# TCSS371 – Machine Organization Assignment 2 –Digital Logic Structures

### 20 Points

**Group Members’ Names**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Purpose**: This homework will test your understanding of the digital logic structures concepts that we covered from Chapter 3 of the textbook.

**Student Learning Outcomes**: The following student-learning outcomes are addressed in this assignment:

* explain the function of basic digital logic circuits
* design simple logic circuits after specifying their behavior using a truth table

1. (Circuits) Create the following circuits in ***Logisim*** as you have seen the instructor do on the lectures. **DO NOT** use the built-in MUX, ADDER or D-Latch for this. You must build the circuits using logic gates.
   1. A 10-to-1 mux using basic gates (AND, OR)
   2. A 5 bit full adder. Create a 1-bit adder first and then use it as a block to create a 5-bit adder.
   3. A 1 bit gated D-Latch and then reuse it to as a block to create a 4 bit D-Latch.
2. (Truth table to circuit) Create a truth table for a 4-bit input where the output will be a 1, if it the decimal number represented by the bit pattern is divisible by 3, and/or 5. Assume that ABCD represent the most significant to least significant bits of a binary pattern. A sample is shown below. Generate the gate-level logic circuit, using the implementation algorithm we discussed in lecture. Use ***Logisim*** to create the circuit. ***The truth table is the only deliverable that will be in the written submission.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | Z (Output) |
| 0 | 0 | 1 | 1 | 1 |
|  |  |  |  |  |
| ….. |  |  |  |  |
|  |  |  |  |  |

1. (ALU) Design a **2-bit** ALU. Below is a block diagram of a simplified ALU. The three select lines should determine which operation the ALU performs from the set {NOTA, NOTB, AND, NAND, OR, NOR, XOR, ADD}

ALU

A

OUT

BA

COUT

CIN

S1

S2

S0

All the components that you need to use are already in Logisim. You will need the basic gates, adder, and multiplexer to complete the circuit. *You don’t need to use transistors or build the adder, multiplexer, etc. instead raise your abstraction level to gates and built-in components*. Label which pattern of select bits perform each operation.

Here’s how each operation works:

NOT A and NOT B operations - Only a single input bits is used in these operations. If the A input is 1 and NOT A is the selected operation, then OUT will be 0. If the A input is 0, OUT will be 1, etc.   
  
Add operation – When the Add operation is chosen, the A, B and Cin inputs are used to determine OUT and Cout.

Other operations – A, B inputs are used to perform the corresponding AND, NAND, OR, NOR, or XOR and the output is shown on the OUT. For example: If A is 1, B is 0, and the XOR operation is chosen using the select bits, then the output will be 1.

You want to make sure that your circuit doesn’t output 1 for Cout, when the Add operation is not chosen. For example, when A is 1, B is 1, and the AND operation is chosen, then Cout should be 0. How do you make sure that Cout is a 0, when we are not performing an Add operation?

Build the circuit in steps. The built in tutorial helps to understand Logisim wiring.  
  
**Group Submission guidelines**: Submit your homework on Canvas as ***digitalhw.docx (or pdf or doc)*** for the truth table and ***digitalhw.circ*** for all the circuits. All circuits must be in the same .circ file.

**Grading Criteria**: This assignment is worth 20 points and will be graded using the rubric below.

|  |  |
| --- | --- |
|  | Points |
| Circuits for Mux, Adder, Latch | 5 Points total All the sub problems are solved correctly using basic logic gates (AND, OR, NOT) and meet the input requirements Mux – 1.5  Adder – 1.5  Latch – 1.5 Labeling - .5 |
| Truth table, Circuit for the truth table | 5 Points total (2.5 Points) Truth table has the correct values for all the inputs and output (2.5 Points) Circuit is implemented correctly for the truth table |
| 2-bit ALU | 10 Points total 6 Points - 1 Point for each gate (other than add)  2 Points for adder 1 Point for making sure that Cout is 1 only when add is used  1 Point for labeling |