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

**HA4 – Logic Simplification**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

**Introduction:**

In today’s class we saw that for a given Boolean function there are multiple equivalent combinational logic circuits that implement that function. We saw that some of those circuits will be “better” than others (e.g. faster or cheaper). We then saw that Boolean identities can be used to simplify logic expressions to yield different equivalent circuits. The following activities you will give you additional practical experience using Boolean identities to simplify logic expressions (circuits). You will see (or may have already seen) a more formal treatment of Boolean identities and the Boolean Algebra in MATH 211 – Foundations of Higher Mathematics. These two exposures to this topic will reinforce each other and provide you with a more complete picture of the uses and importance of Boolean Algebra in computing.

You will want to have the Boolean Identities Reference Sheet from the class slides and linked on the course page handy as you work through these activities.

**Boolean Identities:**

In class we worked with a table of Boolean Identities. The exercises in this section will help to familiarize you with some of them. It is not required viewing, but if you get stuck or would just like to see a thorough explanation of each of the Boolean identities you might find Johnson Lambert’s video *Boolean Algebra Identities* helpful:

* <https://www.youtube.com/watch?v=MZX6V7u8tZw> (9:43)

🔑 1. In class we used a truth table to check our results when applying the AND form of the Absorption identity. This question uses the same approach to show that several of the other Boolean identities are valid. Fill the truth tables for each identity below, including the intermediate results. Notice that the expressions in the yellow highlighted columns are the two sides of the identity. Thus, if the values in the yellow highlighted columns match (and they should), then the identity is valid.

a. The AND form of the Distributive identity. Note that this identity may seem strange because it is not valid in regular Algebra. It is however, as you will see valid in Boolean Algebra.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | **A** | **B** | **C** | **(A+B)** | **(A+C)** | **(A+B)(A+C)** | **BC** | **A+(BC)** |  |
|  | 0 | 0 | 0 |  |  |  |  |  |  |
|  | 0 | 0 | 1 |  |  |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

b. The OR form of DeMorgan’s identity (i.e. NOR):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  | **A** | **B** |  |  |  |  |  |  |
|  | 0 | 0 |  |  |  |  |  |  |
|  | 0 | 1 |  |  |  |  |  |  |
|  | 1 | 0 |  |  |  |  |  |  |
|  | 1 | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

2. In a prior activity you derived an SOP expression for the XOR function. You can see that that function appears in the Boolean identities table. There is also an identity for XNOR in the table of identities. In this question will verify the XNOR identity.

a. Fill in the truth table below for the XNOR function of A and B.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **A** | **B** |  |  | | | |
|  | 0 | 0 |  |  | | | |
|  | 0 | 1 |  |  | | | |
|  | 1 | 0 |  |  | | | |
|  | 1 | 1 |  |  | | | |
|  |  |  |  |  |  |  |  |

b. Use your truth table from part a to write a logic expression for XNOR in SOP form.

c. How does the expression in part b confirm that the XNOR identity is valid? Hint: Look at the Table of Boolean identities.

**Logic Simplification:**

In this section you will practice using the Boolean identities to simplify logic expressions. The following are not required viewing, but if you get stuck or would just like to see a few additional worked examples of logic simplification the following are good sources:

* John Phillip Jones does a nice job of illustrating the use of Boolean identities to simplify logic circuits in his video *Boolean Identities*.
  + <https://www.youtube.com/watch?v=zehSxcSyWi0> (9:33)
* David Williams’ works out a number of logic expression simplifications in his video *Boolean Algebra Simplification Part 1*.
  + <https://www.youtube.com/watch?v=mxNa0zrjhBU> (9:39)

🔑 3. For each of the following logic expressions, apply the indicated Boolean identity to rewrite the expression. Do not apply other identities or further simplify the expression.

a. Absorption AND form

b. XOR

c. DeMorgan’s OR Form

d. DeMorgan’s AND form

🔑 4. Each of the following pairs of logic expressions are equivalent. For each one the logic expression on the left can be rewritten as the one on the right using one of the Boolean Identities. Identify the Boolean identity and the form (AND/OR) that can be used to rewrite the left expression as the right.

a.

b.

c.

🔑 5. Consider the logic expression

a. Draw a logic circuit that directly implements the Boolean function Q using one NOT gate, two AND gates and an OR gate. You can use a drawing program or you can draw your circuit by hand and paste in a picture of it. Hint: Use “Edit”->“Paste and Match Formatting” to paste a picture inside of the border.

b. Complete the table below to simplify the logic expression for your circuit. Apply the Boolean identity indicated at each step to rewrite the expression from the prior step. Following the indicated steps should get you to the simplified form that is shown at the end. You can fill in the table or you can write out your answer by hand and paste in a picture of it. Hint: Use “Edit”->“Paste and Match Formatting” to paste a picture inside of the border.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Logic Expression** | **Boolean Identity** |  |
|  |  | Start |  |
|  |  | Commutative AND x 2 |  |
|  |  | Distributive OR |  |
|  |  | Inverse OR |  |
|  |  | Commutative AND |  |
|  |  | Identity AND |  |
|  |  |  |  |

6. Consider the logic expression

a. If you were to build a logic circuit that directly implements the function F, how many of each of the following gates would you need?

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Logic Gate** | **Number Needed** |  |
|  | NOT gate |  |  |
|  | Two input AND gate |  |  |
|  | Three input OR gate |  |  |
|  |  |  |  |

b. Apply each of the Boolean identities indicated in the table below to simplify the logic expression for F.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Logic Expression** | **Boolean Identity** |  |
|  |  | Start |  |
|  |  | Distributive OR |  |
|  |  | Inverse OR |  |
|  |  | Commutative AND |  |
|  |  | Identity AND |  |
|  |  | Distributive AND |  |
|  |  | Inverse OR |  |
|  |  | Identity AND |  |
|  |  |  |  |

c. How many gates of what types would be required to implement the simplified version of F?

7. Consider the logic expression

a. How many gates of what types would be required to directly implement Z?

b. Apply each of the Boolean identities indicated in the table below to simplify the logic expression for Z.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Logic Expression** | **Boolean Identity** |  |
|  |  | Start |  |
|  |  | DeMorgan’s OR |  |
|  |  | Double Negation |  |
|  |  | DeMorgan’s AND |  |
|  |  | Distributive OR |  |
|  |  | Inverse AND |  |
|  |  | Identity OR |  |
|  |  |  |  |

c. How many gates of what types would be required to implement the simplified version of Z?

**Moore’s Law:**

8. Almost as soon as Gordon Moore first proposed that the number of transistors on a chip would double every year other people began predicting the end of Moore’s law. Use a Google search to find some pages that talk about the reasons people give for the end of Moore’s law. Skim a number of the pages until you find one that explains several reasons in a way that you can understand (i.e. avoid the super technical explanations). This may take skimming 5 or 10 pages. Once you find a page (or a few pages) that you understand (at least most of) read them more carefully. Then state and briefly summarize, in your own words, one or two of the reasons people say that Moore’s law must end. Include the URL’s from the sites on which your answer is based.

**Universal Gates:**

A set of gates is called universal if they are sufficient to implement any combinational logic circuit (or equivalently any Boolean function). Recall that any combinational logic circuit can be described by a truth table by simply listing the output for all possible combinations of the inputs. We also know that if we have a truth table, we can then create an SOP expression and from that expression build a circuit for it using just AND, OR and NOT gates. Thus, we can conclude that the set of gates {NOT, AND, OR} is universal. That is, for any Boolean function we can build a circuit that computes it using just NOT, AND and OR gates.

That’s kind of cool but may not be too surprising. What might be more surprising is that the NAND gate is universal all by itself (note: NOR is as well). One way to show that NAND is universal is to show that we can build circuits that compute NOT, AND and OR using only NAND gates.

🏆 9. Show NAND is universal by drawing a circuit using only NAND that computes NOT, AND and OR. Note: You can probably look this up, but it will be more fun and more of a learning experience to try to figure it out for yourself!

a. NOT (hint: uses 1 NAND gate)

b. AND (hint: uses 2 NAND gates)

c. OR (hint: uses 3 NAND gates)

**Circuit Design:**

In the previous activity you developed an SOP expression for an “Even Detector.” This was a Boolean function that determines if an input contains an even number of 1’s. The SOP expression for this function is:

You then went on to produce a circuit that computes this function using the straightforward construction illustrated in class. This construction always works, but rarely gives the best circuit. So, normally after deriving an SOP expression but before implementing it you will simplify the expression. This will often allow you to build a better (cheaper and/or faster) circuit for your function.

🏆10. Use the Boolean identities to simplify the above expression for E as much as possible. Show your work and indicate the identity and form that is used at each step. Hint: E can be reduced so that it can be built with two gates, one XOR and one XNOR. Use a table similar to the ones above or write out your answer and paste a picture of it below.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Logic Expression** | **Boolean Identity** |  |
|  |  | Start |  |
|  |  |  |  |

11. Using the truth tables from earlier as models, complete the truth table below to check that the simplified expression you derived in the previous question gives the correct result for E for all possible inputs. Use the blank columns for helpful intermediate results and the yellow column on the right for the final result from your simplified expression.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  | **X2** | **X1** | **X0** | **E** |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

12. Draw a schematic diagram that implements your simplified expression from question 11.

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.