CprE 381 – Computer Organization and Assembly-Level Programming

Proj-A Report

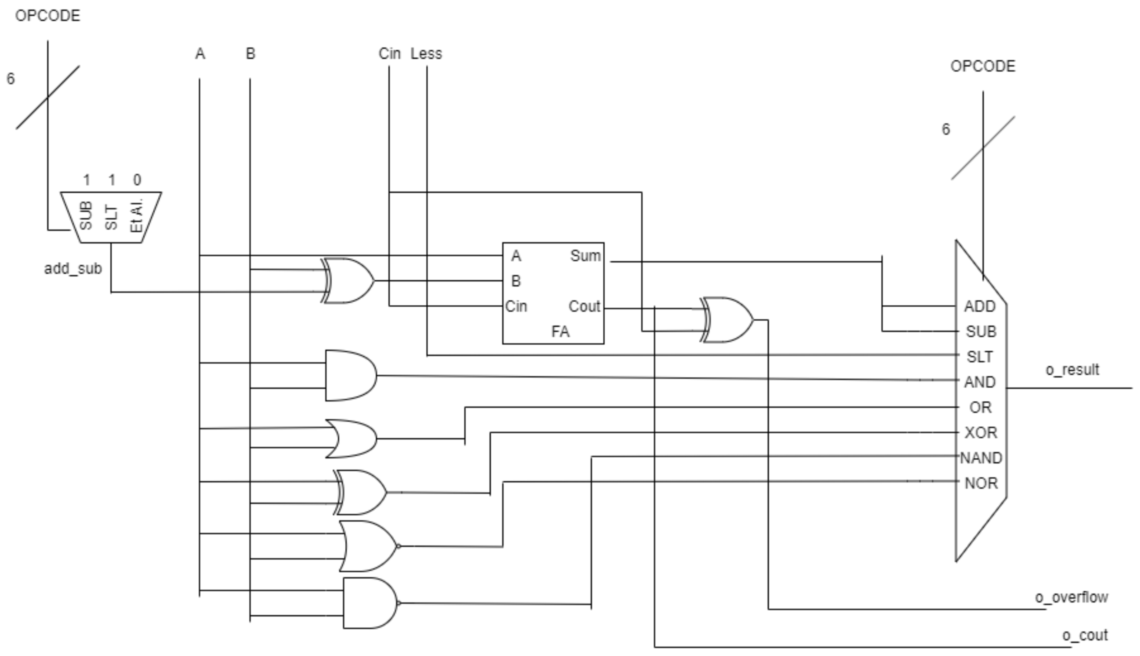
Lab Partners	Colby McKinley
	Tyler Dawson
	Jordan Silvers
Section / Lab Time	Section 5 / W 6:10 p – 8:00

***Submit a typeset pdf version of this on Canvas by the due date. Refer to the highlighted language in the Proj-A instructions for the context of the following questions*.**

a. [Part 0] With your project group members, create a list of best practices / tips for designing, compiling, and testing VHDL modules based on your experiences so far with these labs, both working individually and as a group.

See /prelab.pdf

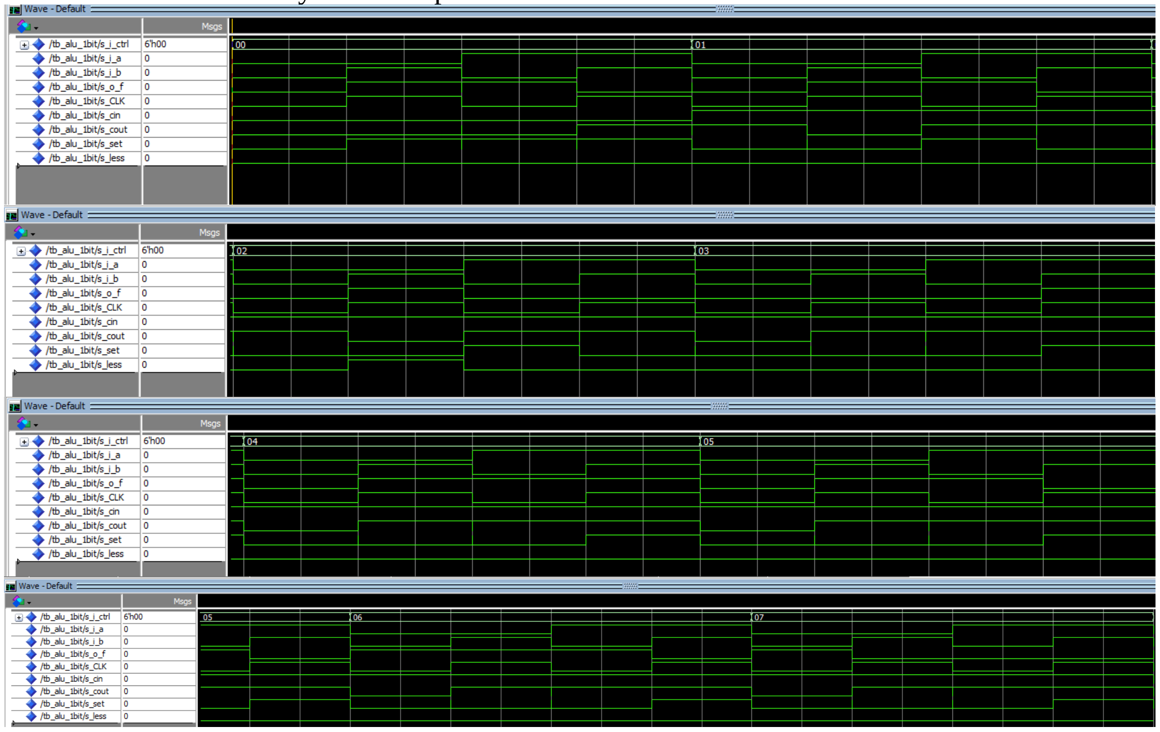
b. [Part 1 (a)] Draw a schematic for a 1-bit ALU that supports the following operations: add/sub (both signed and unsigned), slt, and, or, xor, nand, and nor. What are the inputs and outputs that are needed?



c. [Part 1 (b)] In your project writeup, describe your design in terms of the VHDL coding style you chose and the control signals that are required.

I used the opcode to create a secondary control signal called add_sub, which determines if the full adder will be used to subtract or to add. The operations sub and slt will use a subtracting result. The add_sub signal is used to either negate or preserve b, which is fed into the full adder. The set signal is driven by xoring the overflow value with the sum from the full adder. I think having two control signals, opcode and funct will be required going forward to handle all the ALU operations.

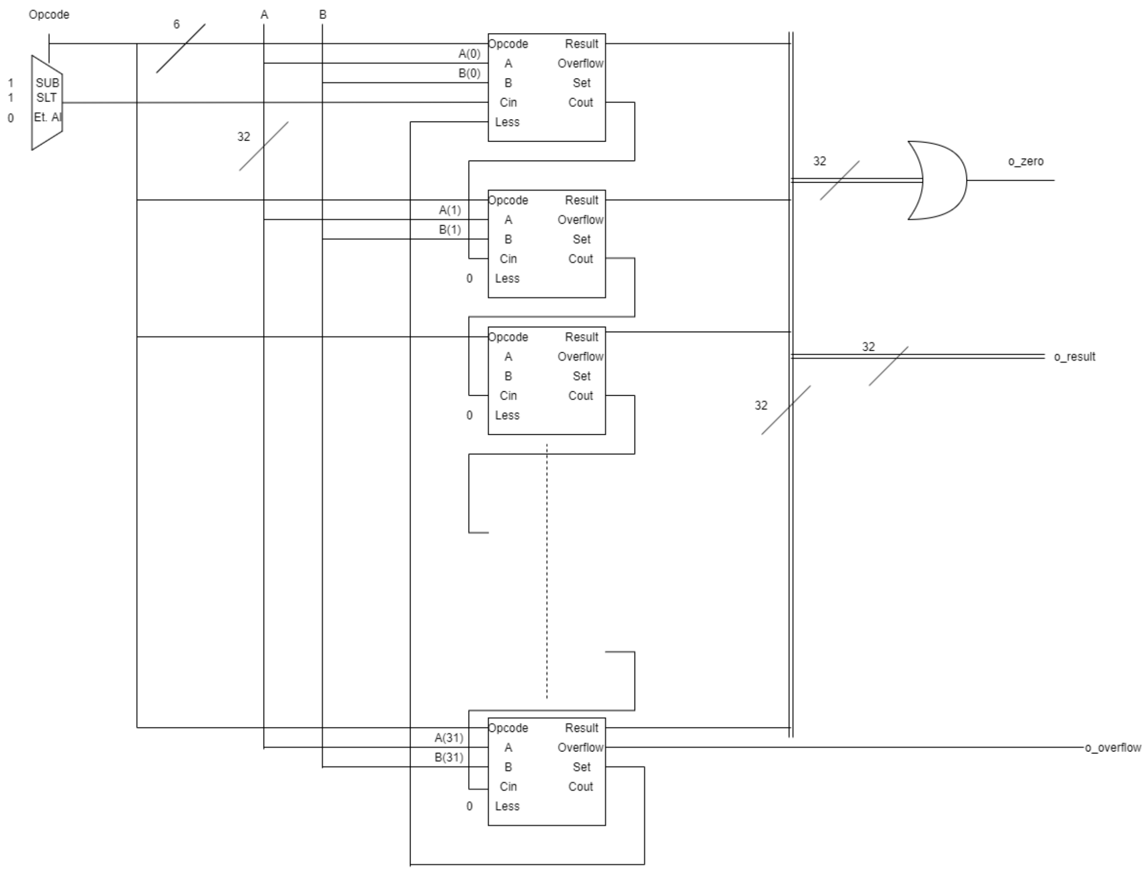
d. [Part 1 (c)] Describe how the execution of the different operations corresponds to the Modelsim waveforms in your writeup.



```
Wave - Default
                         Msgs
/tb_alu_1bit/s_i_ctrl    6'h00     00 | 01
/tb_alu_1bit/s_i_a       0
/tb_alu_1bit/s_i_b       0
/tb_alu_1bit/s_o_f       0
/tb_alu_1bit/s_CLK       0
/tb_alu_1bit/s_cin       0
/tb_alu_1bit/s_cout      0
/tb_alu_1bit/s_set       0
/tb_alu_1bit/s_less      0

Wave - Default
                         Msgs
/tb_alu_1bit/s_i_ctrl    6'h00     02 | 03
/tb_alu_1bit/s_i_a       0
/tb_alu_1bit/s_i_b       0
/tb_alu_1bit/s_o_f       0
/tb_alu_1bit/s_CLK       0
/tb_alu_1bit/s_cin       0
/tb_alu_1bit/s_cout      0
/tb_alu_1bit/s_set       0
/tb_alu_1bit/s_less      0

Wave - Default
                         Msgs
/tb_alu_1bit/s_i_ctrl    6'h00     04 | 05
/tb_alu_1bit/s_i_a       0
/tb_alu_1bit/s_i_b       0
/tb_alu_1bit/s_o_f       0
/tb_alu_1bit/s_CLK       0
/tb_alu_1bit/s_cin       0
/tb_alu_1bit/s_cout      0
/tb_alu_1bit/s_set       0
/tb_alu_1bit/s_less      0

Wave - Default
                         Msgs
/tb_alu_1bit/s_i_ctrl    6'h00     05 | 06 | 07
/tb_alu_1bit/s_i_a       0
/tb_alu_1bit/s_i_b       0
/tb_alu_1bit/s_o_f       0
/tb_alu_1bit/s_CLK       0
/tb_alu_1bit/s_cin       0
/tb_alu_1bit/s_cout      0
/tb_alu_1bit/s_set       0
/tb_alu_1bit/s_less      0
```

e. [Part 2 (a)] Draw a simplified schematic for this 32-bit ALU. Consider the following questions: how is Overflow calculated? How is Zero calculated? How is slt implemented?

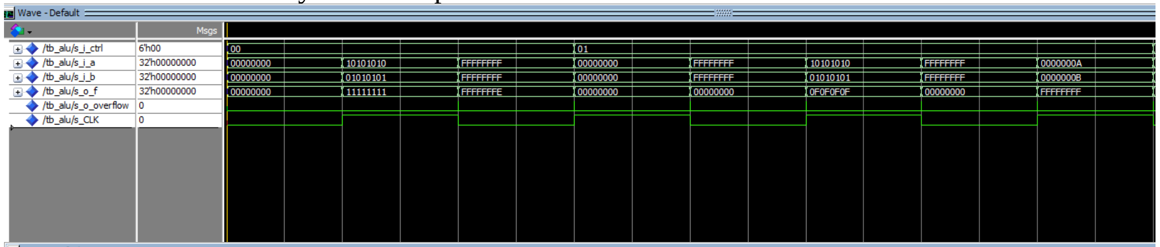
Overflow is calculated by xoring the final carry out with the final carry in. Zero is calculated by xoring the resulting 32 bit to see if any individual one is turn on. SLT is computed by xoring the final overflow with the final sum value of A - B.



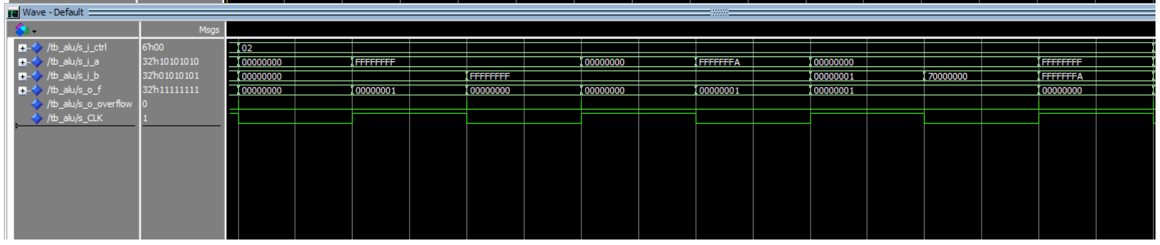
f. [Part 2 (b)] In your writeup, describe what challenges (if any) you faced in implementing this module.

Instruction which required subtracting did not work, i.e. SUB and SLT. After redesigning how subtracting work in the alu, these instructions worked fine.

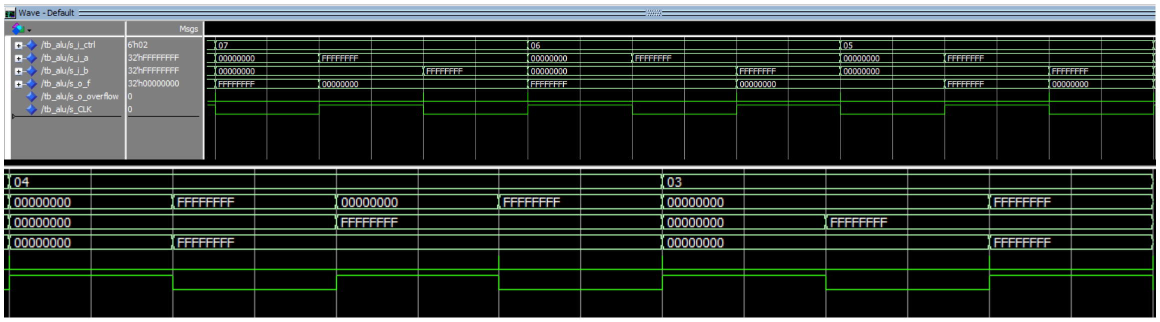
g. [Part 2 (c)] Describe how the execution of the different operations corresponds to the Modelsim waveforms in your writeup.



```
Wave - Default
                       Msgs
/tb_alu/s_i_ctrl       6'h00         00                                                  01
/tb_alu/s_i_a          32'h00000000  00000000  10101010  FFFFFFFF  00000000  FFFFFFFF  10101010  FFFFFFFF  0000000A
/tb_alu/s_i_b          32'h00000000  00000000  01010101  FFFFFFFF  00000000  FFFFFFFF  01010101  FFFFFFFF  0000000B
/tb_alu/s_o_f          32'h00000000  00000000  11111111  FFFFFFFE  00000000  00000000  0F0F0F0F  00000000  FFFFFFFF
/tb_alu/s_o_overflow   0
/tb_alu/s_CLK          0
```



```
Wave - Default
                       Msgs
/tb_alu/s_i_ctrl       6'h00         02
/tb_alu/s_i_a          32'h10101010  00000000  FFFFFFFF            00000000  FFFFFFFA  00000000            FFFFFFFF
/tb_alu/s_i_b          32'h01010101  00000000            FFFFFFFF                      00000001  70000000  FFFFFFFA
/tb_alu/s_o_f          32'h11111111  00000000  00000001  00000000  00000000  00000001  00000001            00000000
/tb_alu/s_o_overflow   0
/tb_alu/s_CLK          1
```



```
Wave - Default
                              Msgs
/tb_alu/s_i_ctrl        6'h02
/tb_alu/s_i_a           32'hFFFFFFFF
/tb_alu/s_i_b           32'hFFFFFFFF
/tb_alu/s_o_f           32'h00000000
/tb_alu/s_o_overflow    0
/tb_alu/s_CLK           0
```

h. [Part 3 (a)] Describe the difference between logical (srl) and arithmetic (sra) shifts. Why does MIPS not have a sla instruction?

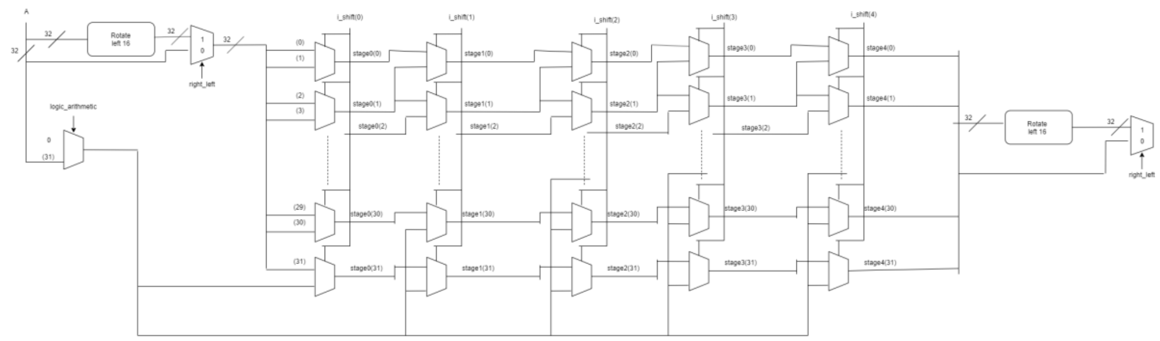
SRL – Shift Right Logical, will shift each bit in binary number a binary number a specified amount, x, of bits right. It loads in 0's into the most significant x bits. This instruction is synonymous with dividing an unsigned number by 2^x.

SLL – Shift Left Logical, will shift each bit in binary number a binary number a specified amount, x, of bits left. It loads in 0's into the least significant x bits. This instruction is synonymous with multiplying a number by 2^x.

SRA – Shift Right Arithmetic, will shift each bit in binary number a binary number a specified amount, x, of bits right. It loads in 1's into the most significant x bits. This instruction is synonymous with dividing a signed number by 2^x.

SLA – Shift Left Arithmetic is not supported because it is the same as SLL.

i. [Part 3 (b)] In your writeup, briefly describe how your VHDL code implements both the arithmetic and logical shifting operations.

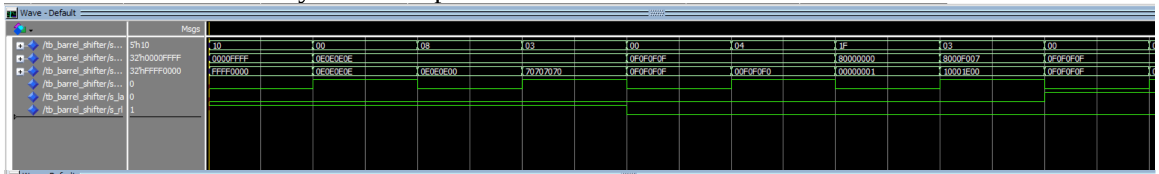


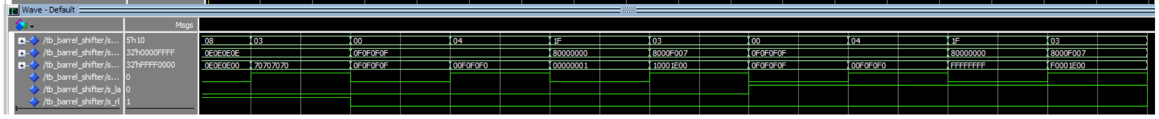
The VHDL code implements a right shifter, which accounts for logic and arithmetic shifts. There is a multiplexer which chooses the value, s_la, which is shifts into the new shifted value. If the instruction is logical, then 0 flows to s_la, otherwise the most significant bit of the base number flows into s_la. The mux tree then uses s_la to flow into the shifted value.

j. [Part 3 (c)] In your writeup, explain how the right barrel shifter from part b) can be enhanced to also support left shifting operations.

If the instruction was to shift left, the base value bits are flipped with its ‘inverse’ bit. That is the value at bit 31 is swapped with the value at bit 0, bit 30 is swapped with bit 1, for all bits 0 to 31.

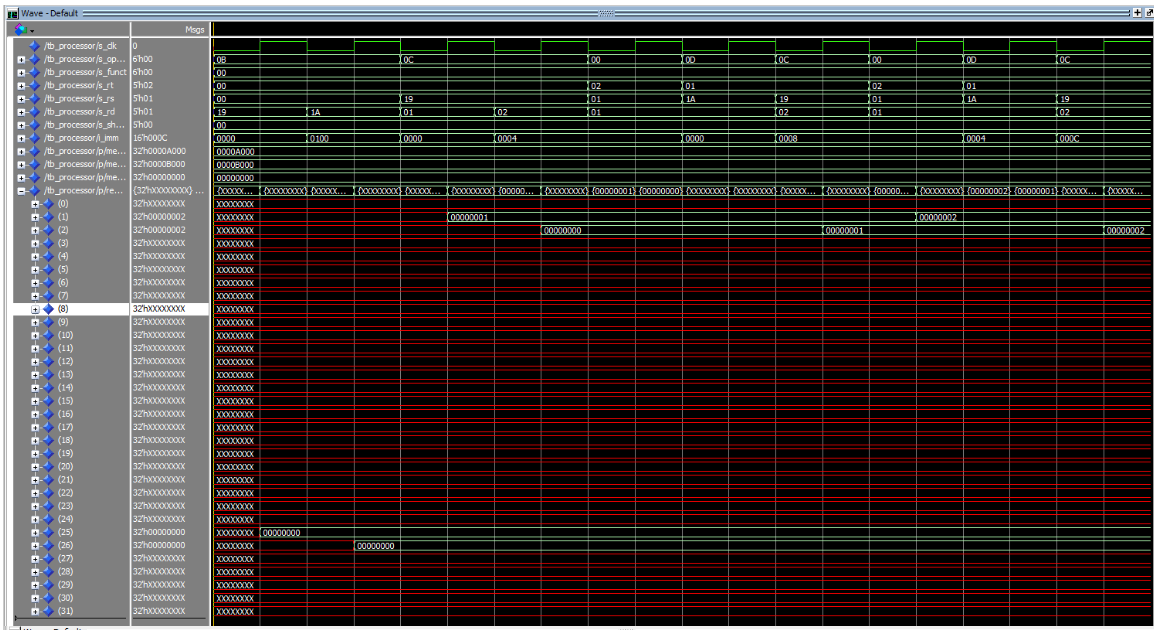
k. [Part 3 (d)] Describe how the execution of the different shifting operations corresponds to the Modelsim waveforms in your writeup.


Wave - Default
Msgs
/tb_barrel_shifter/s... 5'h10
/tb_barrel_shifter/s... 32'h0000FFFF
/tb_barrel_shifter/s... 32'hFFFF0000
/tb_barrel_shifter/s... 0
/tb_barrel_shifter/s_la 0
/tb_barrel_shifter/s_rl 1
10 00 08 03 00 04 1F 03 00
0000FFFF 0E0E0E0E 0F0F0F0F 80000000 8000F007 0F0F0F0F
FFFF0000 0E0E0E0E 0E0E0E00 70707070 0F0F0F0F 00F0F0F0 00000001 10001E00 0F0F0F0F

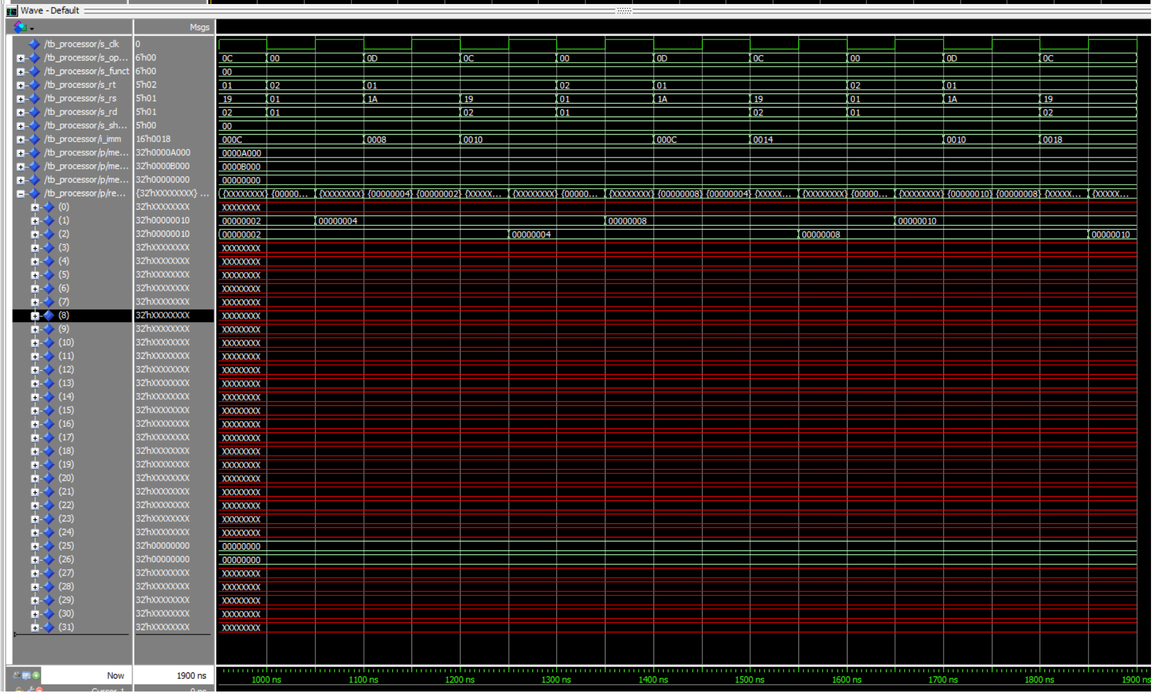

Wave - Default
Msgs
/tb_barrel_shifter/s... 5'h10
/tb_barrel_shifter/s... 32'h0000FFFF
/tb_barrel_shifter/s... 32'hFFFF0000
/tb_barrel_shifter/s... 0
/tb_barrel_shifter/s_la 0
/tb_barrel_shifter/s_rl 1
08 03 00 04 1F 03 00 04 1F 03
0E0E0E0E 0F0F0F0F 80000000 8000F007 0F0F0F0F 80000000 8000F007
0E0E0E00 70707070 0F0F0F0F 00F0F0F0 00000001 10001E00 0F0F0F0F 00F0F0F0 FFFFFFFF F0001E00

l. [Part 4(b)] Justify why your test plan is comprehensive. Include waveforms that demonstrate your test program is functioning.

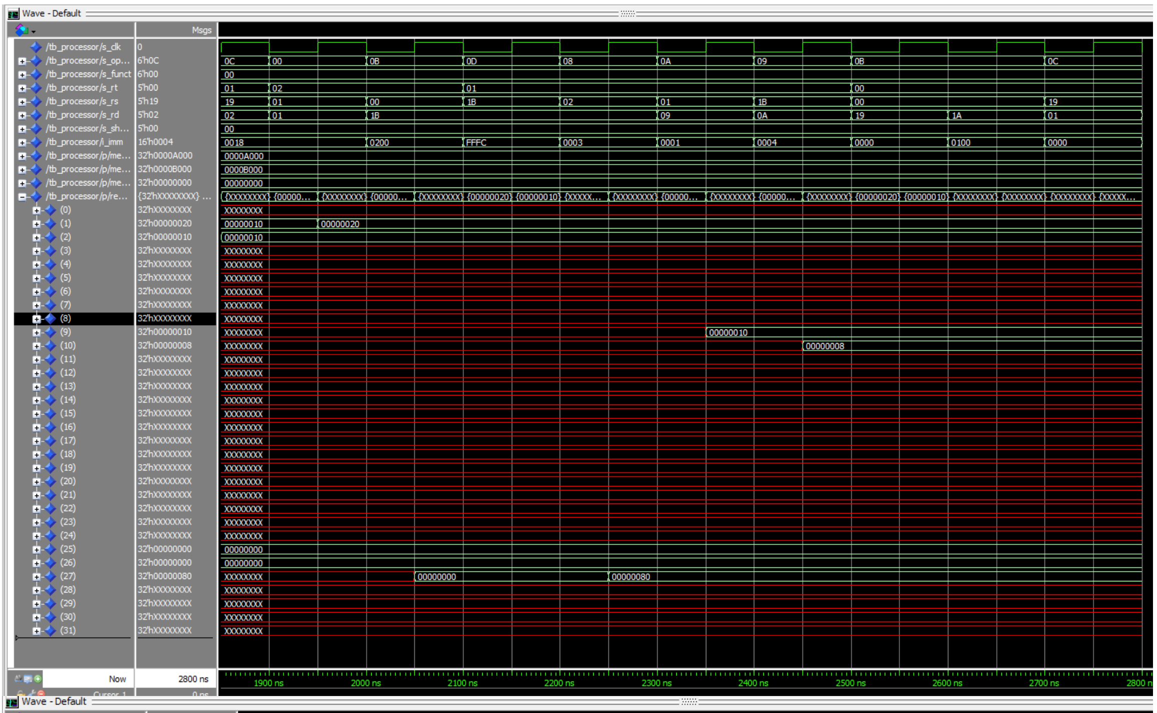
Up to this point, every function has been tested thoroughly and works. This test is meant to confirm that the ALU interfaces with the processor designed in lab 4.



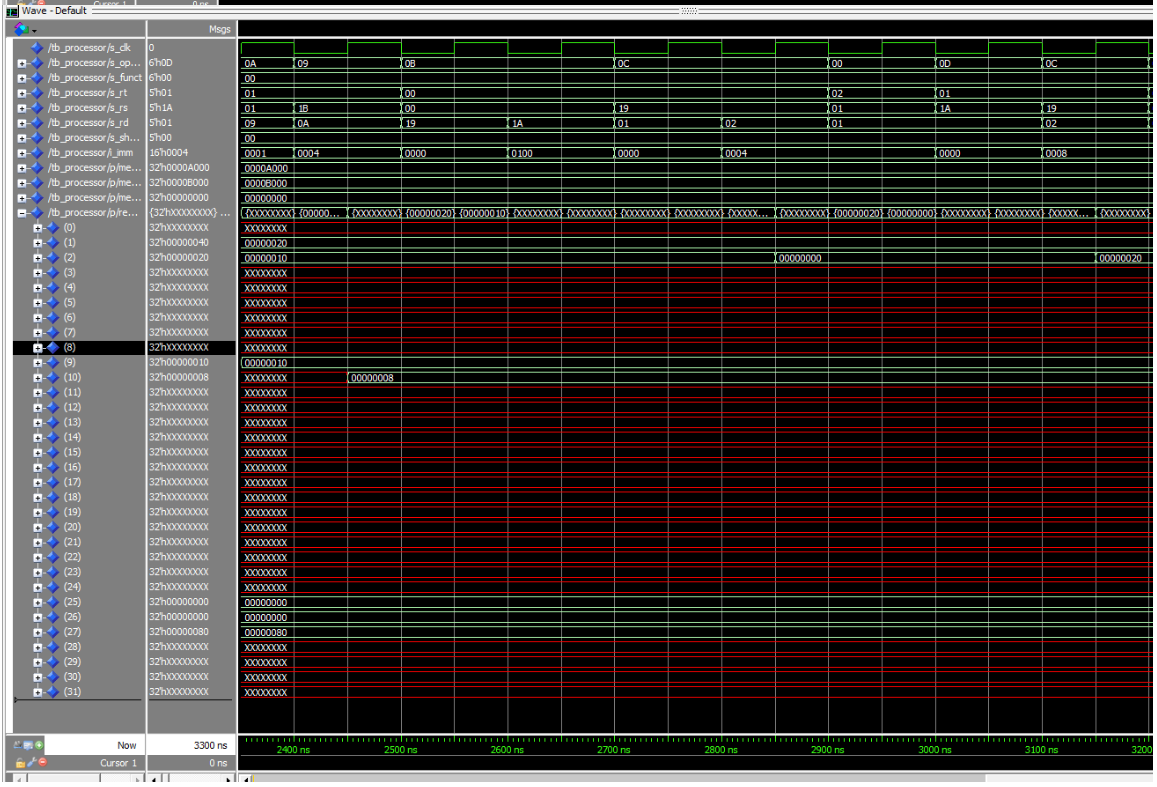
```
Wave - Default
                          Msgs
/tb_processor/s_clk       0
/tb_processor/s_op...     6'h00              0B  0C  00  0D  0C  00  0D  0C
/tb_processor/s_funct     6'h00              00
/tb_processor/s_rt        5'h02              00  02  01  02  01
/tb_processor/s_rs        5'h01              00  19  01  1A  19  01  1A  19
/tb_processor/s_rd        5'h01              19  1A  01  02  01  02  01  02
/tb_processor/s_sh...     5'h00              00
/tb_processor/i_imm       16'h000C           0000  0100  0000  0004  0000  0008  0004  000C
/tb_processor/p/me...     32'h0000A000       0000A000
/tb_processor/p/me...     32'h0000B000       0000B000
/tb_processor/p/me...     32'h00000000       00000000
/tb_processor/p/re...     {32'hXXXXXXXX} ... {XXXXX... {XXXXXXXX} {XXXXX... {XXXXXXXX} {XXXXX... {XXXXXXXX} {00000... {XXXXXXXX} {00000001} {00000000} {XXXXXXXX} {XXXXXXXX} {XXXXX... {XXXXXXXX} {00000... {XXXXXXXX} {00000002} {00000001} {XXXXX... {XXXXX...
  (0)                     32'hXXXXXXXX       XXXXXXXX
  (1)                     32'h00000002       XXXXXXXX  00000001  00000002
  (2)                     32'h00000002       XXXXXXXX  00000000  00000001  00000002
  (3)                     32'hXXXXXXXX       XXXXXXXX
  (4)                     32'hXXXXXXXX       XXXXXXXX
  (5)                     32'hXXXXXXXX       XXXXXXXX
  (6)                     32'hXXXXXXXX       XXXXXXXX
  (7)                     32'hXXXXXXXX       XXXXXXXX
  (8)                     32'hXXXXXXXX       XXXXXXXX
  (9)                     32'hXXXXXXXX       XXXXXXXX
  (10)                    32'hXXXXXXXX       XXXXXXXX
  (11)                    32'hXXXXXXXX       XXXXXXXX
  (12)                    32'hXXXXXXXX       XXXXXXXX
  (13)                    32'hXXXXXXXX       XXXXXXXX
  (14)                    32'hXXXXXXXX       XXXXXXXX
  (15)                    32'hXXXXXXXX       XXXXXXXX
  (16)                    32'hXXXXXXXX       XXXXXXXX
  (17)                    32'hXXXXXXXX       XXXXXXXX
  (18)                    32'hXXXXXXXX       XXXXXXXX
  (19)                    32'hXXXXXXXX       XXXXXXXX
  (20)                    32'hXXXXXXXX       XXXXXXXX
  (21)                    32'hXXXXXXXX       XXXXXXXX
  (22)                    32'hXXXXXXXX       XXXXXXXX
  (23)                    32'hXXXXXXXX       XXXXXXXX
  (24)                    32'hXXXXXXXX       XXXXXXXX
  (25)                    32'h00000000       XXXXXXXX  00000000
  (26)                    32'h00000000       XXXXXXXX  00000000
  (27)                    32'hXXXXXXXX       XXXXXXXX
  (28)                    32'hXXXXXXXX       XXXXXXXX
  (29)                    32'hXXXXXXXX       XXXXXXXX
  (30)                    32'hXXXXXXXX       XXXXXXXX
  (31)                    32'hXXXXXXXX       XXXXXXXX
```



```
Wave - Default
                          Msgs
/tb_processor/s_clk       0
/tb_processor/s_op...     6'h00              0C  00  0D  0C  00  0D  0C  00  0D  0C
/tb_processor/s_funct     6'h00              00
/tb_processor/s_rt        5'h02              01  02  01  02  01  02  01
/tb_processor/s_rs        5'h01              19  01  1A  19  01  1A  19  01  1A  19
/tb_processor/s_rd        5'h01              02  01  02  01  02  01  02
/tb_processor/s_sh...     5'h00              00
/tb_processor/i_imm       16'h0018           000C  0008  0010  000C  0014  0010  0018
/tb_processor/p/me...     32'h0000A000       0000A000
/tb_processor/p/me...     32'h0000B000       0000B000
/tb_processor/p/me...     32'h00000000       00000000
/tb_processor/p/re...     {32'hXXXXXXXX} ... {XXXXXXXX} {00000... {XXXXXXXX} {00000004} {00000002} {XXXXX... {XXXXXXXX} {00000... {XXXXXXXX} {00000008} {00000004} {XXXXX... {XXXXXXXX} {00000... {XXXXXXXX} {00000010} {00000008} {XXXXX... {XXXXX...
  (0)                     32'hXXXXXXXX       XXXXXXXX
  (1)                     32'h00000010       00000002  00000004  00000008  00000010
  (2)                     32'h00000010       00000002  00000004  00000008  00000010
  (3)                     32'hXXXXXXXX       XXXXXXXX
  (4)                     32'hXXXXXXXX       XXXXXXXX
  (5)                     32'hXXXXXXXX       XXXXXXXX
  (6)                     32'hXXXXXXXX       XXXXXXXX
  (7)                     32'hXXXXXXXX       XXXXXXXX
  (8)                     32'hXXXXXXXX       XXXXXXXX
  (9)                     32'hXXXXXXXX       XXXXXXXX
  (10)                    32'hXXXXXXXX       XXXXXXXX
  (11)                    32'hXXXXXXXX       XXXXXXXX
  (12)                    32'hXXXXXXXX       XXXXXXXX
  (13)                    32'hXXXXXXXX       XXXXXXXX
  (14)                    32'hXXXXXXXX       XXXXXXXX
  (15)                    32'hXXXXXXXX       XXXXXXXX
  (16)                    32'hXXXXXXXX       XXXXXXXX
  (17)                    32'hXXXXXXXX       XXXXXXXX
  (18)                    32'hXXXXXXXX       XXXXXXXX
  (19)                    32'hXXXXXXXX       XXXXXXXX
  (20)                    32'hXXXXXXXX       XXXXXXXX
  (21)                    32'hXXXXXXXX       XXXXXXXX
  (22)                    32'hXXXXXXXX       XXXXXXXX
  (23)                    32'hXXXXXXXX       XXXXXXXX
  (24)                    32'hXXXXXXXX       XXXXXXXX
  (25)                    32'h00000000       00000000
  (26)                    32'h00000000       00000000
  (27)                    32'hXXXXXXXX       XXXXXXXX
  (28)                    32'hXXXXXXXX       XXXXXXXX
  (29)                    32'hXXXXXXXX       XXXXXXXX
  (30)                    32'hXXXXXXXX       XXXXXXXX
  (31)                    32'hXXXXXXXX       XXXXXXXX

Now                       1900 ns      1000 ns  1100 ns  1200 ns  1300 ns  1400 ns  1500 ns  1600 ns  1700 ns  1800 ns  1900 ns
```


Wave - Default

	Msgs
/tb_processor/s_clk	0
/tb_processor/s_op...	6'h0C
/tb_processor/s_funct	6'h00
/tb_processor/s_rt	5'h00
/tb_processor/s_rs	5'h19
/tb_processor/s_rd	5'h02
/tb_processor/s_sh...	5'h00
/tb_processor/i_imm	16'h0004
/tb_processor/p/me...	32'h0000A000
/tb_processor/p/me...	32'h0000B000
/tb_processor/p/me...	32'h00000000
/tb_processor/p/re...	{32'hXXXXXXXX} ...
(0)	32'hXXXXXXXX
(1)	32'h00000020
(2)	32'h00000010
(3)	32'hXXXXXXXX
(4)	32'hXXXXXXXX
(5)	32'hXXXXXXXX
(6)	32'hXXXXXXXX
(7)	32'hXXXXXXXX
(8)	32'hXXXXXXXX
(9)	32'h00000010
(10)	32'h00000008
(11)	32'hXXXXXXXX
(12)	32'hXXXXXXXX
(13)	32'hXXXXXXXX
(14)	32'hXXXXXXXX
(15)	32'hXXXXXXXX
(16)	32'hXXXXXXXX
(17)	32'hXXXXXXXX
(18)	32'hXXXXXXXX
(19)	32'hXXXXXXXX
(20)	32'hXXXXXXXX
(21)	32'hXXXXXXXX
(22)	32'hXXXXXXXX
(23)	32'hXXXXXXXX
(24)	32'hXXXXXXXX
(25)	32'h00000000
(26)	32'h00000000
(27)	32'h00000080
(28)	32'hXXXXXXXX
(29)	32'hXXXXXXXX
(30)	32'hXXXXXXXX
(31)	32'hXXXXXXXX
Now	2800 ns


Wave - Default

	Msgs
/tb_processor/s_clk	0
/tb_processor/s_op...	6'h0D
/tb_processor/s_funct	6'h00
/tb_processor/s_rt	5'h01
/tb_processor/s_rs	5'h1A
/tb_processor/s_rd	5'h01
/tb_processor/s_sh...	5'h00
/tb_processor/i_imm	16'h0004
/tb_processor/p/me...	32'h0000A000
/tb_processor/p/me...	32'h0000B000
/tb_processor/p/me...	32'h00000000
/tb_processor/p/re...	{32'hXXXXXXXX} ...
(0)	32'hXXXXXXXX
(1)	32'h00000040
(2)	32'h00000020
(3)	32'hXXXXXXXX
(4)	32'hXXXXXXXX
(5)	32'hXXXXXXXX
(6)	32'hXXXXXXXX
(7)	32'hXXXXXXXX
(8)	32'hXXXXXXXX
(9)	32'h00000010
(10)	32'h00000008
(11)	32'hXXXXXXXX
(12)	32'hXXXXXXXX
(13)	32'hXXXXXXXX
(14)	32'hXXXXXXXX
(15)	32'hXXXXXXXX
(16)	32'hXXXXXXXX
(17)	32'hXXXXXXXX
(18)	32'hXXXXXXXX
(19)	32'hXXXXXXXX
(20)	32'hXXXXXXXX
(21)	32'hXXXXXXXX
(22)	32'hXXXXXXXX
(23)	32'hXXXXXXXX
(24)	32'hXXXXXXXX
(25)	32'h00000000
(26)	32'h00000000
(27)	32'h00000080
(28)	32'hXXXXXXXX
(29)	32'hXXXXXXXX
(30)	32'hXXXXXXXX
(31)	32'hXXXXXXXX
Now	3300 ns
Cursor 1	0 ns

m. [Part 5(c) BONUS] Justify why your test plan is comprehensive. Include waveforms that demonstrate your test program is functioning.

I used the testing program from Lab04. I then added tests for the shifting instructions since these were the last ones left untested from the base.

n. [Feedback] You must complete this section for your lab to be graded. Please complete each column **separately** for each team member; I expect it to take roughly 10 minutes (do not take more than 20 minutes).

i. How many hours did you spend on this lab?

Task	**During lab time**			**Outside of lab time**		
Team Initials	CM	TD	JS	CM	TD	JS
Reading lab	15 min	15 min	15 min	0	0	0
Pencil/paper design	1 hr	0	1 hr	30 min	0	1 hr
VHDL design	1 hr	2 hr	1 hr	6 hr	3 hr	3 hr
Assembly coding	0	0	0	0	0	0
Simulation	0	0	1 hr	1 hr	0	1 hr
Debugging	2 hr	2 hr	1 hr	5 hr	3 hr	1 hr
Report writing	0	0	0	2 hr	0	0
Other:						
Total	4 hr 15 min	4 hr 15 min	4 hr 15 min	14.5 hr	6 hr	6 hr

ii. If you could change one thing about the lab experience, what would it be? Why?

It would be helpful if I was given the components input and output pins for the given final project. Even if the pins are not going to be used this week, understanding how MIPS handles these would help drive my final design.

iii. What was the most interesting part of the lab?

Getting the single bit ALU to work concurrently with 31 other single bit ALUs was interesting design choice.