
620000 ps
640000 ps
660000 ps

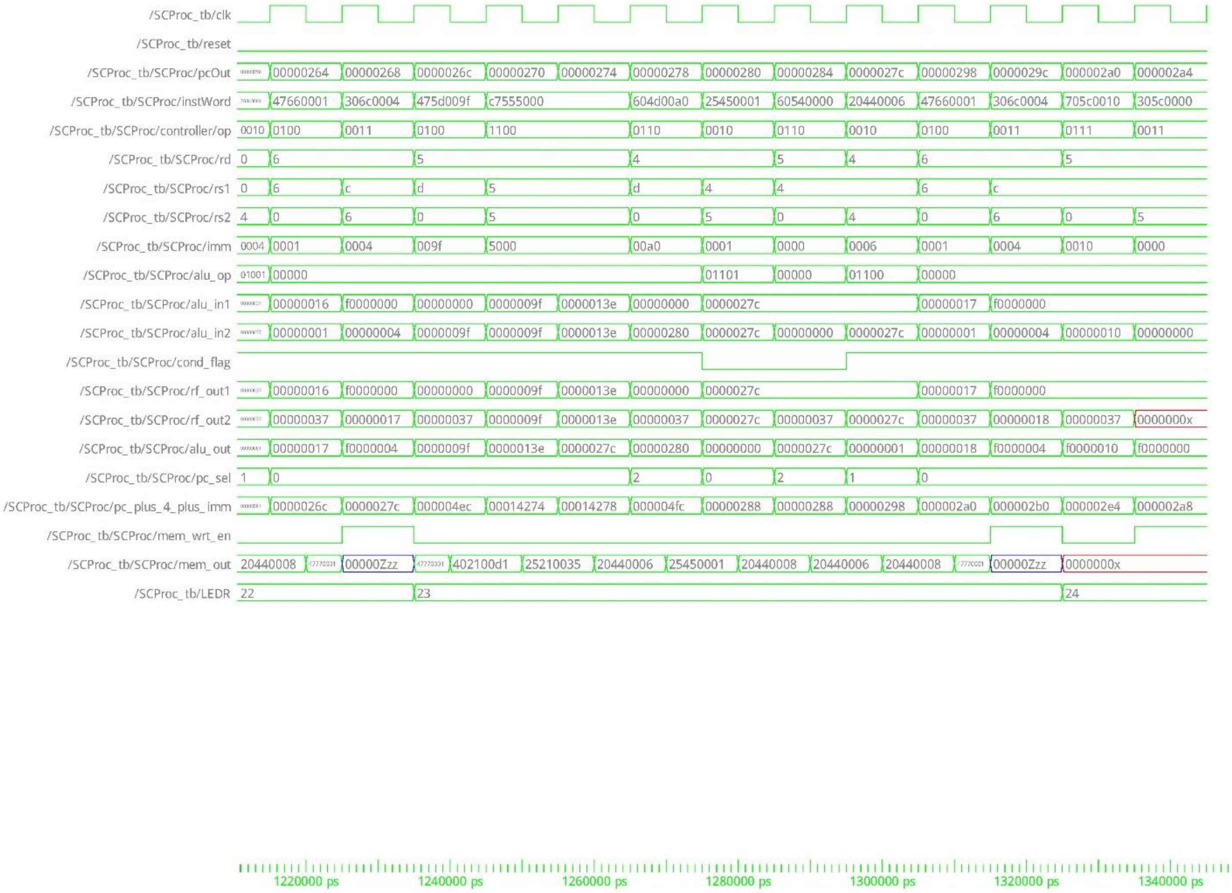

/SCProc_tb/clk
/SCProc_tb/reset
/SCProc_tb/SCProc/pcOut 0000016c 00000170 00000174 00000178 0000017c 00000180 00000184 00000188 0000018c 00000190 00000194 00000198 0000019c
/SCProc_tb/SCProc/instWord 252d0029 47660001 306c0004 d6255000 25210025 47660001 306c0004 d5255000 252d0021 47660001 306c0004 df255000 252d001d
/SCProc_tb/SCProc/controller/op 0010 0100 0011 1101 0010 0100 0011 1101 0010 0100 0011 1101 0010
/SCProc_tb/SCProc/rd 2 6 2 6 2 6 2 6
/SCProc_tb/SCProc/rs1 5 2 6 c 5 2 6 c 5 2 6 c 5 2 6
/SCProc_tb/SCProc/rs2 5 d 0 6 5 1 0 6 5 d 0 6 5 d 0
/SCProc_tb/SCProc/imm 0029 0001 0004 5000 0025 0001 0004 5000 0021 0001 0004 5000 001d
/SCProc_tb/SCProc/alu_op 01101 00000 01001 01101 00000 01101 00000 01111 01101
/SCProc_tb/SCProc/alu_in1 00000000 0000000b f0000000 00000022 00000001 0000000c f0000000 00000022 00000000 0000000d f0000000 00000022 00000000
/SCProc_tb/SCProc/alu_in2 00000000 00000001 00000004 00000022 00000001 00000001 00000004 00000022 00000000 00000001 00000004 00000022 00000000
/SCProc_tb/SCProc/cond_flag
/SCProc_tb/SCProc/rf_out1 00000000 0000000b f0000000 00000022 00000001 0000000c f0000000 00000022 00000000 0000000d f0000000 00000022 00000000
/SCProc_tb/SCProc/rf_out2 00000000 ffffffb0 0000000c 00000022 00000001 ffffffb0 0000000d 00000022 00000000 ffffffb0 0000000e 00000022 00000000
/SCProc_tb/SCProc/alu_out 00000000 0000000c f0000004 00000001 00000000 0000000d f0000004 00000000 0000000e f0000004 00000000
/SCProc_tb/SCProc/pc_sel 0 0
/SCProc_tb/SCProc/pc_plus_4_plus_imm 00000214 00000178 00000188 0001417c 00000214 00000188 00000198 0001418c 00000214 00000198 000001a8 0001419c 00000214
/SCProc_tb/SCProc/mem_wrt_en
/SCProc_tb/SCProc/mem_out 20440008 00000Zzz 20440008 00000Zzz 20440008 00000Zzz 20440008
/SCProc_tb/LEDR 11 12 13 14
680000 ps
700000 ps
720000 ps
740000 ps
760000 ps
780000 ps
800000 ps



Entity:SCProc_tb Architecture: Date: Mon Nov 07 6:58:46 PM Pacific Standard Time 2016 Row: 7 Page: 7


/SCProc_tb/clk
/SCProc_tb/reset
/SCProc_tb/SCProc/pcOut 000001d8 000001dc 000001e0 000001e4 000001e8 000001ec 000001f0 000001f4 000001f8 000001fc 00000200 00000204 00000208
/SCProc_tb/SCProc/instWord d6204000 252d000d 47660001 306c0004 d5204000 25210009 47660001 306c0004 df204000 252d0005 47660001 306c0004 da204000
/SCProc_tb/SCProc/controller/op 1101 0010 0100 0011 1101 0010 0100 0011 1101 0010 0100 0011 1101
/SCProc_tb/SCProc/rd 6 2 6 2 6 2 6 2
/SCProc_tb/SCProc/rs1 c 0 2 6 c 0 2 6 c 0 2 6 c 0
/SCProc_tb/SCProc/rs2 6 4 d 0 6 4 1 0 6 4 d 0 6 4
/SCProc_tb/SCProc/imm 4000 000d 0001 0004 4000 0009 0001 0004 4000 0005 0001 0004 4000
/SCProc_tb/SCProc/alu_op 01001 01101 00000 01101 00000 01111 01101 00000 01110
/SCProc_tb/SCProc/alu_in1 ffffffb0 00000000 00000012 f0000000 ffffffb0 00000001 00000013 f0000000 ffffffb0 00000000 00000014 f0000000 ffffffb0
/SCProc_tb/SCProc/alu_in2 0000004b 00000000 00000001 00000004 0000004b 00000001 00000001 00000004 0000004b 00000000 00000001 00000004 0000004b
/SCProc_tb/SCProc/cond_flag
/SCProc_tb/SCProc/rf_out1 ffffffb0 00000000 00000012 f0000000 ffffffb0 00000001 00000013 f0000000 ffffffb0 00000000 00000014 f0000000 ffffffb0
/SCProc_tb/SCProc/rf_out2 0000004b 00000000 ffffffb0 00000013 0000004b 00000001 ffffffb0 00000014 0000004b 00000000 ffffffb0 00000015 0000004b
/SCProc_tb/SCProc/alu_out 00000000 00000000 00000013 f0000004 00000001 00000000 00000014 f0000004 00000000 00000015 f0000004 00000000
/SCProc_tb/SCProc/pc_sel 0 0 0
/SCProc_tb/SCProc/pc_plus_4_plus_imm 000101dc 00000214 000001e8 000001f8 000101ec 00000214 000001f8 00000208 000101fc 00000214 00000208 00000218 0001020c
/SCProc_tb/SCProc/mem_wrt_en
/SCProc_tb/SCProc/mem_out 20440008 00000Zzz 20440008 00000Zzz 20440008 00000Zzz 20440008
/SCProc_tb/LEDR 17 18 19 20 21
960000 ps
980000 ps
1000000 ps
1020000 ps
1040000 ps
1060000 ps


/SCProc_tb/clk
/SCProc_tb/reset
/SCProc_tb/SCProc/pcOut 0000020c 00000210 00000224 00000228 0000022c 00000230 00000234 00000238 0000023c 00000240 00000244 00000248 0000024c
/SCProc_tb/SCProc/instWord 252d0001 20440004 47660001 306c0004 474d0037 475d00e1 472d0400 30420000 30520004 47220004 70020000 25050002 7002fffc
/SCProc_tb/SCProc/controller/op 0010 0100 0011 0100 0011 0100 0111 0010 0111 0010
/SCProc_tb/SCProc/rd 2 4 6 4 5 2 4 5 2 0
/SCProc_tb/SCProc/rs1 2 4 6 c d 2 0 2 0
/SCProc_tb/SCProc/rs2 d 4 0 6 0 4 5 0 5 f 4
/SCProc_tb/SCProc/imm 0001 0004 0001 0004 0037 00e1 0400 0000 0004 0000 0002 fffc 0004
/SCProc_tb/SCProc/alu_op 01101 01100 00000 01101 00000 01001
/SCProc_tb/SCProc/alu_in1 00000000 0000004b 00000015 f0000000 00000000 00000400 00000404 000000e1 00000404
/SCProc_tb/SCProc/alu_in2 00000000 0000004b 00000001 00000004 00000037 000000e1 00000400 00000000 00000004 00000000 000000e1 fffffffc
/SCProc_tb/SCProc/cond_flag
/SCProc_tb/SCProc/rf_out1 00000000 0000004b 00000015 f0000000 00000000 00000400 00000404 000000e1 00000404
/SCProc_tb/SCProc/rf_out2 00000000 0000004b ffffffb0 00000016 ffffffb0 00000037 000000e1 ffffffb0 000000e1
/SCProc_tb/SCProc/alu_out 00000000 00000001 00000016 f0000004 00000037 000000e1 00000400 00000404 00000000 00000400
/SCProc_tb/SCProc/pc_sel 0 1 0 0 1
/SCProc_tb/SCProc/pc_plus_4_plus_imm 00000214 00000224 0000022c 0000023c 0000030c 000005b8 00001238 0000023c 00000250 00000254 00000248 00000254 00000240
/SCProc_tb/SCProc/mem_wrt_en
/SCProc_tb/SCProc/mem_out 20440008 00000Zzz 47460bad 47660001 00000037 000000e1 20440008 00000037
/SCProc_tb/LEDR 21 22
1080000 ps
1100000 ps
1120000 ps
1140000 ps
1160000 ps
1180000 ps
1200000 ps



Entity:SCProc_tb Architecture: Date: Mon Nov 07 6:58:46 PM Pacific Standard Time 2016 Row: 10 Page: 10

Overall Description

For this project, we basically followed what Professor Hadi recommended in the class, so we first implemented all the small modules and tested them such as ALU, ClockDivider, RegFile, Muxes, etc. After that, we implemented the overall data-path and the controller. For the controller, we created a spreadsheet listing the opcode and corresponding control signals. We formatted the spreadsheet in such a way that we can copy the content from the spreadsheet to the Verilog file as the case statement. We simulated the processor in ModelSim first and then programmed it to the FPGA once the simulation result seemed correct.

One of the problems that we encountered during our implementation was converting some labels, used before they are declared, into our binary codes. At that moment, we realized that we need to do a pre-running the assembly code to have all values of labels for the case that we encountered. Because our assembler was generating our binary codes as soon as it read each line of the assembly file. Therefore, we implemented a separate function for the pre-running, and it resolves the problem that we encountered.

Another big issue was initializing the memory in ModelSim. Since ModelSim does not take mif files, we had to manually convert the mif file to a mti file and ran the following code in the simulator to load the memory content every time we restarted the simulation.

```
mem load -i //VBOXSVR/CS3220-Group/Proj2/SCProcChenkaiShao/Test2.mem -format mti /SCProc_tb/SCProc/instMem/data
mem load -i //VBOXSVR/CS3220-Group/Proj2/SCProcChenkaiShao/Test2.mem -format mti /SCProc_tb/SCProc/dataMem/data
```

We also had some bugs in the code such as alu_in1, alu_in2, and alu_out should be declared as signed and wires should be declared before used.

Except for the problem above, rest of the problems that we encountered were mostly caused by compatibility issues of our work. When we were combining our work, we noticed that bit-widths of several outputs and types of variables such as signed or unsigned. Even we were implementing Register files in two different ways for one same purpose at some moment. We think that these happened because of lack of communications. Therefore, we agree communicating more frequently and spending more time on designing basic structures of our implementation for our future project.

Contribution

Jeongsoo Kim:

Tested all the single modules and generated waveforms
Implemented ClockDivider and ALU
Complete debugging for the assembler and finalized it.