**Score: \_\_\_\_\_**

**HA3 – Logic Gates to Logic Circuits**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

**Introduction:**

In today’s activities you will continue to work your way up the hardware abstraction hierarchy. Today’s class introduced the idea of a *combinational logic circuit*. We saw how the behavior of such a circuit can be characterized using a truth table. It was argued that any combinational logic circuit can be represented using a schematic (circuit) diagram, a truth table or a logic expression. We then saw how given any one of these representations we can get any of the others. In the activities below you will explore these representations and practice moving between them a little more. Then in a later activity and lab we will see how a facility with these representations can be used to design circuits that perform useful computations.

**Gates as Abstractions:**

It will be important to be able to recognize the basic logic gates and to be able to associate them with their schematic symbols, logic expressions and truth tables. The next few questions ask you to practice this. Your second homework, the first slide from today’s class or this link (<https://www.electronics-tutorials.ws/boolean/bool_7.html>) will be helpful. You should return to these references (or another one that you find and like better) whenever you need to look these up. Eventually, if you use them enough that you will just remember them.

* The following is not required viewing, but if you would like a detailed explanation of each of the gates and their truth tables before starting you can watch the first 3:30 of the *Logic Gate Combinations* video from the Computer Science channel:
	+ <https://www.youtube.com/watch?v=BnB2m1nXZ84> (3:30)

🔑 1. Draw *the schematic symbol* for the gate corresponding to each of the following logic expressions. Place labels on the inputs and output of your schematic that agree with the expression.

 a. $Z=A+B $

 b. $P=\overbar{X ⨁ Y}$

🔑 2. Give the *truth table* for the following logic gate. Assume the inputs are A and B and the output is Q.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **A** | **B** | **Q** |  |
|  |  |  |   |  |
|  |  |  |   |  |
|  |  |  |   |  |
|  |  |  |   |  |
|  |  |  |  |  |

🔑 3. Give the *logic expression* that corresponds to the following truth table:



**Starting from a Circuit:**

In today’s class we saw how to generate a truth table and a logic expression from a combinational logic circuit. The questions in this section will ask you to do the same for different circuits. They will also ask you to extend these ideas to circuits with more than two inputs.

The following are not required viewing, but if you would like to see a few more examples of how to do these conversions before starting, or later if you get stuck, the following videos contain additional worked examples of generating truth tables and logic expressions from combinational logic circuits.

* The part of the *Logic Gate Combinations* video from the Computer Science channel that comes after 3:30 gives some examples of generating truth tables that describe logic circuits.
	+ <https://www.youtube.com/watch?v=BnB2m1nXZ84&t=03m30s> (8:41)
* Mandy Elmore does several examples of finding expressions from circuits in her video *Getting the Logic Expression and Truth Table from a Circuit.* She also shows how you can derive a truth table directly from an expression (one of the conversions we skipped in class but shows up later in this assignment.)
	+ <https://www.youtube.com/watch?v=UNAU7ti4r8E> (9:24)

🔑 4. Consider the following combinational logic circuit:



a. Give a truth table that describes the behavior of this circuit. Be sure to list the input combinations in the “computer science” order (i.e. the same order as our examples).

b. Use the *Substitution Method* illustrated class to derive a logic expression that directly describes the behavior of this circuit.

* You must show all of the intermediate variables you create and their formulas.
* You must also show each step of substitution that you used to get your expression for the output Q.
* Your final expression for the output Q must be expressed in terms of the inputs M and N.

🏆 5. The circuits that we have seen thus far have operated on just two inputs. However, it is frequently the case that combinational logic circuits will have more that 2 inputs. For example, the following circuit has 3 inputs.



a. With three inputs there are more possible combinations of the inputs to be considered. The table below shows all 8 possible combinations of the inputs (x,y,z) to this circuit, along with some of the outputs. Use the provided outputs to check your understanding of the circuit. Then complete the table by filling in the blank cells with the correct values.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | **x** | **y** | **z** | **A** | **B** | **C** | **D** | **F** |  |
|  | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |  |
|  | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |  |
|  | 0 | 1 | 0 |  |  |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

b. Use the *Substitution Method* illustrated class to derive a logic expression for the Boolean function computed by this circuit. Give expressions for each of the intermediate variables (A,B,C,D). Clearly show each step of substitution that you used to determine the expression for the output F. Your final expression for the output F must be expressed in terms of the inputs x, y and z.

🔑 6. We have seen circuits with 2 or 3 inputs. But in practice, a circuit might have any number inputs. The number of inputs that the circuit has will determine how many different combinations of 1’s and 0’s could be used as the input to the circuit. Thus, since a truth table must list all of these combinations, the number of inputs also determines the number of rows that will appear in the truth table.

a. We have seen circuits with 1, 2 or 3 inputs.

i. Fill in the rows in the table below for 1, 2 or 3 inputs (n).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Number of Inputs****(n)** | **Number of Rows****In Truth Table** |  |
|  | 1 |  |  |
|  | 2 |  |  |
|  | 3 |  |  |
|  | 4 |  |  |
|  | 5 |  |  |
|  | 8 |  |  |
|  | 10 |  |  |
|  | 20 |  |  |
|  |  |  |  |

ii. Fill in the row of the table above for circuits with n=4 or n=5 inputs. Hint: Look at the number of rows when there is 1 input, 2 inputs and 3 inputs and find the pattern.

Nothing to write here, just be sure to fill in the table above.

b. Use the pattern that you discovered in part a.ii to give a general arithmetic expression for the number of rows that would be in the truth table for a circuit with *n* inputs.

rows(n) = ???

c. Use your expression from part b to fill in the rows of the table above for circuits with n=8, 10 and 20 inputs. Then be very happy I haven’t asked you to create a truth table for a circuit with 20 inputs!

Nothing to write here, just be sure to fill in the table above.

🏆 d. Give an expression for the number of different Boolean functions that can be computed using logic circuits with n inputs. Explain your answer in a few sentences of your own words.

Bf(n) = ???

**Starting from an Expression:**

In class we also saw that is possible to go directly from a logic expression to a logic circuit. In this section you will practice with another small example and then attempt a more complex example.

* The following is not required viewing, but if you would like to see a complex example worked out before you start or if you get stuck and want another example, then *Digital Logic - implementing a logic circuit from a Boolean expression* from www.finallyunderstand.com does a thorough example:
	+ <https://www.youtube.com/watch?v=0zSHzQJ6vgo> (8:02)

🔑 7. Use the process of *Direct Implementation* illustrated in class to create a combinational logic circuit that is equivalent to the logic expression shown below. Give a schematic diagram of your logic circuit. You can use a drawing program, or you can draw it by hand on paper and then paste a picture of it here. Hint: Use “Edit”->“Paste and Match Formatting” to paste a picture inside of the border.

 $k=\overbar{M+\overbar{MN}}$

8. Use the process of *Direct Implementation* illustrated in class to create a combinational logic circuit that is equivalent to the logic expression below. Give a schematic diagram of your logic circuit. Be sure to arrange your circuit so that it is easy to understand. You can use a drawing program, or you can draw it by hand on paper and then paste a picture of it here. Hint: Use “Edit”->“Paste and Match Formatting” to paste a picture inside of the border.

 $q=\overbar{A+AB} ⨁ \overbar{A\overbar{CD}+BD}$

**Truth Tables to SOP Expressions:**

To go from a truth table to a circuit, you first generate an expression in sum-of-products (SOP) form. From an SOP expression, you use then use the construction for going from an *SOP Logic Expression to a Schematic* that was illustrated in class. This process will always give a circuit that implements the desired function. Note that it may not always be the most efficient or cost-effective solution, but it will work. Next time we’ll be seeing how to improve these circuits. But for now, you’ll practice a bit with the straightforward process.

* The following are not required viewing, but if you would like to see some examples worked out before you start, or if you get stuck along the way, and want some additional examples to help get you going again, the following are good ones:
	+ Mandy Elmore also shows how to derive an SOP expression from a 3-input truth table in her aptly titled video *Writing a Logic Expression From a Truth Table: 3 Inputs*.
		- <https://www.youtube.com/watch?v=uZel7wLztM0> (1:57)
	+ Abelardo Pardo does a more complete example by converting a 3-input truth table to an SOP expression and then to a circuit in the first 7 minutes of his video *From Boolean Expressions to Circuits*. Note that he uses the ‘ symbol to represent NOT (i.e. when he writes X’ we would write $\overbar{X}$). After 7:00 he goes on to talk about equivalent circuit which will be a good preview for our next topic if you are interested.
		- <https://www.youtube.com/watch?v=cblZ7Rdaxog> (7:00)

🔑 9. Consider the following truth table:



a. Write *a logic expression* for the output Z in sum-of-products (SOP) form.

b. Use the technique for *going from an SOP Logic Expression to a Schematic* from class to draw a logic circuit that computes the Boolean function Z. Your circuit must use the process from class even though the truth table above can be represented by a single logic gate.

c. What single logic gate performs the Boolean function described in the truth table above?

10. A common mistake when creating SOP expressions is to combine multiple terms under a NOT bar. For example, in the SOP expression from class, one of the terms was $\overbar{A}\overbar{B}$ (notice the break between the NOT bars). When there are two adjacent terms with NOT bars like this, students will sometimes write $\overbar{AB}$ instead (notice the continuous bar). But these are two different expressions and compute two different things. The expression $\overbar{A}\overbar{B}$ is “NOT A AND NOT B”, while the second expression is “NOT (A AND B)”. These two expressions are not equivalent and when creating SOP expressions, the first is the one we want correct. The following question points out the difference.

a. Fill in the columns in the following truth table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  | **A** | **B** | **AB** | $$\overbar{A}$$ | $$\overbar{B}$$ | $$\overbar{A}\overbar{B}$$ | $$\overbar{AB}$$ |  |
|  | 0 | 0 |  |  |  |  |  |  |
|  | 0 | 1 |  |  |  |  |  |  |
|  | 1 | 0 |  |  |  |  |  |  |
|  | 1 | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

b. If two Boolean functions are the same, then they will agree in all rows of their truth table. If there are any differences, then the two functions are different. Highlight the rows in the truth table above where $\overbar{A}\overbar{B}$ is different from $\overbar{AB}$.

Nothing is required here, but you must highlight the rows in the table above.

Each of the rows that you highlighted indicates a pair of inputs A B where a circuit would give the incorrect result if $\overbar{AB}$ were mistakenly used in the SOP expression instead of the correct $\overbar{A}\overbar{B}$.

**Truth Tables, SOPs and Useful Computations:**

Truth tables are often the most intuitive way to express Boolean functions. All we need to do is to determine what output we would like for each possible input. Thus, when designing logic circuits, we will often start by constructing a truth table. We can then use the truth table to write a logic expression in SOP format, and then finally turn that expression into a circuit.

As an example, imagine we want to build a circuit that is an “Even Detector.” This circuit will output a 1 when there are an even number 1’s in its input, and 0 otherwise. For example, for a 3-input “Even Detector,” if the inputs are 0 0 0 then the output would be 1. This is because there are zero 1’s in the input 0 0 0 and zero is even. Conversely, if the inputs are 0 0 1 then the output would be 0, because there is one 1 in 0 0 1 and 1 is not even.

11. The truth table below shows all possible combinations of three inputs (x2, x1, x0). The first two rows of the truth table have been completed with outputs (E) representing the “Even Detector” function. Fill in the missing values so that the truth table fully expresses the “Even Detector” Boolean function.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **x2** | **x1** | **x0** | **E** |  |
|  | 0 | 0 | 0 | 1 |  |
|  | 0 | 0 | 1 | 0 |  |
|  | 0 | 1 | 0 |  |  |
|  | 0 | 1 | 1 |  |  |
|  | 1 | 0 | 0 |  |  |
|  | 1 | 0 | 1 |  |  |
|  | 1 | 1 | 0 |  |  |
|  | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |

Given the truth table above, we can clearly see the input combinations that must produce an output of E=1. For example, E must be 1 if the inputs x2 x1 x0 are 0 0 0 or if they are 0 1 1 or if they are any of the other input combinations for which E=1. Thus, to produce an SOP expression what is needed are terms that will evaluate to 1 when the inputs take on those particular combinations.

For example, the product term $\overbar{x\_{2}} \overbar{x\_{1}} \overbar{x\_{0}}$ (i.e. $\overbar{x\_{2}} AND \overbar{x\_{1}}AND \overbar{x\_{0}}$) corresponds to the first row of the truth table and will evaluate to 1 with the inputs 0 0 0. To see this, substitute 0 in for x2, x1 and x0 in the term $\overbar{x\_{2}} \overbar{x\_{1}} \overbar{x\_{0}}$ . This gives $\overbar{0} \overbar{0} \overbar{0}$, and NOT 0 is 1 so that evaluates to 1 AND 1 AND 1, which in turn evaluates to 1. There are similar product terms for each of the other lines where E=1.

🔑 12. Write a SOP expression for the output (E) of the “Even Detector” function in terms of its inputs x2 x1 x0.

13. Draw a combinational logic circuit for the Even Detector function using the technique illustrated in today’s class for converting SOP expressions to logic circuits.

Note: While an even detector may seem like an arbitrary example, there are in fact important applications where it is necessary to know if there are an even (or an odd) number of 1’s in an input. It is not required reading, but if you are interested in learning about one see: <https://www.tutorialspoint.com/what-is-a-parity-bit>.

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.