CSCE 3303 – Fundamental MicroelectronicsProject: CMOS Circuit SPICE Generator

Mohamed A. Mohamed 900182267

Joesph Bulios Hany

Mark George

**A brief Description to our Project:** This program is basically generating a data Netlist for a circuit of a given Boolean expression. Keep in mind that you are not allowed to have spaces nor brackets. Briefly, the main idea of our project is as follow. We will just need to have two structs, MOS and combination, to help us deal with the expression without any sort of parsing. More clearly, each Mosfet has " A gate, A drain, and A source". Then, I will need combination to help us parse ( I will explain further as we move along) and have some permanent values for PDN. Let's take step by step for further explanation.

1. Utility\_function:

Now, we understand that "Y = A+B" means CMOS -> A + inverter and B +inverter. That is why I have a for loop that checks if one of the inputs is not complemented, I will need to complement it. If it is inverted, I will need to delete ' ' '. Then, call the first important function: "PUN".

1. Is\_valid function

This function is basically check whether the input is valid or not. Validity in my program means that the user is not allowed to enter an input with number or spaces.

1. NOT function:

This functions shows the inverter functionality using PMOS and NMOS.

1. PUN function:

It receives two strings: the complemented expression and non-complemented expression. First, the purpose of this function is to prioritize Anding over ORing to construct PUN. The main reason is that we have many branches due to oring, but using "ANDS", I will need to count the number of wires. In other words, ANDS are in series, whereas ORs are in Parallel. After I am "anding all functions", I will need to take all ORing with each other. In that sense, I have covered the part of precedence. Second important note is "using the struct MOS to connect the current drain value with the next source value and so on. Then, I will need to call the other function to construct PDN.

1. PDN\_with\_Less\_than\_4

To improve the time and space complexity, we can make it more efficient if I have less than 4 inputs. I will simply tokenize by prioritizing ANDing over ORing again. And show the output of PDN.

1. PDN:

The same idea of PUN will be taken, but before it, I will need to complement AND to OR and OR to AND. Another difference we can see in this function compared to PUN is "initializing the next drain with the current Source".

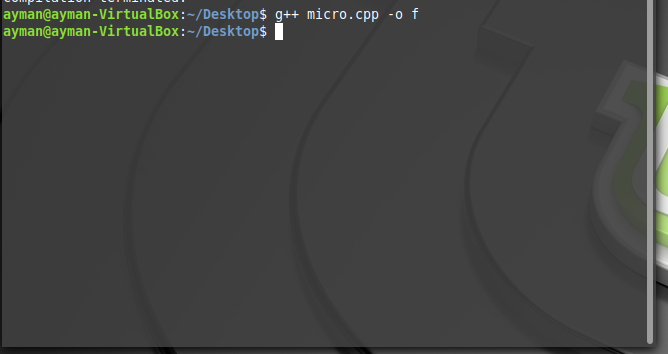
* **We have implemented the bonus number 5.**

**After finishing these steps, the data netlist will be shown in the screen.**

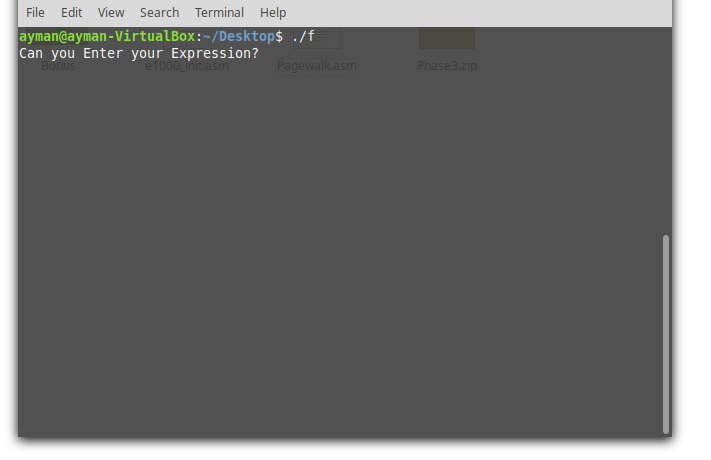
**Data Structures have been used:** We basically used normal vectors and structs.

**How to Run the Program:**

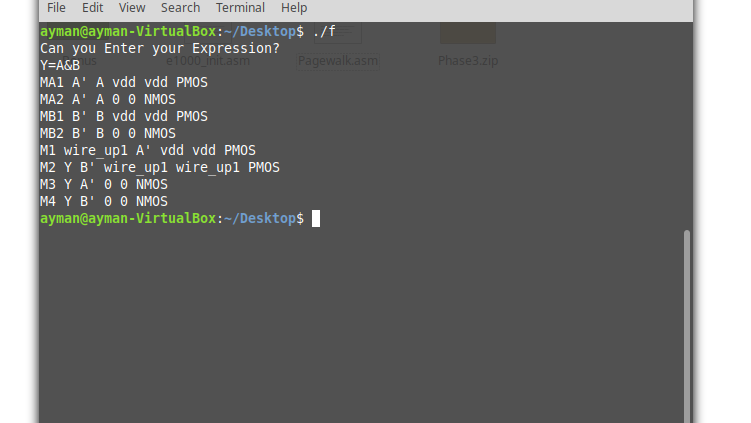
Using Linux:



Then,

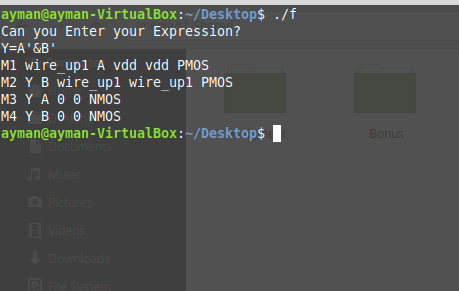


You can see the Netlist result in the terminal.

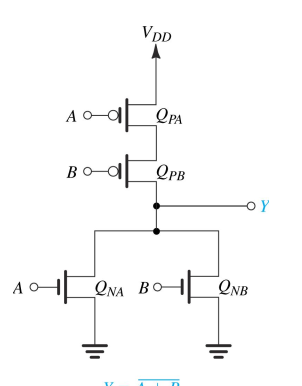


**Test Cases:**

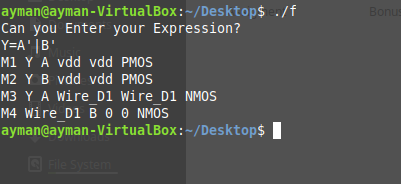
Test Case1:



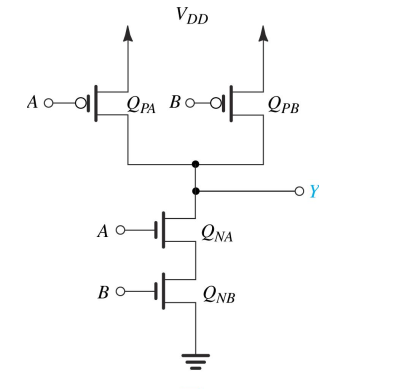
The circuit:



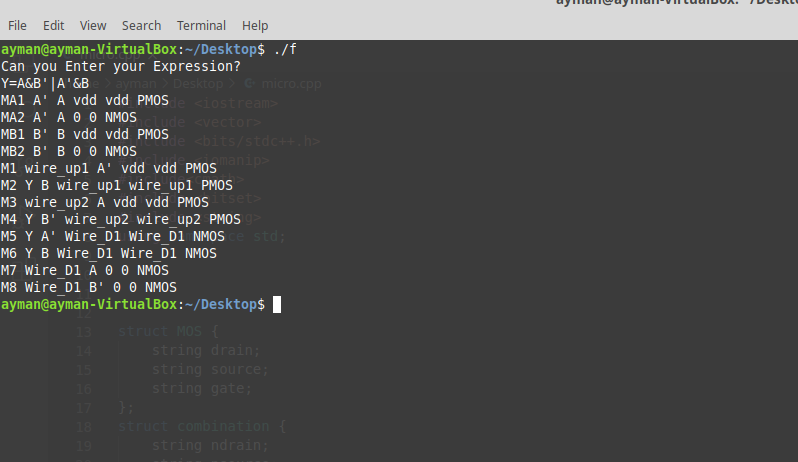
Test Case2:

'

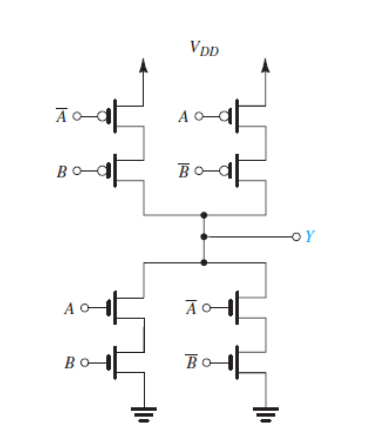
The Circuit:

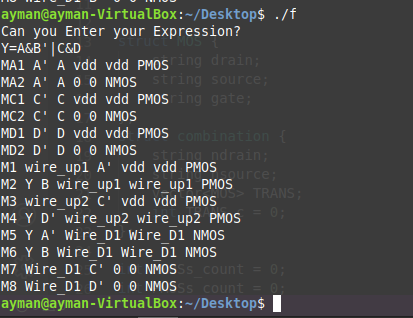


Test Case3:

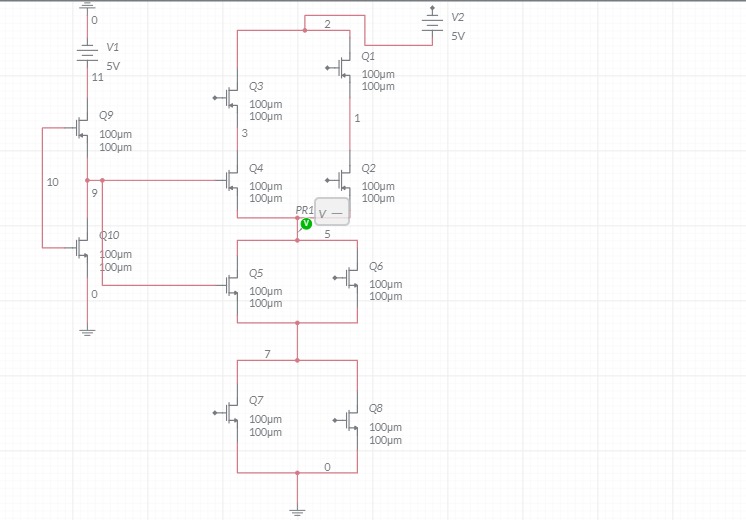


The Circuit:



Test Case3:   


Circuit:



Testing the Bonus:



**LOG**

We have tried to think about the code or the logic that we have implemented. All of us contributed in the part of thinking.

Mohamed Ayman Mohamed:

Implemented PDN, Utility, PDN\_Less\_than, and is\_valid Function. Part of Report.

Joseph Builous:

Implemented PUN Function and NOT function.

Mark George:

Revised, Tested, and Contributed to a part of the Report.