Name: _______________________

Lab for ITSC 3181, Introduction to Computer Architecture, Spring 2023

Lab #07: Logic design using Digital (https://github.com/hneemann/Digital): Logic basics, multiplexer, and decoder. Due 03/15. Total points: 1% of the final accumulated grade.

From Lab 07 to Lab 12, you will build a RISC-V CPU to support an essential set of instructions including at least add, and, or, sub, lw, sw, and beq instructions. While you are encouraged to design a 32-bit CPU (each register stores a 32-bit word, and the ALU operates on 32-bit data), you are ok to design 8-bit or 16-bit CPU (each register stores a 8-/16-bit data, and the ALU operates on 8/16 bits data). Designing 8-bit/16-bit is much easier comparing with designing 32-bit, but with same design principle. But regardless of the bitwidth you chose, each instruction is still encoded as a 32-bit word and the address for accessing data from memory are also 32-bit address. You need to select the bitwidth of your CPU and use that bitwidth for the rest of the design in this and future labs. For this lab, you will experiment basic circuit design and create multiplexer, and decoder.

Preparation:

1. Go to Digital (https://github.com/hneemann/Digital) website and download the latest Digital package, which is in a zip file. After download, unzip the file and open the program using Digital.exe, or Digital.sh or Digital.jar depending on what OS you are using.
2. Read the document which is the docu/Documentation_en.pdf file in the unzipped folder, and follow the document 1.2 to create the first circuit of XOR.
3. Read the rest of the document till section 9 to understand the features and capabilities of the Digital, e.g. for creating and using subcircuit please check 1.4.

Important: when designing the circuit, please make sure you follow these rules for adding, changing components, input/output and wires. These rules are applied to all the design lab tasks of the course.

1) **Each input and output of a design MUST be properly and meaningfully labeled.**
2) **Each component, input and output should be correctly configured in terms of its bitwidth and signal control width.**
3) **Keep wires and components organized and layed out according to the circuit schematics rules, a) Inputs on the left (or top), b) Outputs on right (or bottom), c) gates flow from left to right, d) Straight (not angled) wires are best, e) Wires always connect at a T junction (only 90-degree turn or connection), f) A dot where wires cross indicates a connection between the wires, and g) Wires crossing *without* a dot make no connection. While you may not need to follow all those rules for creating correct and small circuit, it is very important when for creating complex circuit. So, make sure you properly organize the components and wires, make them structured and look good. That will help reduce errors.**
4) **For drawing straight wires using mouse, make one turn per each draw. You should not use one draw to make a connection that needs to have two or more 90-degree turns, which would create angled wires. For those wires, you have to make a wire that has a 90-degree turn, and then connect it with another wire that also make 90 turn, and so on. Try to minimize the turns as much as possible. If two wires have to be crossed, but should not be connected, make sure no dot is marked on where the wire is crossed.**

Your grade will be based on both the correctness of your design and the organization of the design, 60% and 40% respectively.

Task	1 (20%)	2 (20%)	3 (20%)	4 (20%)	5 (20%)	Total
Correctness (components, input/output, bitwidth, connection, width of control, etc), 60%						
Organization (I/O labeled and positioned correctly, all straight wires and T junction, dot used correctly, readability etc), 40%						
Total						

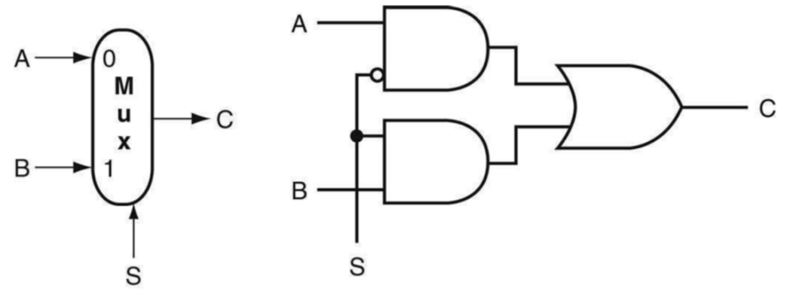
Tasks for Lab #07:

0. Selecting your CPU bitwidth (8-bit, 16-bit or 32-bit): ________________________________

1. (20%) Given the following truth table, 1) write a Boolean equation in the sum-of-products form; 2) minimize the Boolean equation; 3) implement the design from the Digital and experiment with the input (A, B, and C) from the truth table. Check that the output Y matches the truth table.

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

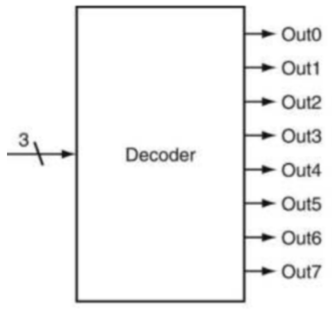
2. (20%) Design and simulate the 1-bit multiplexer from Digital based on the logic from slide 63 of Appendix A. See below. Add input and output to the logic and simulate the logic. Change the input values to check the output value changes according to the behavior of the multiplexer. Save the design in a file.



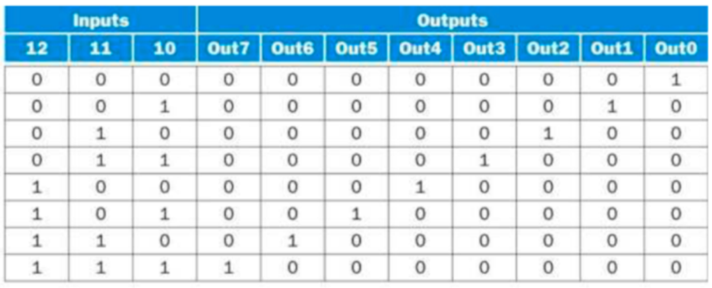
3. (20%) Create a new circuit and design a 32-1 Multiplexer using the Digital-provided multiplexer (Components→Plexer→Multiplexer). The multiplexer is used to read the value of one of the 32 registers (X0, …, X31), the register to read is RS1 and the value is represented as [RS1]. (RS1 is one of the source operands of most instructions, e.g. add rd, rs1, rs2). The bitwidth of the input and output of the multiplexer should be the same bitwidth you selected for your CPU design. The multiplexer is used for read one of 32 registers; thus, you need to set the correct number of Selector bits for the multiplexer. After that, add input and output, input should be named as X0, X1, …, X31, and the selector should be named as RS1 and the output should be named as [RS1].

Configure the multiplexer in the circuit configuration window (Control+ Mouse Click). After that, add all the inputs (X0, …, X31, RS1) and output [RS1], and configure the right data bits for each of the input/output components. This new circuit should be named as “32-1-mux”. Simulate the Multiplexer by setting different value of RS1, then updating the selected register input to see whether the output is changed while you simulate.

4. (20%) Design and simulate a 3-8 decoder from Digital based on the logic from slide 66. See below. Add input and output to the logic, simulate the logic, change the input values, and check which output line is set and whether it is set according to the behavior of a decoder (use the following truth table to check. For inputs, A, B and C can be considered as 10, 11, and 12, which are considered as the bit position of a certain address). Note Command/Ctl-C and Command/Ctl-V work very conveniently for adding the same components

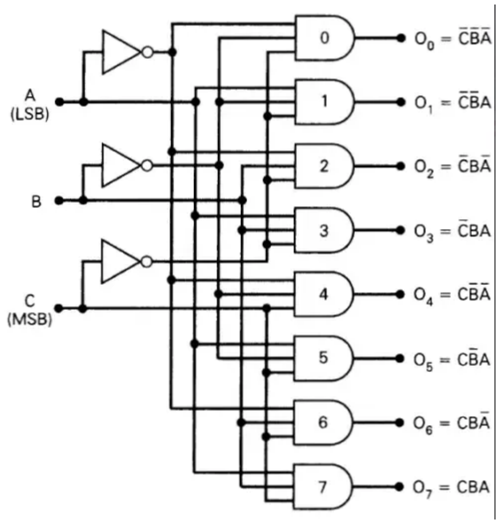


a. A 3-bit decoder



Inputs			Outputs							
12	11	10	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

b. The truth table for a 3-bit decoder



5. (20%) Create a new circuit and design a 5-32 decoder using the Digital-provided decoder (Components→Plexer→Decoder). The decoder is used to set the Write-Enable bit of one of the 32 registers (X0, …, X31) for writing a value to a register. The register to write is RD (RD is the destination operand of instructions, e.g. add rd, rs1, rs2). Configure the decoder in the circuit configuration window (Control+ Mouse Click) to have the correct number of selector bits. After that, add the input (RD) and output [WE0, WE1, … WE31], and configure the right data bits for each of the input/output components. This new circuit should be named as “5-32-decoder”. Simulate the decoder by setting different value of RD to see whether the correct output bit is set while you simulate.

After finishing your design, put your solutions in a file and submit it from canvas. For the design, you can save the design to an image and copy to file, or can screenshot your design and save to file.

If you complete all the tasks for Lab #07, you can start working on Lab #8.