The American University in Cairo

CSCE3303 – Fundamentals of Microelectronics

Spring 2022

Project 1: SPICE Simulator

Dr Dalia Selim

Alaa Alkady - 900191122

Salma Zaghloul - 900183256

Omar Fayed - 900191831

Table of Contents

Description

The following program is a SPICE simulator that receives a boolean expression from the user and outputs the SPICE data statements (netlist code). The output depends on a set of interconnected NMOS and PMOS transistors.

We did not implement any bonus features.

Implementation and Data Structures:

Implementation

To be able to describe the boolean expression or the CMOS circuit, we needed to acknowledge the PMOS and NMOS parts of the CMOS circuit aka the pull up and pull down networks. For that reason, we created two vectors to cater each of these networks.

- ***pullup function:***

  The pull up network vector, which is basically responsible for the PMOS, was done by putting in mind that we needed to check for the transistors iteration to update the terminals accordingly. Therefore, we initiated 2 iterators (check1, check2) that check for the placement of each PMOS to determine its drain, source, body and gate. Hence, if the PMOS is at the top of the circuit, the source and body are "VDD", and the drain is the wire connecting it to the following PMOS, *else* the source and body are the wires connecting the PMOS with the previous one, while the drain will be the output wire.

- ***pulldown function:***

  Similarly, we decided to think of the pull down network, however, with slight changes. These changes were the fact that the drain of the NMOS will be the output wire connected along with the drain of the PMOS. Additionally, the source of the first NMOS will be the wire connecting it to the following NMOS, while the source of the final NMOS will be the ground.

- ***output negations:***

  This function aims to negate the input expression. For example, if the expression is y=a&b|c', then the negation of it will be y'=(a'+b')c. As the original expression is used to express the pullup network that consists of the complemented inputs and the negated expression is used to express the pulldown network that consists of the uncomplemented inputs. This happens by checking the variable size so if it's **greater than** 1, that means that it's already negated so the output is var[i][0] which is the non-negated variable only. However, if it's **equal to** 1, then the output is the negation of the variable.

- ***main function:***

  The main function is where we combine all the logic together. We initialized all our variables with zero, and then asked the user to enter their desired boolean expression that's going to be simulated. Then we added a function to tell the user if their expression is invalid (if the output is in the input boolean equations). Then we had if statements that go through the operations listed in the given boolean expression and adjusted calculations accordingly. Next, we filled in our vectors, "var and op", to be able to use them in the functions we created. Then we called on the functions of the "pullup" and "pulldown" to call on the vectors we constructed with the right parameters given. Lastly, we called on the output functions for the final result to be displayed, the spice data statement.

- ***output function:***

  The function that prints out the SPICE data statement in the form:

  Mname drain gate source body type

for both the pullup and pulldown functions. Moreover, it increases the number of the transistor (Mname) with each iteration.

Data structures

1. *Classes:*
   We created the class “MOSFET” to be able to make our own variables that have to do with the transistor. Therefore, we have the drain, source, gate, etc. The class also includes the constructors to initialize the variables of the class.

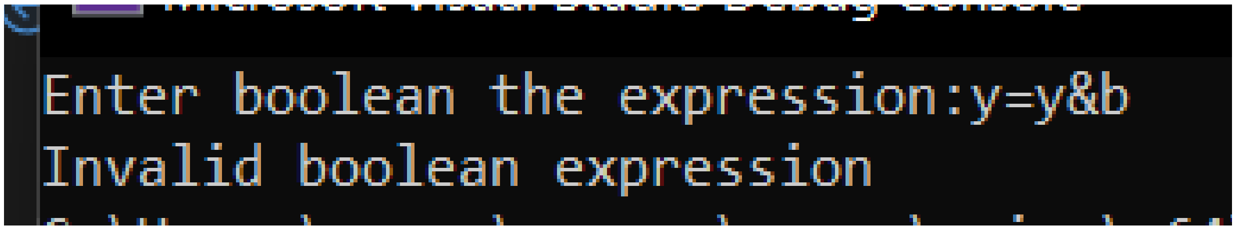
2. *Vector:*
   We used vectors instead of arrays to be able to store our data dynamically and not worry about the size. For example, one of the vectors we created was “expression”, which stores the whole expression for each entry that the use plugged in.

User Guide:

The user will input the boolean expression that will be read by the program and used to output the SPICE netlist code. The program will check for invalid input. For example, if the user entered the output variable as an input in the same expression, an error message will appear.

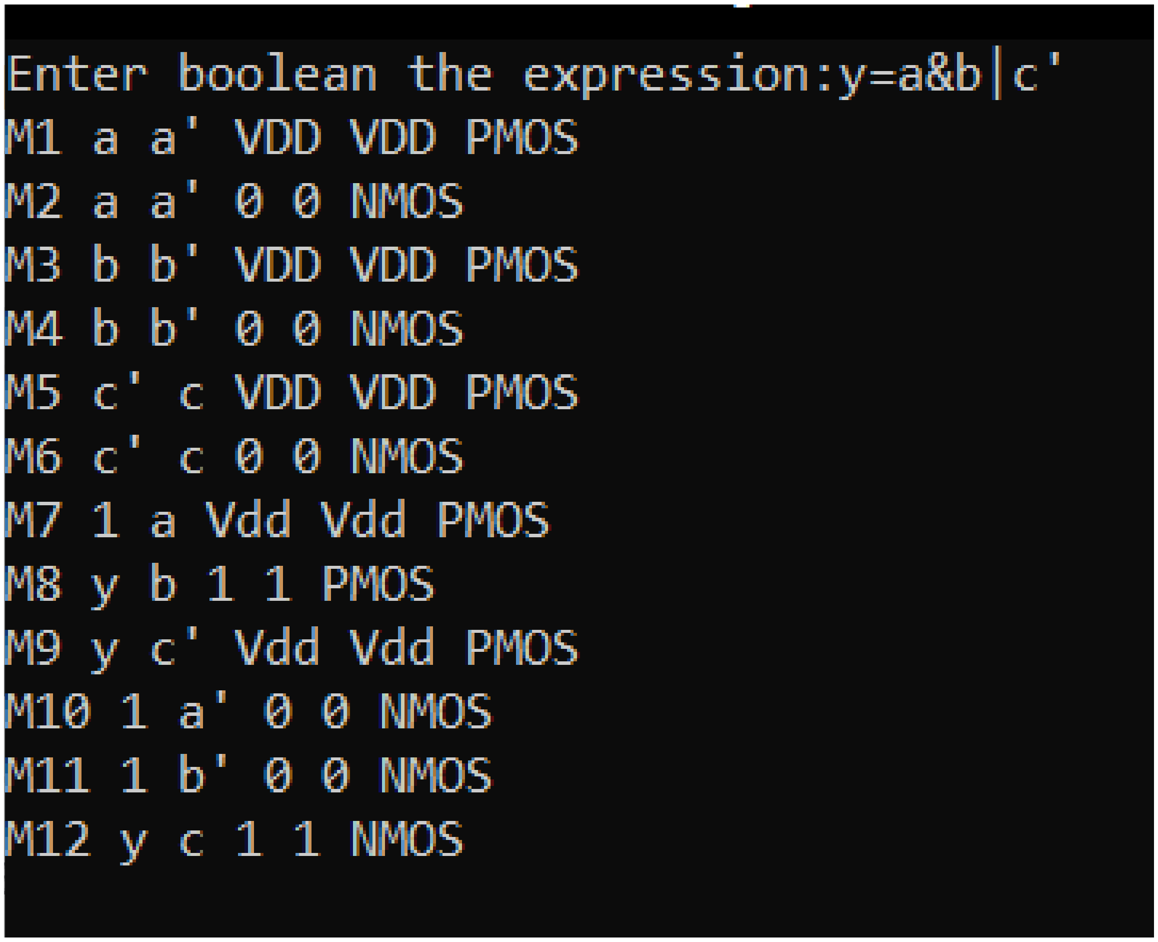
Test Cases

Test case 1: The invalid input

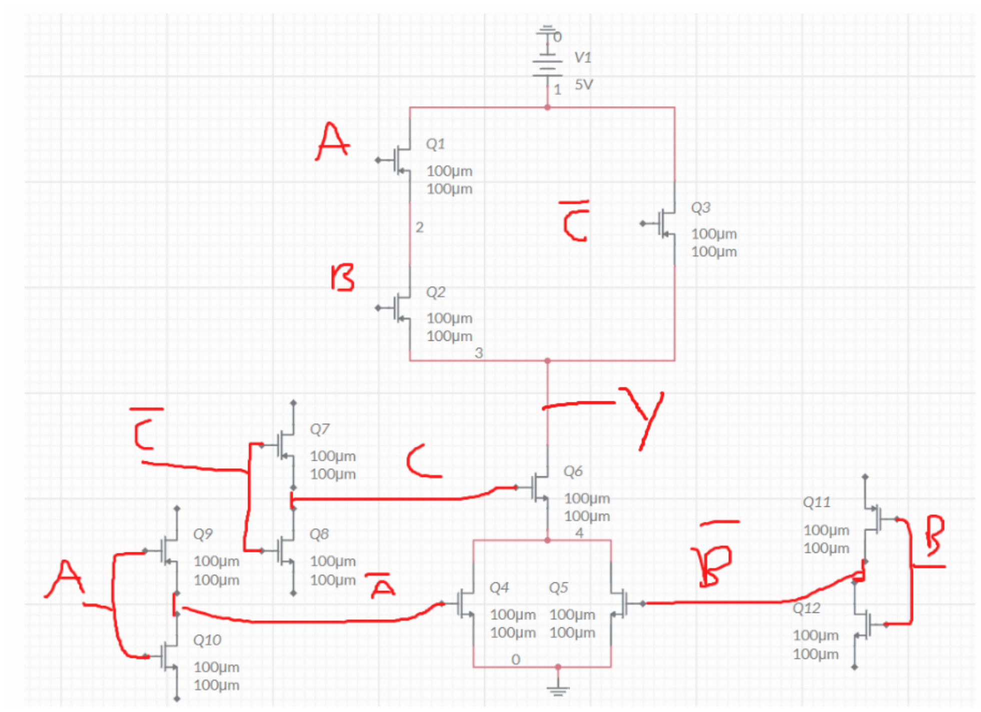


```
Enter boolean the expression:y=y&b
Invalid boolean expression
```

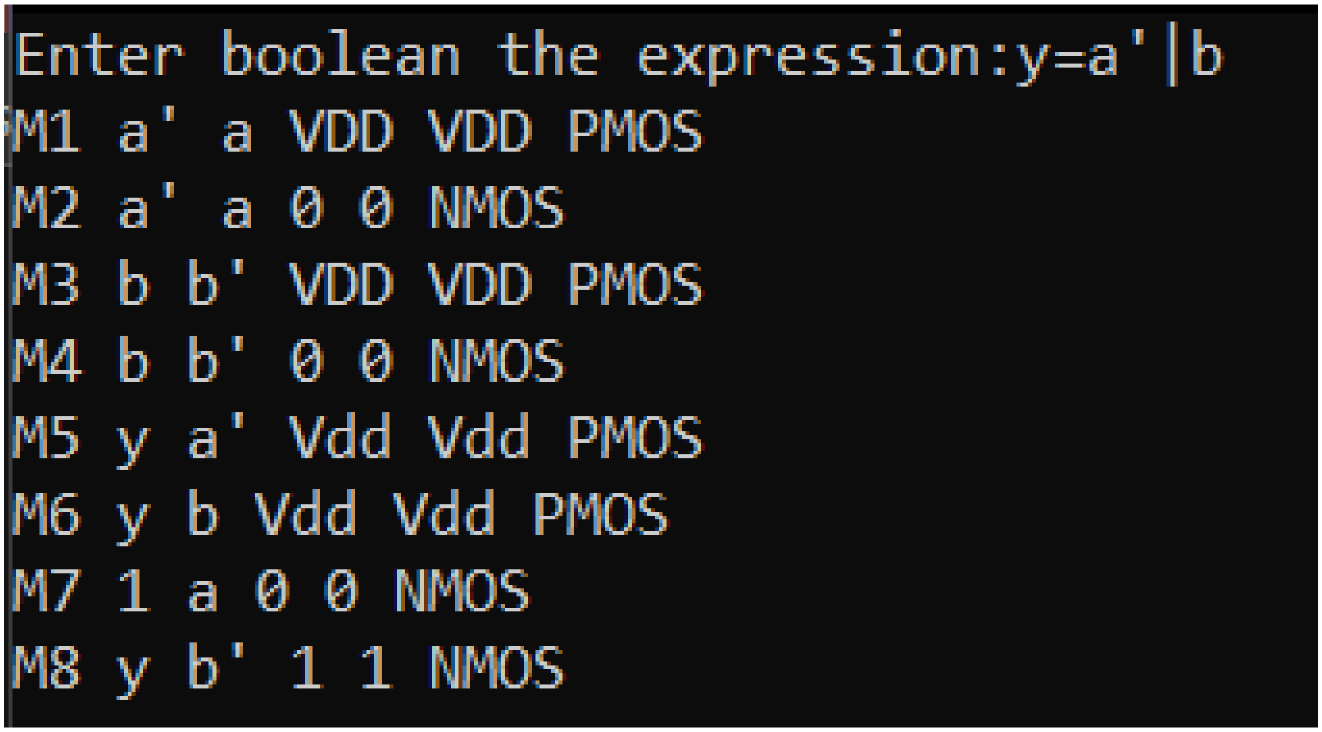
Test case 2:



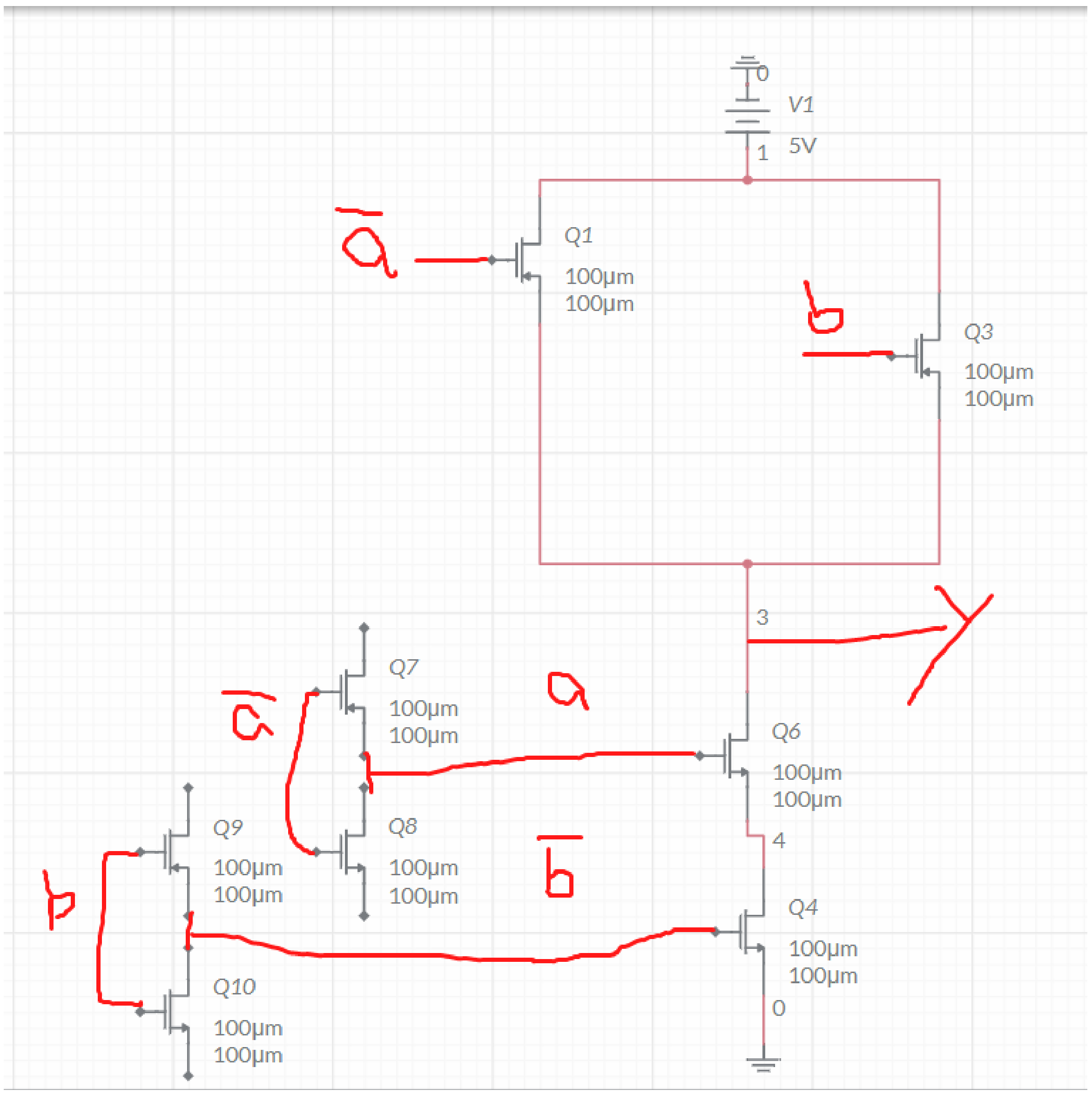
```
Enter boolean the expression:y=a&b|c'
M1 a a' VDD VDD PMOS
M2 a a' 0 0 NMOS
M3 b b' VDD VDD PMOS
M4 b b' 0 0 NMOS
M5 c' c VDD VDD PMOS
M6 c' c 0 0 NMOS
M7 1 a Vdd Vdd PMOS
M8 y b 1 1 PMOS
M9 y c' Vdd Vdd PMOS
M10 1 a' 0 0 NMOS
M11 1 b' 0 0 NMOS
M12 y c 1 1 NMOS
```



Test Case 3:



```
Enter boolean the expression:y=a'|b
M1 a' a VDD VDD PMOS
M2 a' a 0 0 NMOS
M3 b b' VDD VDD PMOS
M4 b b' 0 0 NMOS
M5 y a' Vdd Vdd PMOS
M6 y b Vdd Vdd PMOS
M7 1 a 0 0 NMOS
M8 y b' 1 1 NMOS
```



Team Contribution:

We entered zoom together and worked simultaneously, helping each other when someone needed the other.