Fundamental Microelectronics
CMOS Circuit SPICE Generator

Abdelrahman Said – 900192935

Mohammed Farag – 900193654

Mostafa Ibrahim – 900192397

Submission Date: May 21st 2021

Program Description

The program prompts the user to enter a Boolean expression and generates the corresponding SPICE deck corresponding to the CMOS circuit realizing that expression. The program supports multiple expressions in a single input given that they are separated by a semi-colon. It supports “&” and “.” as symbols for AND, “|” and “+” as symbols for the OR operation, and apostrophe “ ’ ” as a symbol for NOT.

Implementation Details

Data Structures

The program takes the Boolean expression(s) entered by the user in the form of a string, separate them in case of multiple expressions based on the semi-colon, and then store them in a **vector of strings**, which allows us to loop on the expressions and process them one by one.

Algorithms

NOT function: Takes a string input, and a passed by reference string output that acts as the wire coming out from the inverter (the inverted input), and returns the SPICE deck representing the CMOS inverter.

AND function: Takes two string inputs, and a passed by reference string output which acts as the output of the AND gate (the two ANDed inputs) and returns the corresponding SPICE deck. The configuration of the CMOS transistors that was used to implement the AND gate is and inverted NAND gate, so it is implemented using 6 transistors.

OR function: Takes two string inputs, and a passed by reference string output which acts as the output of the OR gate (the two ORed inputs) and returns the corresponding SPICE deck. The configuration of the CMOS transistors that was used to implement the OR gate is and inverted NOR gate, so it is implemented using 6 transistors.

validate function: A Boolean function that takes as input the string Boolean expression entered by the user, and it checks that the symbol used for the output is not being used in the input (e.g., y=y&a is an invalid input). It returns false if the input is invalid and true otherwise.

generateExpressions function: Takes the expressions as an input vector, iterates over them expression by expression, and we iterate over each expression checking for the Boolean symbols in the order of precedence.

First, we check for the apostrophe (NOT), if found, we pass the letter that is right before the apostrophe as input to the NOT function. After that, it replaces the two characters in the Boolean expression, the input and the apostrophe, with the output wire generated by the NOT function.

Next, we check for ampersands "&" and dots "." (AND), if found, we pass the characters representing the 2 inputs, before and after the AND symbol, as input to the AND function. There are cases when the inputs are not single characters entered by the user, but wires coming from other gates That is why we check whether the input symbols are single character symbols (e.g., a), or wires (e.g., W1) to make sure that we are passing the right input to the AND function. Finally, we replace the whole AND expression with its output (e.g., in y=a|b&c, if the output b&c is W1, the new expression would be y=a|W1)and we continue iterating on the same string.

Finally, We check for the OR symbols ("|" or "+"), and we adapt the same logic for AND explained above.

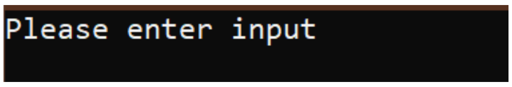
getExpressions function: A function that prompts the user to enter the expression(s), separates the expressions based on semi-colons in case of multiple expressions, and passes them to the validate function. If the input is valid, the expression is pushed back in a vector called "expressions", otherwise, the user will be asked to enter a new valid expression to replace the invalid one.

In case the user passes one expression, it is passed directly to the validate function, and then once we reach a valid expression, it is pushed back to the "expressions" vector.

Finally, we call generateExpressions at the end of the function and pass "expressions" as its input.

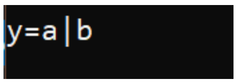
User Guide

After running the code, the user will be asked to enter the Boolean expression(s).



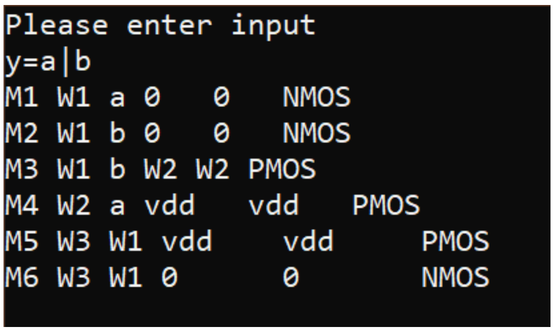
```
Please enter input
```

The user must enter the expression in the form of a letter = expression. Otherwise, the code will not run properly. An example of what the user should enter would be the following:



```
y=a|b
```

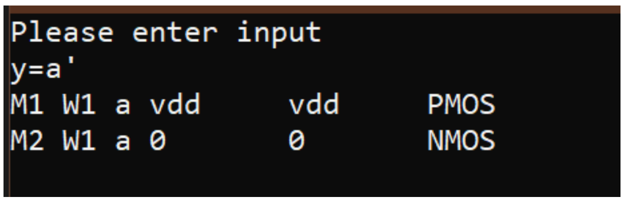
After entering the expression, the output SPICE netlist will be displayed after pressing enter.



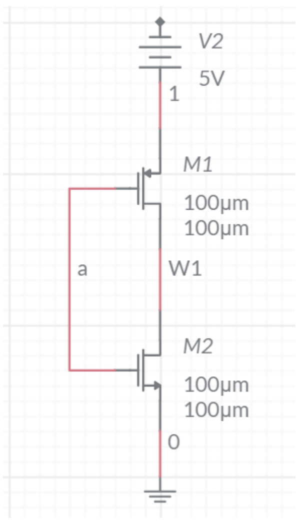
```
Please enter input
y=a|b
M1 W1 a 0   0   NMOS
M2 W1 b 0   0   NMOS
M3 W1 b W2 W2 PMOS
M4 W2 a vdd   vdd   PMOS
M5 W3 W1 vdd     vdd     PMOS
M6 W3 W1 0       0       NMOS
```

Test Cases

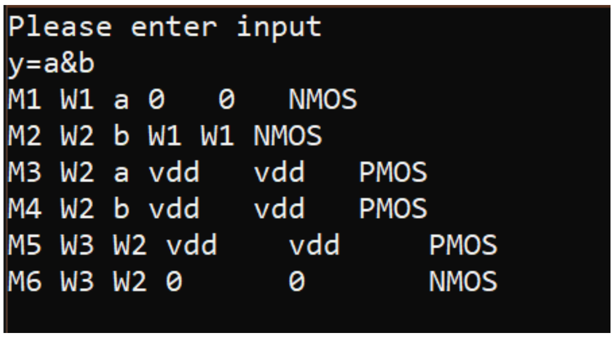
NOT



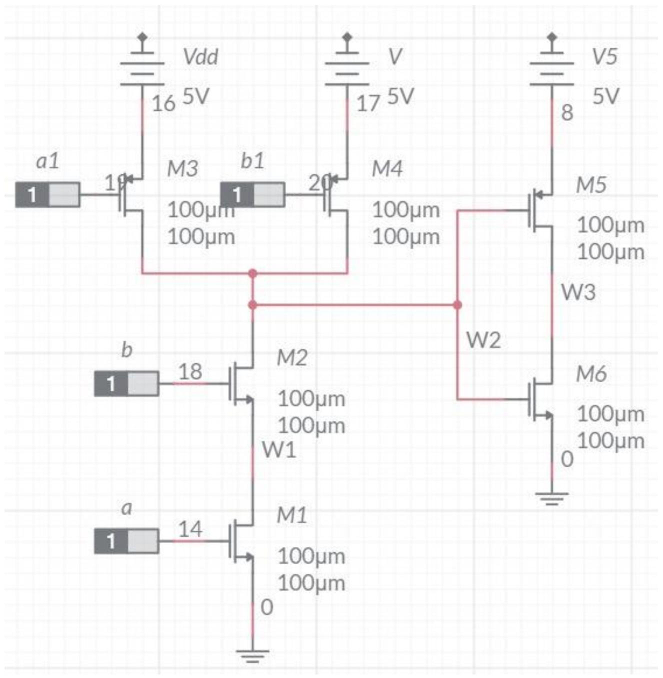
```
Please enter input
y=a'
M1 W1 a vdd      vdd      PMOS
M2 W1 a 0        0        NMOS
```



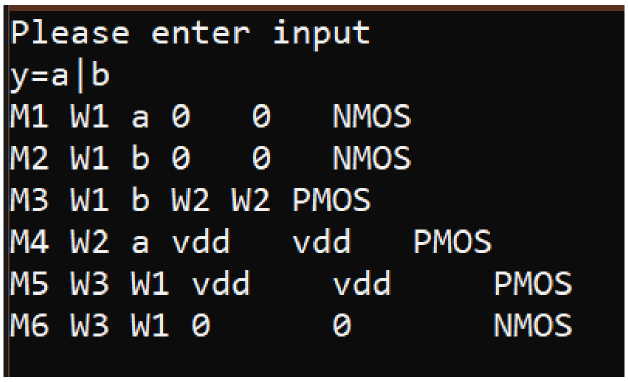
AND



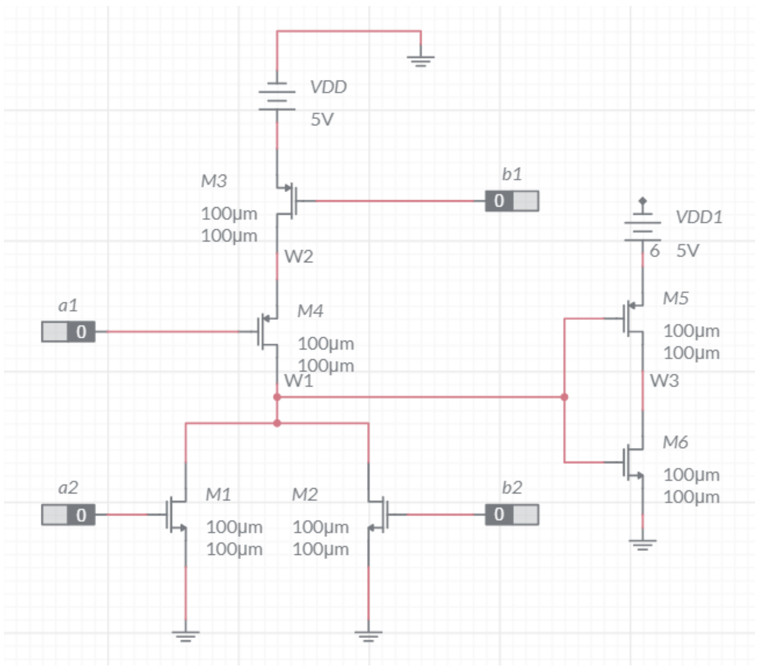
```
Please enter input
y=a&b
M1 W1 a 0   0   NMOS
M2 W2 b W1 W1 NMOS
M3 W2 a vdd   vdd   PMOS
M4 W2 b vdd   vdd   PMOS
M5 W3 W2 vdd    vdd     PMOS
M6 W3 W2 0      0       NMOS
```



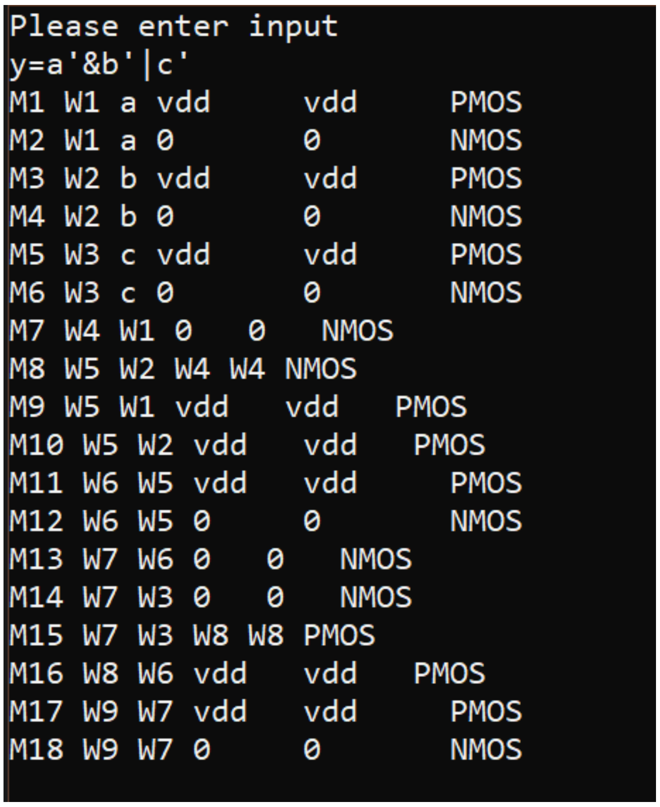
OR



```
Please enter input
y=a|b
M1 W1 a 0   0   NMOS
M2 W1 b 0   0   NMOS
M3 W1 b W2 W2 PMOS
M4 W2 a vdd   vdd   PMOS
M5 W3 W1 vdd    vdd     PMOS
M6 W3 W1 0      0       NMOS
```

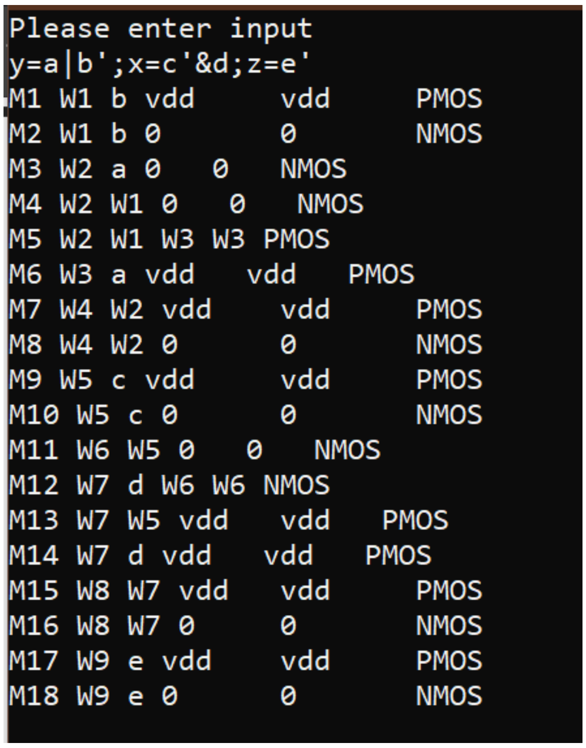


$y = a'\&b'|c'$



```
Please enter input
y=a'&b'|c'
M1  W1  a  vdd      vdd      PMOS
M2  W1  a  0        0        NMOS
M3  W2  b  vdd      vdd      PMOS
M4  W2  b  0        0        NMOS
M5  W3  c  vdd      vdd      PMOS
M6  W3  c  0        0        NMOS
M7  W4  W1  0    0    NMOS
M8  W5  W2  W4  W4  NMOS
M9  W5  W1  vdd    vdd    PMOS
M10  W5  W2  vdd    vdd    PMOS
M11  W6  W5  vdd    vdd      PMOS
M12  W6  W5  0      0        NMOS
M13  W7  W6  0    0    NMOS
M14  W7  W3  0    0    NMOS
M15  W7  W3  W8  W8  PMOS
M16  W8  W6  vdd    vdd    PMOS
M17  W9  W7  vdd    vdd      PMOS
M18  W9  W7  0      0        NMOS
```

Multiple expressions


```
Please enter input
y=a|b';x=c'&d;z=e'
M1  W1  b  vdd      vdd      PMOS
M2  W1  b  0        0        NMOS
M3  W2  a  0    0    NMOS
M4  W2  W1  0    0    NMOS
M5  W2  W1  W3  W3  PMOS
M6  W3  a  vdd    vdd    PMOS
M7  W4  W2  vdd     vdd      PMOS
M8  W4  W2  0       0        NMOS
M9  W5  c  vdd      vdd      PMOS
M10  W5  c  0       0        NMOS
M11  W6  W5  0    0    NMOS
M12  W7  d  W6  W6  NMOS
M13  W7  W5  vdd    vdd    PMOS
M14  W7  d  vdd    vdd    PMOS
M15  W8  W7  vdd    vdd      PMOS
M16  W8  W7  0      0        NMOS
M17  W9  e  vdd     vdd      PMOS
M18  W9  e  0       0        NMOS
```

****Note:** We were not able to add the CMOS circuits for the last two test cases because of the 25 components limit on Multisim.

Members' Contributions

This project was done over Zoom meetings where the three of us together would work on every part of the project together. We brainstorm for feasible ideas, convert them into algorithms, then translate them into C++ code. It is safe to say that everyone contributed equally to the group. Our workflow for this project went as follows, Mostafa shares his screen and types the code we agree on, Abdelrahman would draw the CMOS circuits so we can get a clearer understanding of the connections and consequently be able to turn those connections into code that generates the right output. Mohammed was responsible for researching functions that would help us save time and have a smoother looking code (e.g., find).