CprE 381 – Computer Organization and Assembly-Level Programming

Proj-A Report

Lab Partners Max Solaro ___________________________________

Yuin Ng _____________________________________

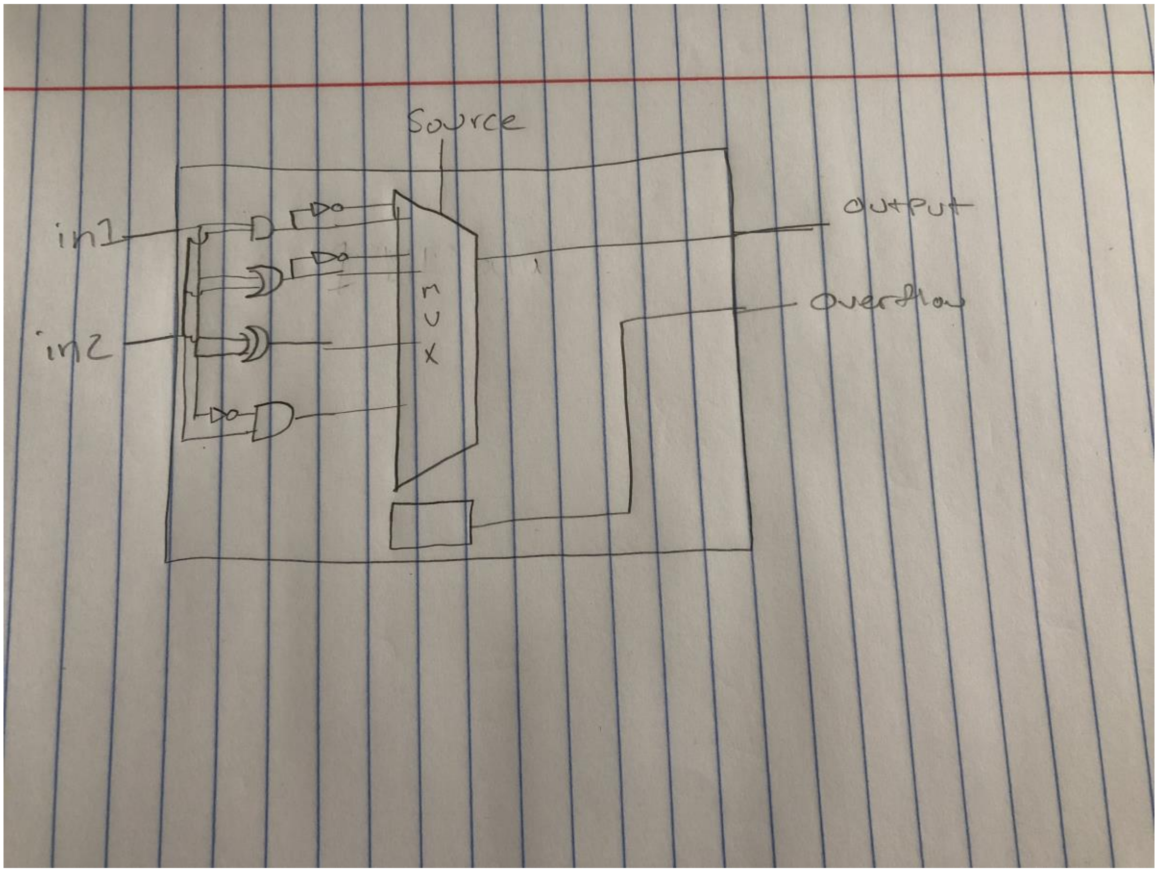
____

Section / Lab Time Wednesdays 4___________________

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.
Test benches are good for testing expected outputs.
For conceptualizing microarchitecture drawing out the schematics is helpful.

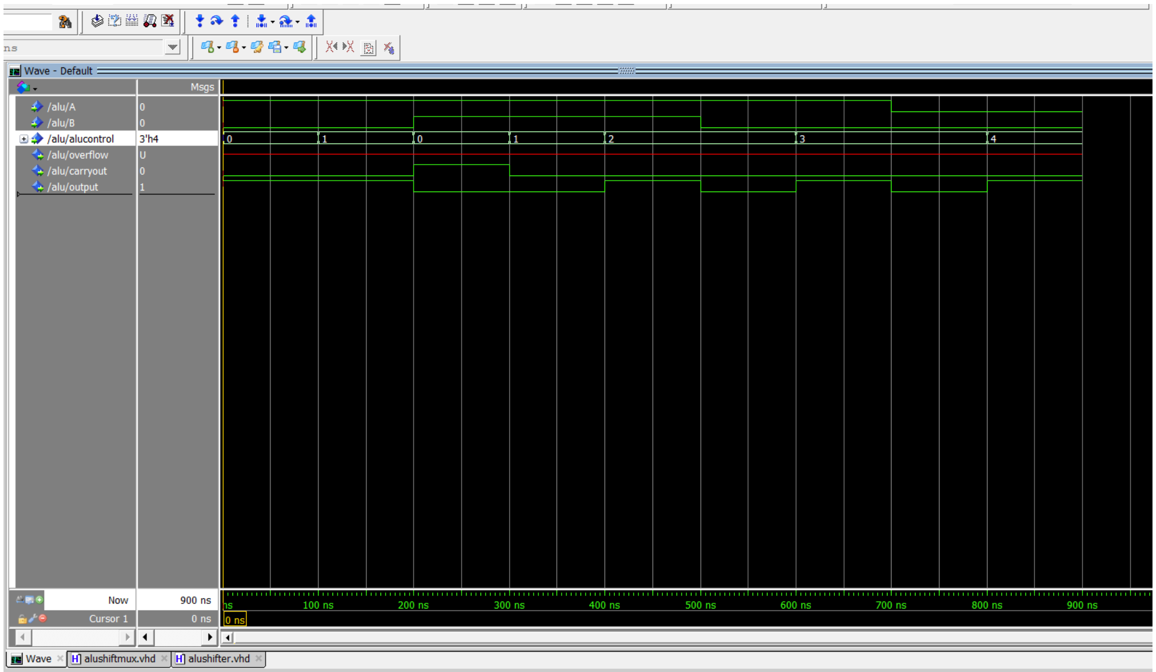
b. [Part 1 (a)] Draw a schematic for a 1-bit ALU that supports the following operations: add/sub (both signed and unsigned), slt, and, or, xor, nand, and nor. What are the inputs and outputs that are needed?



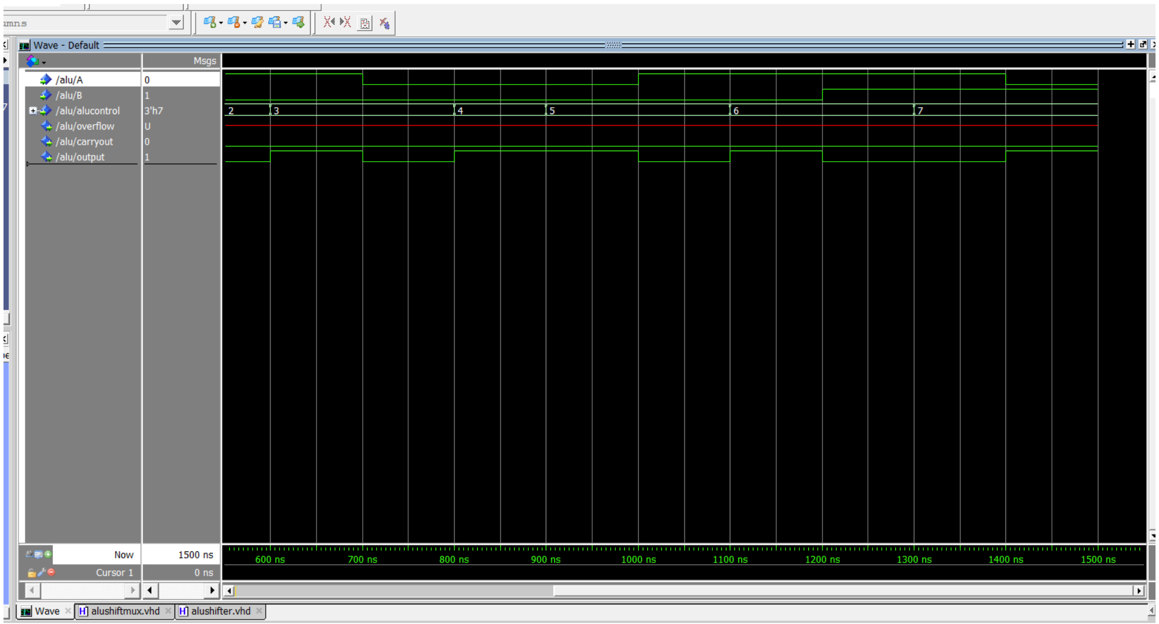
c. [Part 1 (b)] In your project writeup, describe your design in terms of the VHDL coding style you chose and the control signals that are required.

We did a dataflow style vhdl. The alu needs 2 1 bit input signals, a 4 bit control signal and a one bit output. For the add and subtract we will need a carryout and an overflow.

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



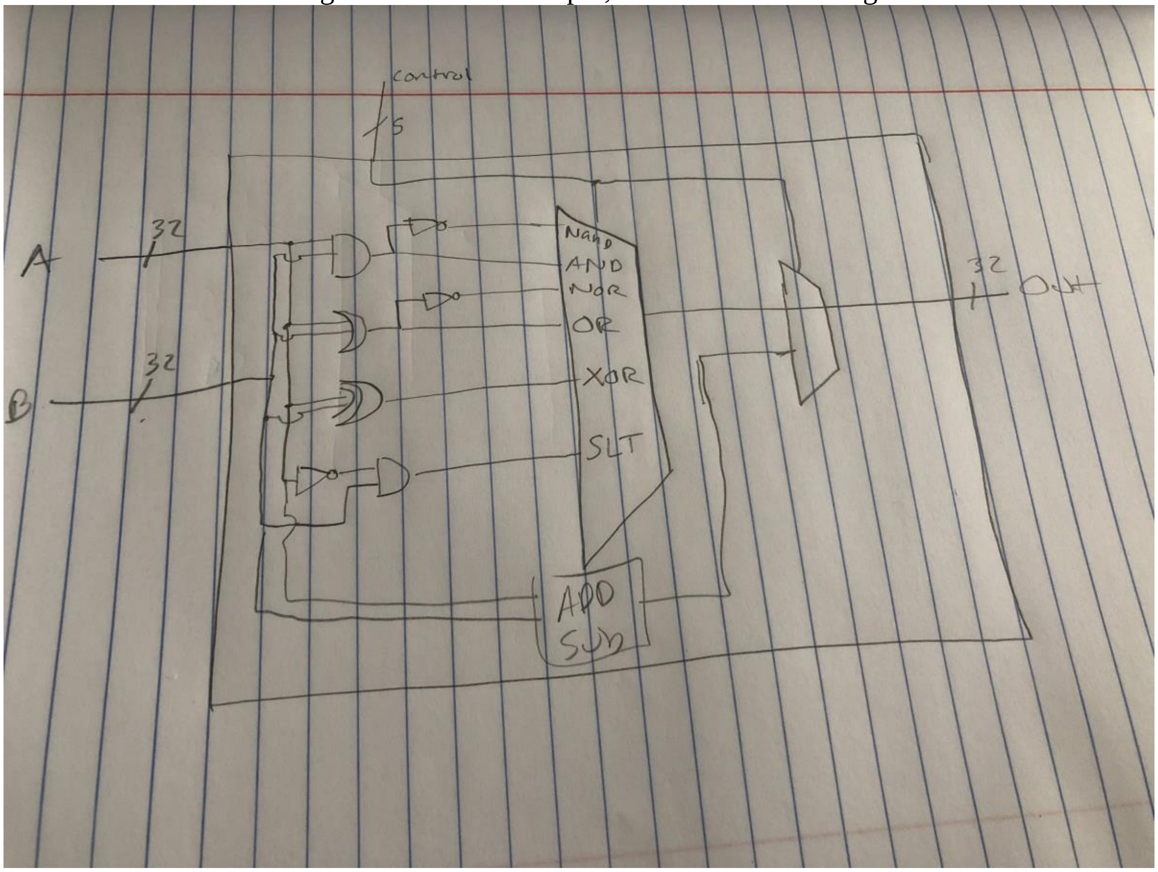
Commands: “000” ➔ add shown
“001” ➔ sub shown
“010” ➔ and shown
“011” ➔ or shown
“100” ➔ nand shown
“101” ➔ nor
“110” ➔ xor
“111” ➔ slt



This wave form shows the nor xor and slt.

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?

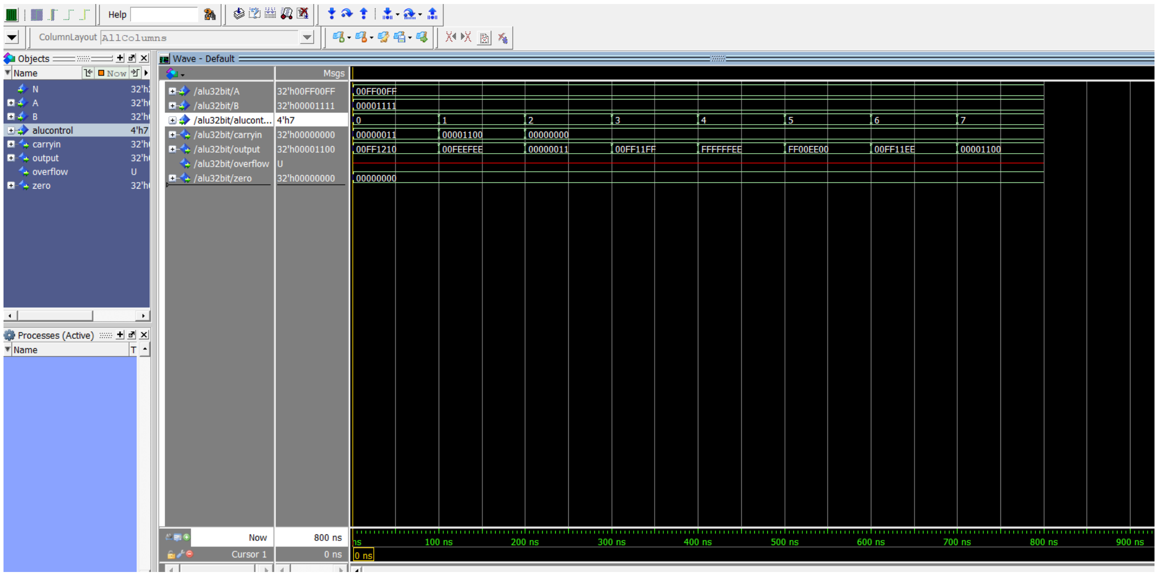
Zero can be calculated by hard wiring an input, branching it, send one end through a not gate, then both ends into an and gate. Or branch an input, send both ends through an xor.



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

The biggest challenge we face was adding and subtracting two logic vectors. We didn't end up implementing a previously built add/sub unit and instead wrote it using dataflow.

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



Name	Now
N	32'h
A	32'h
B	32'h
alucontrol	4'h7
carryin	32'h
output	32'h
overflow	U
zero	32'h

	Msgs								
/alu32bit/A	32'h00FF00FF	00FF00FF							
/alu32bit/B	32'h00001111	00001111							
/alu32bit/alucont...	4'h7	0	1	2	3	4	5	6	7
/alu32bit/carryin	32'h00000000	00000011	00001100	00000000					
/alu32bit/output	32'h00001100	00FF1210	00FEEFEE	00000011	00FF11FF	FFFFFFEE	FF00EE00	00FF11EE	00001100
/alu32bit/overflow	U								
/alu32bit/zero	32'h00000000	00000000							

Now	800 ns
Cursor 1	0 ns

0 ns

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

Shift right logical and shift right arithmetic are very similar. The only difference is if you are doing an arithmetic shift, you must consider the sign of the number. This is determined by the most significant bit, if it is a 1, then the number is negative and must be extended by its sign. Shift left logical does not have a corresponding shift left arithmetic because regardless of the most significant bit, the shift will be the same.

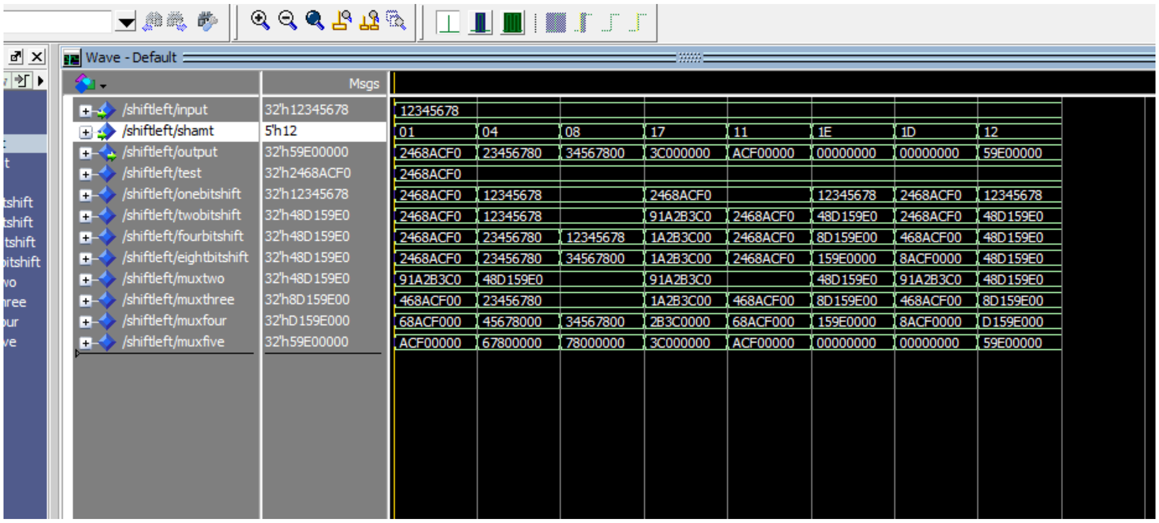
i. [Part 3 (b)] In your writeup, briefly describe how your VHDL code implements both the arithmetic and logical shifting operations.

All of our shifting is done very similarly. We implemented something similar to a barrel shifter. This means we have 5 muxes, each mux takes 2 32bit inputs. The first mux shifts one input by 1 bit and leaves the other the same. Then according to the 5 bit shamt input, it outputs either the shifted input or the original. The second mux acts almost the same way but it shifts one input by 2. The third mux shifts by 4 and the fourth by 8. Lastly the 5th mux shifts one input by 16. When dealing with arithmetic vs logical right shifting, we check the most significant bit and extend the shifted value by that bit.

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.

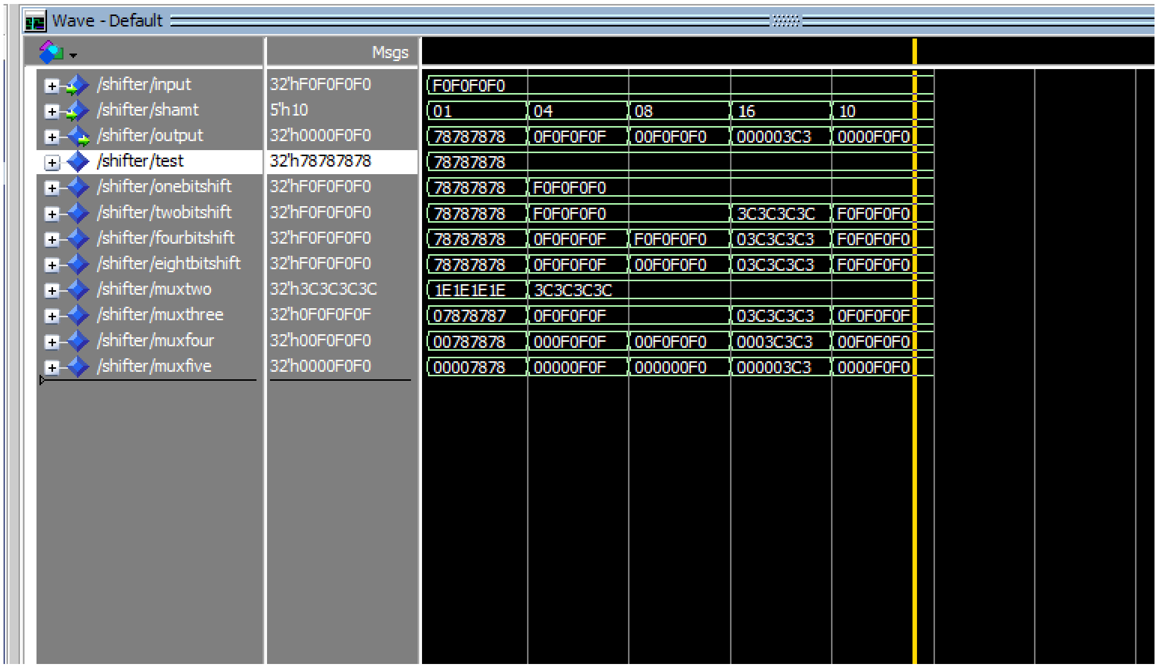
The switch between right shifting and left shifting is trivial. Instead of cutting off the beginning and filling in at the end, you just cut off at the end and fill in at the beginning.

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


Wave - Default

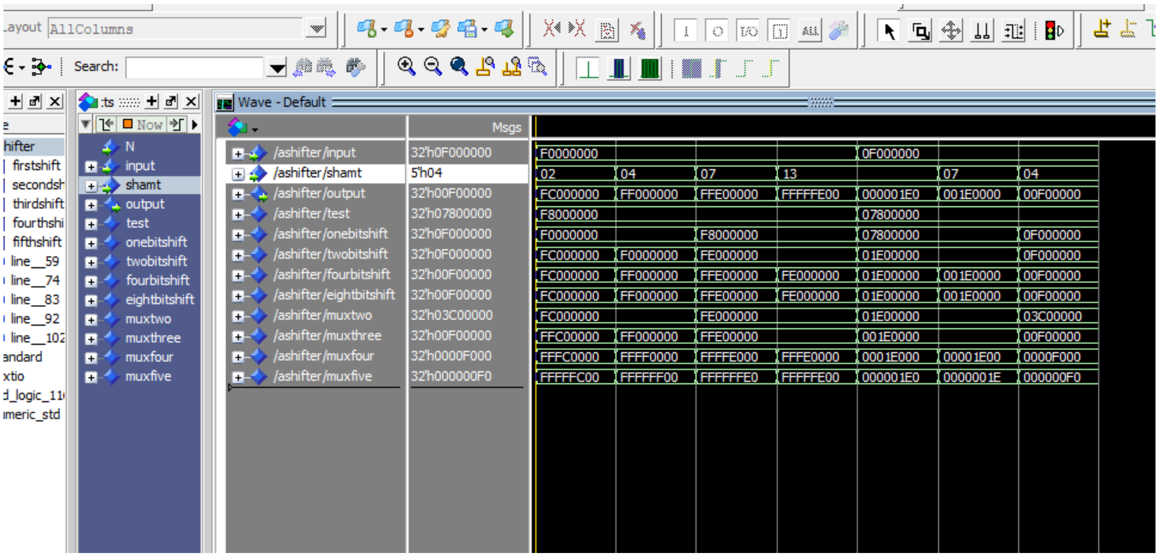
	Msgs								
/shiftleft/input	32'h12345678	12345678							
/shiftleft/shamt	5'h12	01	04	08	17	11	1E	1D	12
/shiftleft/output	32'h59E00000	2468ACF0	23456780	34567800	3C000000	ACF00000	00000000	00000000	59E00000
/shiftleft/test	32'h2468ACF0	2468ACF0							
/shiftleft/onebitshift	32'h12345678	2468ACF0	12345678		2468ACF0		12345678	2468ACF0	12345678
/shiftleft/twobitshift	32'h48D159E0	2468ACF0	12345678		91A2B3C0	2468ACF0	48D159E0	2468ACF0	48D159E0
/shiftleft/fourbitshift	32'h48D159E0	2468ACF0	23456780	12345678	1A2B3C00	2468ACF0	8D159E00	468ACF00	48D159E0
/shiftleft/eightbitshift	32'h48D159E0	2468ACF0	23456780	34567800	1A2B3C00	2468ACF0	159E0000	8ACF0000	48D159E0
/shiftleft/muxtwo	32'h48D159E0	91A2B3C0	48D159E0		91A2B3C0		48D159E0	91A2B3C0	48D159E0
/shiftleft/muxthree	32'h8D159E00	468ACF00	23456780		1A2B3C00	468ACF00	8D159E00	468ACF00	8D159E00
/shiftleft/muxfour	32'hD159E000	68ACF000	45678000	34567800	2B3C0000	68ACF000	159E0000	8ACF0000	D159E000
/shiftleft/muxfive	32'h59E00000	ACF00000	67800000	78000000	3C000000	ACF00000	00000000	00000000	59E00000

This waveform shows a shift left logical implementation. You can see the input being shifted left and being filled in at the end with zeros.


Wave - Default

	Msgs					
/shifter/input	32'hF0F0F0F0	F0F0F0F0				
/shifter/shamt	5'h10	01	04	08	16	10
/shifter/output	32'h0000F0F0	78787878	0F0F0F0F	00F0F0F0	000003C3	0000F0F0
/shifter/test	32'h78787878	78787878				
/shifter/onebitshift	32'hF0F0F0F0	78787878	F0F0F0F0			
/shifter/twobitshift	32'hF0F0F0F0	78787878	F0F0F0F0		3C3C3C3C	F0F0F0F0
/shifter/fourbitshift	32'hF0F0F0F0	78787878	0F0F0F0F	F0F0F0F0	03C3C3C3	F0F0F0F0
/shifter/eightbitshift	32'hF0F0F0F0	78787878	0F0F0F0F	00F0F0F0	03C3C3C3	F0F0F0F0
/shifter/muxtwo	32'h3C3C3C3C	1E1E1E1E	3C3C3C3C			
/shifter/muxthree	32'h0F0F0F0F	07878787	0F0F0F0F		03C3C3C3	0F0F0F0F
/shifter/muxfour	32'h00F0F0F0	00787878	000F0F0F	00F0F0F0	0003C3C3	00F0F0F0
/shifter/muxfive	32'h0000F0F0	00007878	00000F0F	000000F0	000003C3	0000F0F0

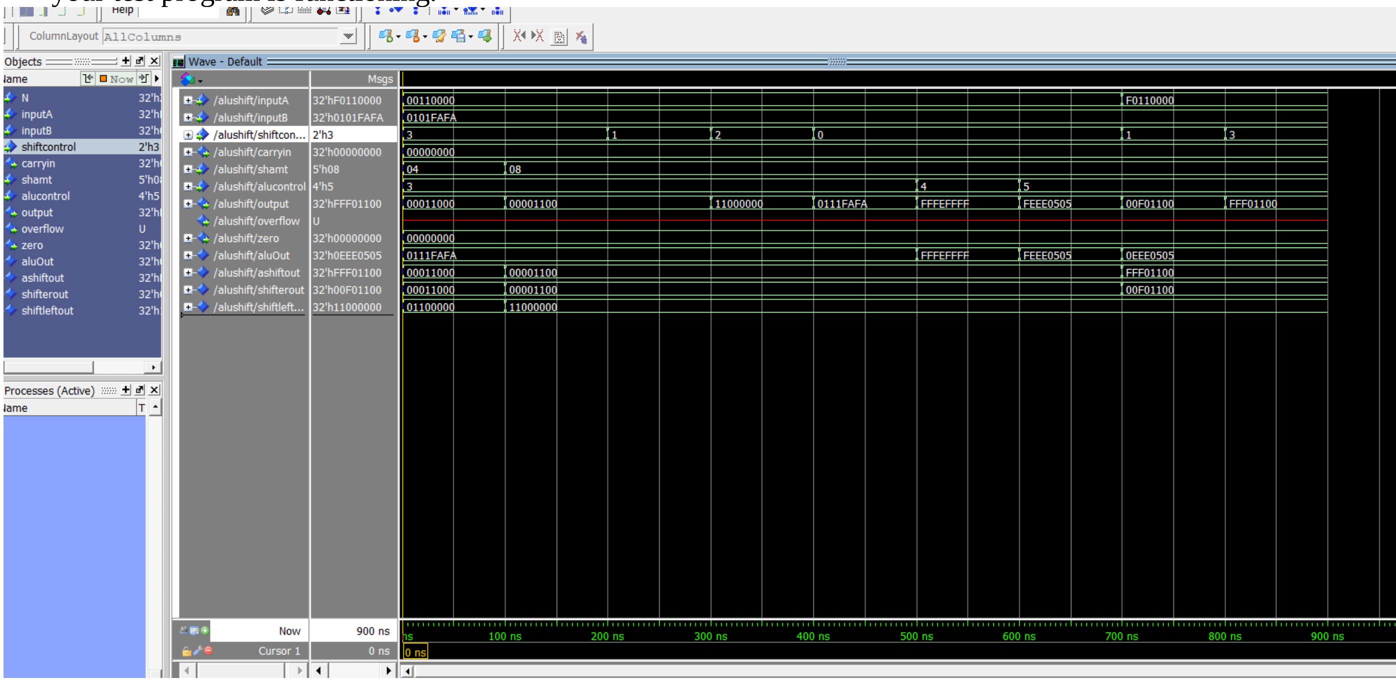
Next we look at a shift right logical. You can tell this is a logical shift because regardless of most significant bit, the output is extended with zeros.



```
                        Msgs
/ashifter/input         32'h0F000000  F0000000                                        0F000000
/ashifter/shamt         5'h04         02        04        07        13                           07        04
/ashifter/output        32'h00F00000  FC000000  FF000000  FFE00000  FFFFFE00  000001E0  001E0000  00F00000
/ashifter/test          32'h07800000  F8000000                                        07800000
/ashifter/onebitshift   32'h0F000000  F0000000            F8000000            07800000            0F000000
/ashifter/twobitshift   32'h0F000000  FC000000  F0000000  FE000000            01E00000            0F000000
/ashifter/fourbitshift  32'h00F00000  FC000000  FF000000  FFE00000  FE000000  01E00000  001E0000  00F00000
/ashifter/eightbitshift 32'h00F00000  FC000000  FF000000  FFE00000  FE000000  01E00000  001E0000  00F00000
/ashifter/muxtwo        32'h03C00000  FC000000            FE000000            01E00000            03C00000
/ashifter/muxthree      32'h00F00000  FFC00000  FF000000  FFE00000            001E0000            00F00000
/ashifter/muxfour       32'h0000F000  FFFC0000  FFFF0000  FFFFE000  FFFE0000  0001E000  00001E00  0000F000
/ashifter/muxfive       32'h000000F0  FFFFFC00  FFFFFF00  FFFFFFE0  FFFFFE00  000001E0  0000001E  000000F0
```

Lastly we have a shift right arithmetic. You can see the input being shifted to the right and being filled in with either ones or zeros depending on the most significant bit.

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



Name	
N	32'h
inputA	32'h
inputB	32'h
shiftcontrol	2'h3
carryin	32'h
shamt	5'h0
alucontrol	4'h5
output	32'h
overflow	U
zero	32'h
aluOut	32'h
ashiftout	32'h
shifterout	32'h
shiftleftout	32'h

	Msgs
/alushift/inputA	32'hF0110000
/alushift/inputB	32'h0101FAFA
/alushift/shiftcon...	2'h3
/alushift/carryin	32'h00000000
/alushift/shamt	5'h08
/alushift/alucontrol	4'h5
/alushift/output	32'hFFF01100
/alushift/overflow	U
/alushift/zero	32'h00000000
/alushift/aluOut	32'h0EEE0505
/alushift/ashiftout	32'hFFF01100
/alushift/shifterout	32'h00F01100
/alushift/shiftleft...	32'h11000000

There is a lot going on in this waveform but everything works as expected.
So shiftcontrol determines what the output is. If shift control is 0 then the output defers to the alu output. If shiftcontrol is 01 then a srl is performed. “10” == sll, “11” == sra.
If shift control is 00 the alu outputs. The alu takes its opcode from the alucontrol signal. This wave form tested an opcode of 4 and 5 which correspond to nand and nor respectively.

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

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	ms			ms		
Reading lab				30		
Pencil/paper design				30		
VHDL design	2 hr			2 hr		
Assembly coding	0					
Simulation	1 hr					
Debugging	1 hr					
Report writing				1hr		
Other:						
Total	4			4		

ii. If you could change one thing about the lab experience, what would it be? Why?
This whole lab was pretty good I don’t think I’d change anything.

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

I enjoyed making the shifting units. I thought it was pretty interesting.