Design of a 4-bit ALU

Objective

To design a 4-bit ALU (similar to the 32-bit ALU shown in your textbook). The functions performed by the ALU are AND/NOR, OR, ADD/SUB, and SLT (set less than for signed numbers).

References

1. 1-bit ALU building block (figure C.5.9 of the 4th edition or figure B.5.9 of the 3rd edition)

2. 4-bit ALU built by instantiating 4 of the above building blocks and adding needed glue logic for SLT implementation (figure C.5.12 of the 4th edition or figure B.5.12 of the 3rd edition)

3. Problem B.24 (3.24) from Appendix-B on the CD associated with the 3rd edition (same as problem C.24 of from Appendix-C on the CD associated with the 4th edition) reproduced at the end of this lab assignment.

Verilog design provided

You are provided with the following files.

`alu_4_bit.v` (incomplete) (to be completed by you)
`alu_4_bit_tb.v` (complete)
`alu_4_bit_different_stimuli_tb.v` (incomplete, but we will discuss this later)

The **`alu_4_bit.v`** contains two modules:
1. a 1-bit ALU building block (**`module alu_1_bit`**)
2. a 4-bit ALU built by instantiating 4 of the above building blocks (**`module alu_4_bit`**)

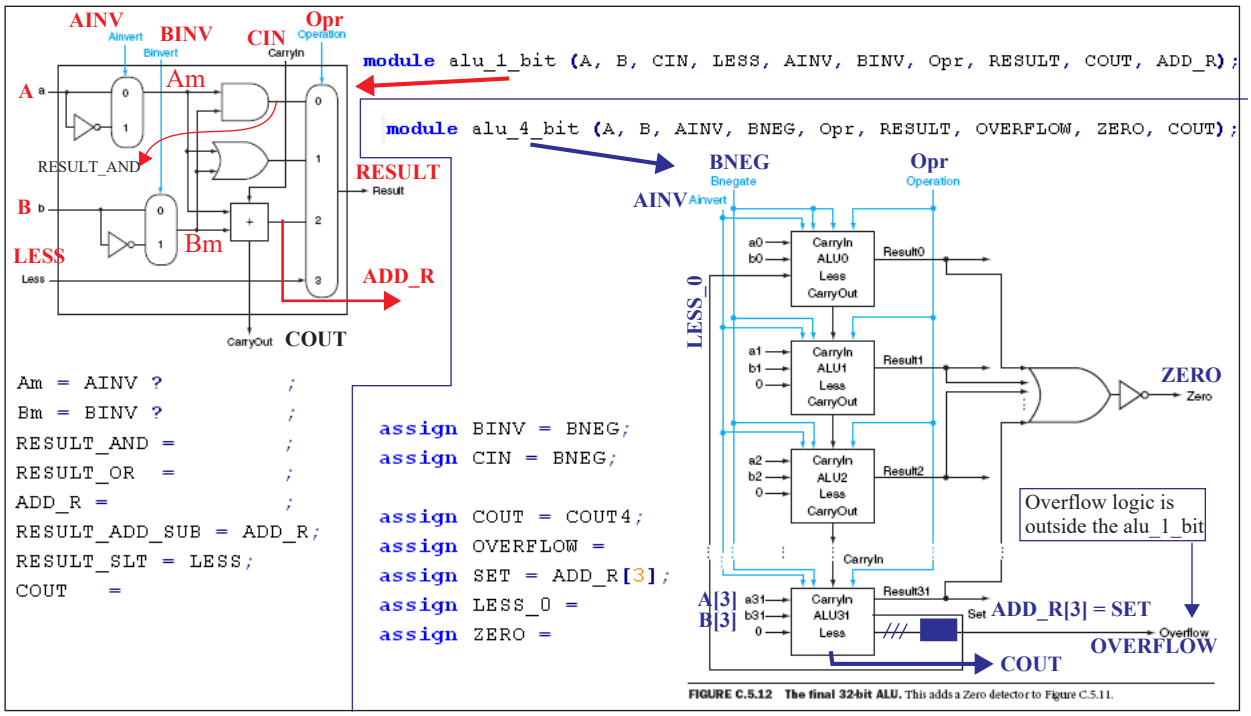
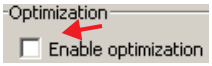


FIGURE C.5.12 **The final 32-bit ALU.** This adds a Zero detector to Figure C.5.11.

Please notice the differences between our verilog design and the textbook design of the building block. The textbook provided different building blocks for the *msb* block (msb = most significant bit) and *non-msb* block. We are providing one building block (**`module alu_1_bit`**) for both *msb* and *non-msb*s. Our building block does not have overflow logic. The overflow logic shall be added by the students ***outside*** the four building blocks.

What you have to do

- Import the .zip file into your `C:\Modelsim_projects` directory and extract files to form a projector directory `C\Modelsim_projects\ee457_lab3_4bit_ALU`
- The directory will have three verilog files and two empty txt files:
- Fix the design on page 4 of this assignment. Do it by hand (using a pen or a pencil).
- Complete the two design modules in the file: **`alu_4_bit.v`**
- Create a modelsim project with the name, `ee457_lab3_4bit_ALU` (and project directory `C\Modelsim_projects\ee457_lab3_4bit_ALU`) . Compile the two files: **`alu_4_bit.v`** and **`alu_4_bit_tb.v`**. Start simulation of the **`alu_4_bit_tb`**. Do *not* select "optimization" option



in the start simulation dialog box 

. Add signals in the core design (instance **`alu_4`** of **`alu_4_bit`**) to the waveform. Simulate for 250ns. Inspect the text messages in the console window which are also recorded in the text file, **`alu_output_results.txt`**, in your project directory. Debug if necessary.

- Complete the table on page 3 and add the values of A and B to the separate testbench, **`alu_4_bit_different_stimuli_tb.v`**, designed for testing these values. Add this testbench to your project, compile it, and simulate this time the **`alu_4_bit_different_stimuli_tb`**. Simulate for 200ns and inspect the text messages in the console window which are also recorded in the text file, **`alu_add_subtract_overflow_results.txt`**, in your project directory. Debug if necessary.

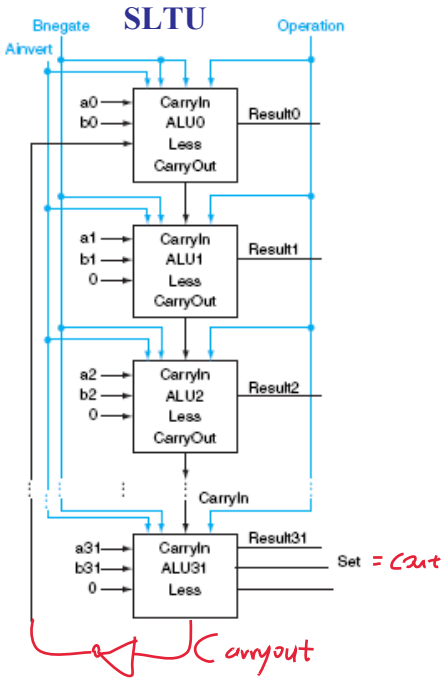
What you have to turn in

All four items are team effort. All team members shall work together and each team member shall be able to answer any question on their submission by the teaching staff.

1 Page 4 of this assignment with your hand-drawn fix.

2. Completed table on page 3.

3. Hand-drawn design on the side for the implementation of SLTU (set-less-than-unsigned) , in place of SLT.



4. Short explanation of why the textbook design reproduced on previous page or your design do not amount to committing the "*sin*" of performing a "combinational feedback"?

The first adder need to receive the set signal from the last adder. However, only the Result0 will change. The rest of the adder will not receive any changing input. Hence, there's no feedback.

5. Online submission of your verilog codes and text files generated

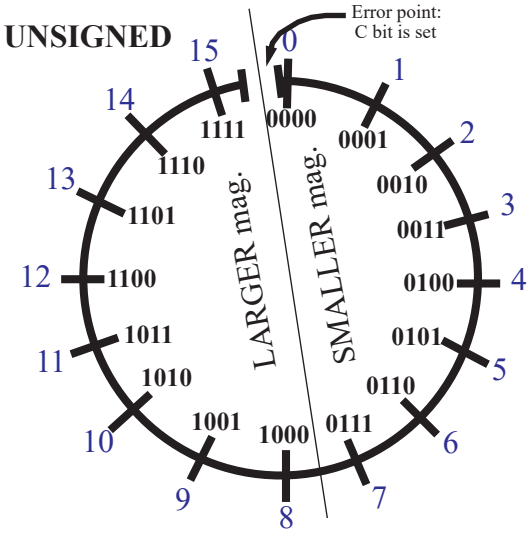
```
submit -user ee457lab -tag puvvada_lab3 alu_4_bit.v alu_output_results.txt
alu_4_bit_different_stimuli_tb.v alu_add_subtract_overflow_results.txt names.txt
```

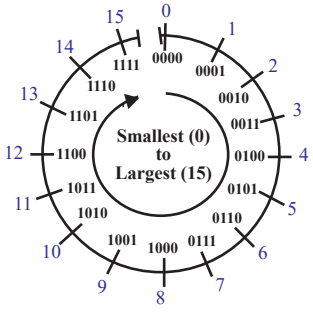
If you try to perform addition and subtraction with each possible pair of numbers in the range of 0000 to 1111, is it possible to encounter all combinations of overflows (signed overflow true or false *and unsigned overflow* true or false*)?* **Yes!**

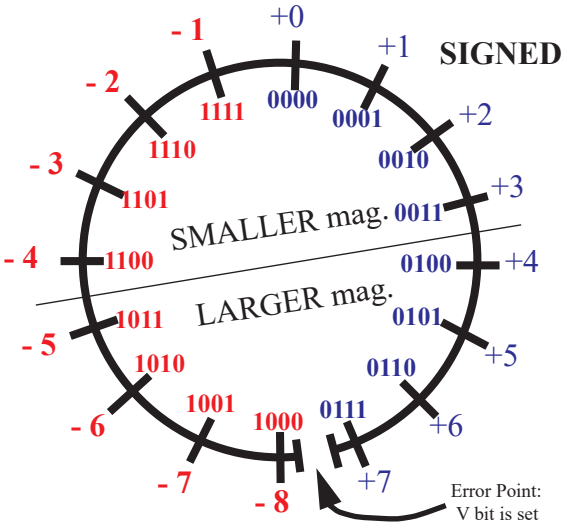
Fill-in the table below with such pairs of numbers and use them in your in your alu_4_bit_different_stimuli_tb.v.

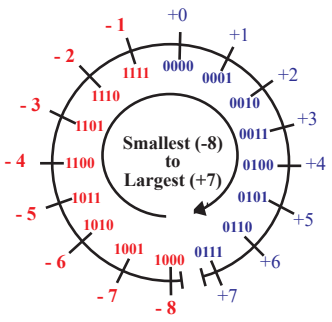
Please **avoid using 0000 and 1000** numbers for A and/or B for this part of the exercise.

4-bit A	4-bit B	Decimal Values of A, B				Operation	4-bit Result RESULT	Result Right/Wrong if numbers are treated as signed numbers	Result Right/Wrong if numbers are treated as unsigned numbers	V OVERFLOW	Raw Carry C4 (COUT)
		Signed Numbers		Unsigned Numbers							
		A	B	A	B						
0010	0011	2	3	2	3	Addition	0101	Right	Right	0	0
0111	1001	7	-7	7	9	Addition	0000	Right	Wrong	0	1
0111	0111	7	7	7	7	Addition	1110	Wrong	Right	1	0
1000	1111	-8	-1	8	15	Addition	0111	Wrong	Wrong	1	1
0101	0100	5	4	5	4	Subtraction	0001	Right	Right	0	1
0010	0011	2	3	2	3	Subtraction	1111	Right	Wrong	0	0
1000	0111	-8	7	8	7	Subtraction	0001	Wrong	Right	1	1
0111	1000	7	-8	7	8	Subtraction	1111	Wrong	Wrong	1	0









Reference: Read problem B.24 (3.24) from Appendix-B on the CD associated with the 3rd edition of the textbook reproduced below (same as Q4.23 in the second edition of the textbook or Q4.25 in the first edition) in addition to the material in section B.5 in the Appendix B on the CD associated with the 3rd edition of the textbook (section 4.5 of the first/second editions) as preparation for this lab.

Problem B.24 (3.24) from Appendix-B on the CD associated with the 3rd edition (same as C.24 of the 4th edition) is reproduced below for your immediate reference.

3.24 [15] <§B.5> The ALU supported set on less than (`slt`) using just the sign bit of the adder. Let's try a set on less than operation using the values -7_{ten} and 6_{ten}. To make it simpler to follow the example, let's limit the binary representations to 4 bits: 1001_{two} and 0110_{two}.

$$1001_{two} - 0110_{two} = 1001_{two} + 1010_{two} = 0011_{two}$$

This result would suggest that $-7 > 6$, which is clearly wrong. Hence, we must factor in overflow in the decision. Modify the 1-bit ALU in Figure B.5.10 on page B-33 to handle `slt` correctly. Make your changes on a photocopy of this figure to save time.

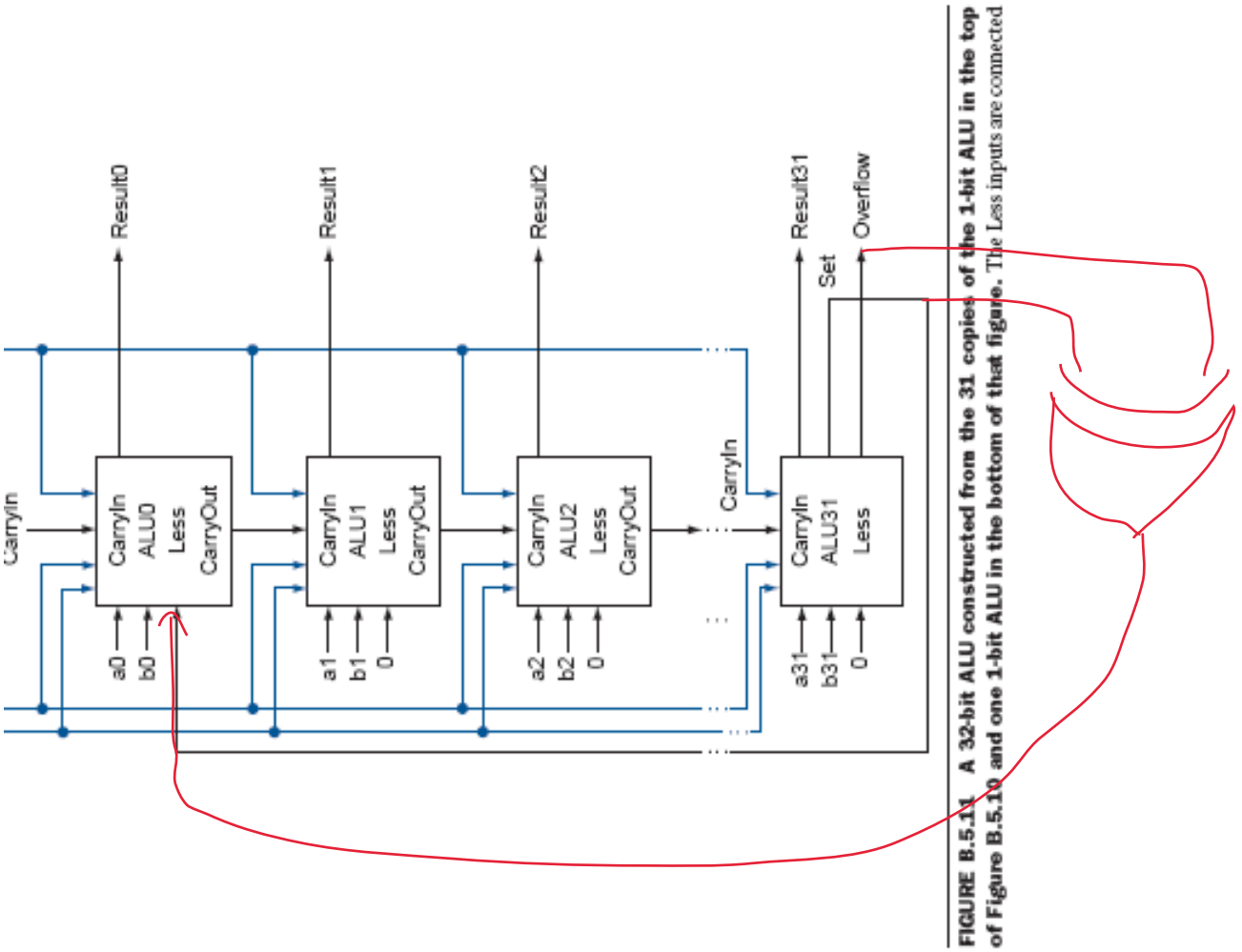


FIGURE B.5.11 A 32-bit ALU constructed from the 31 copies of the 1-bit ALU in the top of Figure B.5.10 and one 1-bit ALU in the bottom of that figure. The Less inputs are connected