University of Windsor

School of Computer Science
Faculty of Science

COMP-2650: Computer Architecture I: Digital Design
Fall 2020

Assignment#	Date	Title	Due Date	Grade Release Date
Lec 08	Nov 16-18, 2020	**W10: Three-State Logic**	Dec. 02, 2020 Wednesday Midnight AoE	Dec. 09, 2020

The objectives of the lecture (weekly) assignments are to practice on topics covered in the lectures as well as improving the student's critical thinking and problem-solving skills in ad hoc topics that are closely related but not covered in the lectures. Lecture assignments also help students with research skills, including the ability to access, retrieve, and evaluate information (information literacy).

Lecture Assignments Deliverables

You should answer two of the below questions based on your preference using an editor like MS Word, Notepad, and the likes or pen in papers. You have to write and scan the papers clearly and merge them into a single file in the latter case. In the end, you have to submit all your answers in one single pdf file `COMP2650_Lec08_{UWinID}.pdf` containing the following items:

1. Your name, UWinID, and student number
2. The question Id for each answer. Preferably, the questions should be answered in order of increasing Ids. *Please note that if your answers cannot be read, you will lose marks.*
3. Including the questions in your submission pdf file is optional.

Please follow the naming convention as you lose marks otherwise. Instead of {UWinID}, use your own UWindsor account name, e.g., mine is hfani@uwindsor.ca, so, my submission would be: `COMP2650_Lec08_hfani.pdf`

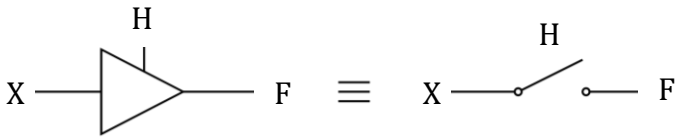
Lecture Assignments
(select only 2 questions based on your preference)

1. **High impedance** or High-Z or Hi-Z is a state when the output is not driven by the input(s), that means output is neither high (1) nor low (0) in which:
   a. The logic behaves like an open circuit, which means that the output appears to be disconnected (the output is electrically disconnected from the circuit);
   b. The circuit has no logic significance; and
   c. The circuit connected to the output is not affected by the inputs to the gate.

   Logical gates such as AND, OR, etc may benefit from this state to exhibit three states in the output. Then, they are called **three-state (tristate) gates**. For instance, a tristate OR would be:

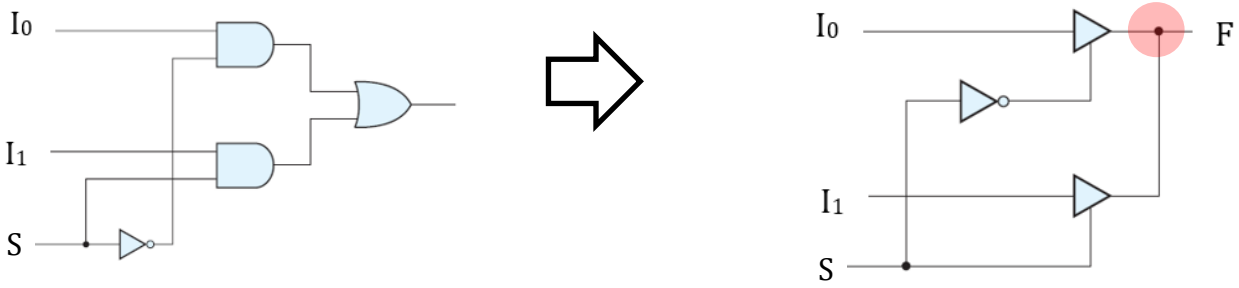
Y	X	H	F
0	0	1	0
0	1		1
1	0		1
1	1		1
X	X	0	Z

However, the one most commonly used is the buffer gate:

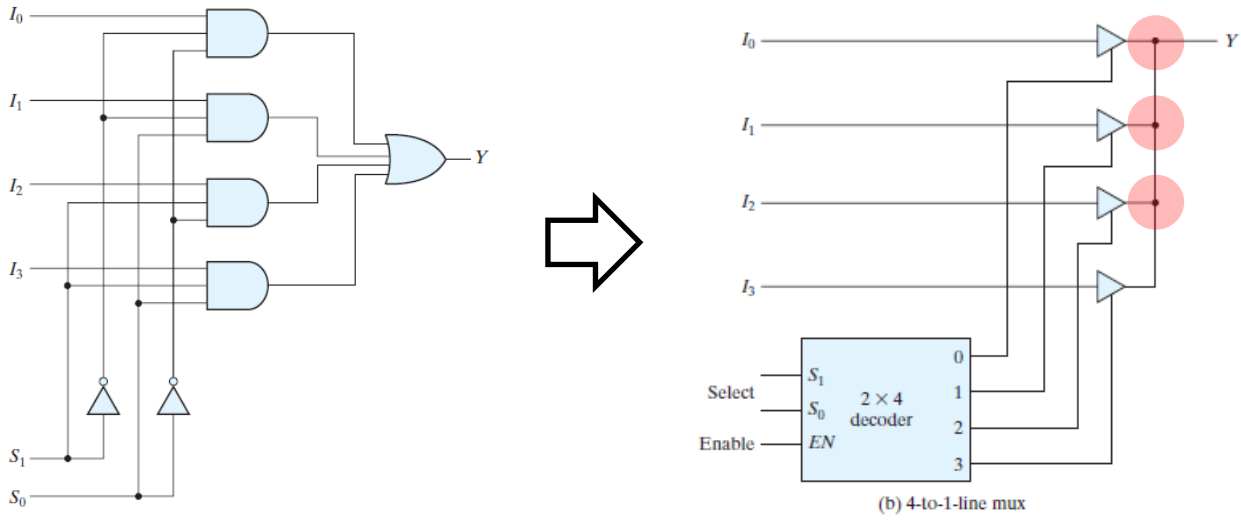


X	H	F
0	1	0
1		1
×	0	Z

Tristate buffer satisfies the need to disconnect the output from the input and can be used to build multiplexers where multiple inputs are going to be attached to the single output:



As seen, the switch input S activates only one AND gate at a time in the original design of the MUX (left). Likewise, S activates one tristate buffer at a time in the new design (right) and connects one and only one wire to the output. *That is why there will be no problem with connecting two wires (red circle) although there is no gate; only one wire at a time is connected*. On the other hand, decoders can activate one and only one of the outputs at a time. Decoders and tristate buffers can be integrated to build MUX as below:



There would be no problem in connecting two wires (red circles) without having any gates since, at a time, only one wire from the input is connected to the output and all others are disconnected. Using decoders and tristate buffers, redesign 1-to-8-line De-mux.

2. Implement the following Boolean function with a 4-to-1-line multiplexer and external gates. *Hint: connect inputs A and B to the selection lines. The input requirements for the four data lines will be a*

function of variables C and D. These values are obtained by expressing F as a function of C and D for each of the four cases when AB = 00, 01, 10, and 11. These functions may have to be implemented with external gates.

a. $F(A, B, C, D) = \sum(11, 3, 4, 11, 12, 13, 14, 15)$
b. $F(A, B, C, D) = \sum(1, 2, 5, 7, 8, 10, 11, 13, 15)$

3. Construct a 16-to-1 multiplexer with 8-to-1 and 2-to-1 multiplexers. Use block diagrams.

4. Implement the following Boolean function with a multiplexer:

a. $F(A, B, C, D) = \sum(0, 2, 5, 8, 10, 14)$
b. $F(A, B, C, D) = \prod(2, 6, 11)$

5. Design an Aiken-to-Gray decoder. *Hint: while Gray code can code 0 to 15, Aiken code is able to code 0 to 9. Hence, the Gray codes from 10 to 15 can be used as don't care conditions.*

6. Design a Gray-to-Aiken decoder. *Hint: while Gray code can code 0 to 15, Aiken code is able to code 0 to 9. Hence, the Aiken codes from 10 to 15 must be generated by concatenating the Aiken code for 1 and the Aiken codes for 0 to 5.*

7. Implement a full adder with two 4-to-1 multiplexers.

8. An 8-to-1 multiplexer has inputs A, B, and C connected to the selection inputs S_2, S_1, and S_0, respectively. The data inputs I_0 through I_7 are as follows. Determine the Boolean function that the multiplexer implements:

a. $I_1=I_2=I_7=0$; $I_3=I_5=1$; $I_0=I_4=D$; $I_6=D'$.
b. $I_1=I_2=0$; $I_3=I_7=1$; $I_4=I_5=D$; $I_0=I_6=D'$.