LAB 4

Laboratory Report for CS 2420

Brent Johnson

Computer Science

Texas State University

Department of Computer Science

Bj1107@txstate.edu

*Abstract* – This lab served as an introduction to digital representations of circuits using the DSCH software package installed on our lab machines. DSCH allows us to represent circuits digitally without actually building them on the Elvis boards. This is important because it allows us to build and test more complex circuits without relying on the equipment found in the lab that can sometimes be faulty.

# Introduction

This lab helped to introduce students to digital representation and simulations of electrical circuits. This is an important tool in the electrical engineer's tool-belt because it allows him or her to build complex circuits and test them without worry about the faulty nature of electrical wires and logic boards. Logic chips and gates can be placed wherever the user would like with practically no limit to space. Real-estate is not a problem on the digital board, allowing users to create non-optimal configurations in order to test the logic, then worry about optimization later.

# Experimental Method

To begin the project, I powered on my PC and ran the DSCH program by clicking on its icon on the desktop. As a class, we became familiar with the software as our instructor built a circuit using the following expression:

F = ab + a'c

After we became more comfortable with the software, the instructor released us on our own to build other circuits. I simulated a NOR to NOR circuit (using nor and not gates) for the following function:

F = (x + y + z') (x' + y' + z)

Using buttons for input, a light for the output, and gates from the symbol library, I built the circuit.

Finally, using the DSCH software, I built the majority function. That is, I built a function using three inputs (X, Y, and Z) that is true if and only if at least 2 inputs are true. I made a truth table for said function in order to verify my results. I then used a Kmap in order to simplify this function into its simplest sum of products form. I then drew a circuit diagram for this function using the DSCH software using buttons as input and a single light as the output. I used these buttons and this light to verify that my logic was sound by comparing the outputs against the expected output of my truth table.

# Results

Because this lab was based on digital software rather than actual electrical circuits built on an antiquated logic board, the results were not based on faulty equipment. This allowed be to be more sure of the underlying logic in my functions.

To begin, we got familiar with the software by following the instructor's directions and built the circuit for the function:

F = ab + a'c

I used devices in library, including buttons for inputs, lights for output, and digital wire to connect them all. Because this circuit was created as a group with the guidance of our lab instructor, I did not capture it for inclusion in this results sections.

Next, I used the DSCH software to simulate a NOR to NOR circuit for the following function:

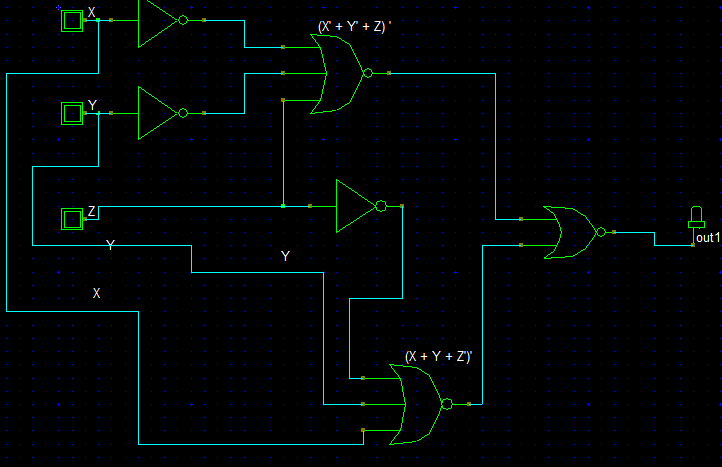
F = (x + y + z') (x' + y' + z)

After applying double negation, then DeMorgan's law, the equation is equivalent to:

F = ((x + y + z')' + (x' + y' + z)')'

I devised the following truth table to test my results.

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | Z | F |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

The following image is a reproduction of my circuit that produced the correct results indicated by the above truth table:

Next, it was time to implement the majority function. I first made the following k-map to simplify the function.

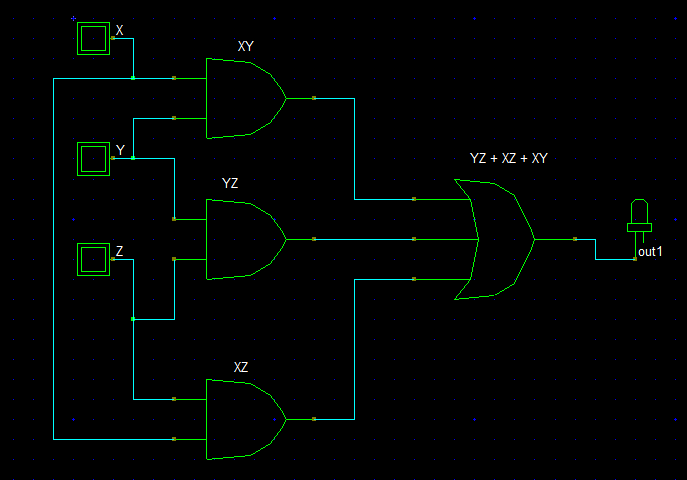
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| z | xy | 00 | 01 | 11 | 10 |
| 0 | | 0 | 0 | 1 | 0 |
| 1 | | 0 | 1 | 1 | 1 |

This allows the function to be reduced to its simplest form of: F = yx + xz + yz

The truth table for this function is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | Z | F |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

I implemented this function digitally. The following is a picture of the circuit I created that produced the correct truth table results:



# Conclusion

In conclusion, this lab was very helpful because it allowed me to get comfortable with digital representations of circuits using software rather than actual electrical wire. This allows for abstraction to a greater degree and allows easier testing and debugging of logical functions. As functions become more complex and take up more real-estate, the software I used in lab today will become even more essential and useful.