Lab 6

Published

Required lab tools: Logisim

[Step 1: Designing an ALU]

An arithmetic logic unit is simply a collection of logic gates that run through a multiplexer in order to select the result. For example, the ALU will take two input pins A and B, and have one output pin Q. In order to select the addition, the selector bit into the ALU will be 0. If I want subtraction, the selector bit will be 1, just like what you did in lab 4.

Now, I set the pins A and B to the binary values, and select whether or not I want them subtracted or added together to get the result Q.

1. You will design your own ALU with four separate functions: invert pin A, pin A AND B, pin A OR B, and A + B. You may NOT use the built-in Logisim adder for A + B. However, you may use the built-in, multibit logic gates for INVERT, AND, and OR.

2. Create three input pins (A, B, and OP). OP is the selector and must be two bits, A and B are eight bits. Create an output pin (Q) which will also be eight bits. For the adder, you will need to create a subcircuit (under Project, Add Circuit...). Then link eight of them together (remember to use the splitter to both split the 8-bit bus into individual bits, and then combine the result!)

3. Design the logic such that the following operations occur:

IF OP = 00, Q=~A
IF OP = 01, Q=A & B
IF OP = 10, Q=A | B
IF OP = 11, Q=A + B

4. Test your ALU by changing the input bits to see if Q gets the proper result. Remember that you can make an 8-bit AND gate, an 8-bit OR gate, and an 8-bit inverter.

[Step 2: Status Registers]

1. Design three, one-bit status registers: Zero (Z), Overflow (V), and Negative (N). **You may NOT use logisim's built-in flip-flops, instead you must create your own**. To account for ALU propagation delay, add another input pin called "Write Enabled". This pin will enable the D flip flop that you will design for the status registers so that it takes the value only when "Write Enabled" is 1.

2. You will need to create three, one-bit output pins: Z, V, and N.

3. When Q = 0000_0000, then Zero will be 1, otherwise, it will always be 0. Do NOT invert all of your inputs. Think of DeMorgan's law when designing the logic for this step!

4. When Q = 1XXX_XXXX, then Negative will be 1 (sign bit is 1)

5. For the overflow bit, use the following formula: $V = \overline{A}\cdot\overline{B}\cdot Q + A\cdot B\cdot\overline{Q}$. Where A and B are the most significant bits (the sign bit) of the 8-bit inputs A and B and where Q is the sign bit of the result. In other words, if A and B are positive, but the result is negative, you have an overflow. Or, if A and B are negative, but the result is positive, you have an overflow as well. NOTE: You are only doing addition, but it is with signed integers.

6. Test your circuit to see if the status registers correctly represent the output of the ALU

7. Under File, export your circuits (alu and 8-bit adder) as a .png image.

[You are finished with this lab!]

Submit both your .png images and your .circ file using Canvas.

Points	100
Submitting	a file upload
File Types	circ and png

Due	For	Available from	Until
Oct 3, 2017	Everyone	Sep 29, 2017 at 7:58am	Oct 3, 2017 at 11:59pm

Lab 6

Criteria	Ratings					Pts
OP=00 (~A)	**10.0 pts** **Full Marks**				**0.0 pts** **No Marks**	10.0 pts
OP=01 (A & B)	**10.0 pts** **Full Marks**				**0.0 pts** **No Marks**	10.0 pts
OP=10 (A | B)	**10.0 pts** **Full Marks**				**0.0 pts** **No Marks**	10.0 pts
OP=11 (A + B)	**20.0 pts** **Full Marks**	**17.0 pts** **6 out of 8 bits work**	**10.0 pts** **4 out of 8 bits work**	**5.0 pts** **2 out of 8 bits work**	**0.0 pts** **No Marks**	20.0 pts
Zero Status Register (No Inverted Inputs)	**20.0 pts** **Full Marks**		**12.0 pts** **Latch/Flip-flop problems**		**0.0 pts** **No Marks**	20.0 pts
Negative Status Register	**15.0 pts** **Full Marks**		**9.0 pts** **Latch/Flip-flop problems**		**0.0 pts** **No Marks**	15.0 pts
Overflow Status Register	**15.0 pts** **Full Marks**		**9.0 pts** **Latch/Flip-flop problems**		**0.0 pts** **No Marks**	15.0 pts
						Total Points: 100.0