CSCE 3303 – Fundamental Microelectronics
Spring 2021

Project: CMOS Circuit SPICE Generator

Project Overview: The goal of this project is to implement a program that generates the part of a SPICE netlist (also known as a SPICE deck) that describes a CMOS circuit realizing an arbitrary Boolean function provided by the user.

Implementation language: Any general-purpose programming language of your choice (e.g., C, C++, C#, Java, Python, etc.). The resulting application is not expected by default to have any GUI (i.e., it can be a simple console application).

A very brief description of SPICE decks:
A SPICE deck is a file starting with a **title statement** (one line) which is mostly ignored by SPICE simulators. It ends with the "**`.END`**" **statement**. In between, there are 3 parts which can appear in any order. For the purposes of this project, the most important part (usually appearing immediately after the title) is the **data statements** part which lists the circuit components including how they are interconnected and their models if needed. The SPICE deck also contains **control statements** which specify the type of analysis to perform on the circuit and **output statements** which specify which signals should be monitored and displayed. Any line starting with an asterix "**`*`**" is considered a comment and ignored by the simulator. Long statements spanning more than one line should be split using the plus "**`+`**" symbol used in the beginning of a line to indicate that this line is a continuation of the previous one.

In the **data statements** part, each circuit component is given a label (just an identifying name). The same applies to each circuit node (also known as a wire or net). Each component is described by a single statement specifying its label, the labels of the nodes it is connected to, and any component specific parameters needed. Note that numbers can be used as node labels. The ground node is typically labelled as "**0**". If two components are interconnected, a common node label will appear in both statements describing them. Note that statements order is not important as long as all the components are listed.

Since a resistor has two terminals, a basic resistor element statement will look as follows:

`Rname net1 net2 value`

where:

`Rname`: is a label you pick for that resistor; however, it must start by an uppercase **`R`** to indicate that this is a resistor.

`net1`: is the label of the wire connected to the first resistor terminal

`net2`: is the label of the wire connected to the second resistor terminal

`value`: is the value of the resistor in Ohm

and a basic independent DC voltage source element statement will look as follows:

`Vname net+ net- value`

where:

`Vname`: is a label you pick for that voltage source; however, it must start by an uppercase **`V`** to indicate that this is a voltage source.

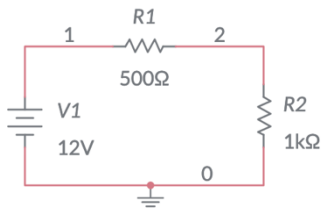
`net+`: is the label of the wire connected to the positive terminal of the voltage source

`net-`: is the label of the wire connected to the negative terminal of the voltage source

`value`: is the value of the voltage in Volt

For example, the **data statements** part describing the simple circuit below, will look like this:

```
R1 1 2 500
R2 2 0 1000
V1 1 0 12
```



Generator input:
The user can input any Boolean expression using any symbols; however, the output symbol cannot be used as an input symbol. The **&**, **|**, and **'** operators are used for AND, OR, and NOT respectively. NOT has the highest precedence while OR has the least precedence.
For example:

y=a&b|c'	is a valid input
y=y|a	is not valid

Also, it is required to allow multiple semicolon-separated output symbols in the input expressions. Implementing this feature would make the following input valid:

```
y=a&b|c'; x=y'|a|c'
```

Generator output:
The program should generate the **data statements** part of the SPICE deck describing a digital CMOS circuit that realizes the Boolean expression entered by the user. This output only consists of a listing of interconnected NMOS transistors (for the PDN) and PMOS transistors (for the PUN). Each element statement should be a line having the following syntax:

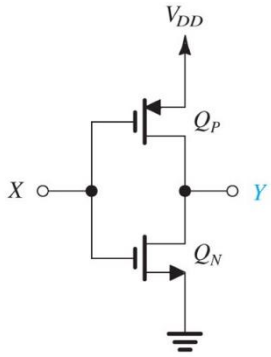
```
Mname drain gate source body type
```

where:
Mname: is a label you pick for that transistor; however, it must start by an uppercase **M** to indicate that this is a MOSFET.
drain: is the label of the wire connected to the drain of the MOS device
gate: is the label of the wire connected to the gate of the MOS device
source: is the label of the wire connected to the drain of the MOS device
body: is the label of the wire connected to the body of the MOS device. For the purpose of this project always assume that the body is connected to the source (i.e., both are connected to the same wire).
type: is either **NMOS** or **PMOS**.

Sample Interaction:
If the user inputs the following expression:

```
y=x'
```

Since this is an inverter that can be implemented using the CMOS circuit below, the program should generate the following output (or something equivalent):




```
M1 y x vdd vdd PMOS
M2 y x 0 0 NMOS
```

Simplifying assumptions:
1. Spaces are not allowed in the input expressions
2. Parentheses are not allowed in the input expressions (except as a bonus feature)
3. The described circuit has a single output symbol (except as a bonus feature)
4. Symbols in the input expression are case-sensitive. X is a different symbol from x.
5. Assume that there is always a net called vdd which represents the positive terminal of an independent DC voltage source.

***Bonus* features:**
1. Generating a full SPICE deck compatible with the free LTSpice (https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html). In this case the SPICE deck should use transient simulation mode and provide circuit inputs (such as **a**, **b**, **c**, etc.) from piecewise linear (PWL) voltage sources. The exact values provided for each of these inputs should be designed such that the entire truth table is tested. Implementing this feature will require more knowledge about SPICE decks than the information presented in the beginning of this project description. Students are expected to do some

research to bridge the necessary gap. This feature is strongly recommended and will be counted as two features as far as grading is concerned.

2. Building the application as a web-based application.
3. Building the application as a GUI application (not necessarily web-based) that would also display the described circuit schematic. This feature will be counted as two features as far as grading is concerned.
4. Allowing parentheses to appear in input expressions to override original operator precedence. Implementing this feature would make the following input valid:
   `y=((a|b)&(c|d)')'`

General Guidelines and Project Submission

- Work in a **team of 2 to 3 students**. Team members do not need to be from the same section. A team leader must be elected. She/he will be responsible for submitting the project. Any team member may interact with the course instructor and TA for any technical issue.
- Teams of 4 students are not recommended but allowed under the extra requirement to implement one of the bonus features above without it being counted as a bonus.
- **Deadline**: Thursday May 14, 2020 (last day of classes)
- **Project Submission**: Only one submission is required by team (via the group leader). The project should be submitted as a single compressed folder (ZIP/RAR file) on Blackboard that includes:
  - **Source code folder**: Your code in the general-purpose language of your choice
  - A **PDF report** that contains:
    - A brief description of the project including any bonus features included.
    - All your implementation details including the data structures and all algorithms used and of course any assumptions made.
    - A user guide including screenshots illustrating how to compile and run your program
    - A list of test cases you used to test your program. A test case is a pair consisting of a Boolean expression and its corresponding SPICE statements. You should also show the corresponding circuit for each test case.
    - A section where you clearly state the contributions of each team member to the project. You can structure this section as a log.
    - Optionally, you can include a section about your experience working on this project. This section will NOT affect your grade in any way.
- **Project Demo**: Optionally, your team might be required to make a demo of their implementation. In this case you will be directly contacted by me to schedule a demo session.

Important Plagiarism notice: You have to write your own code from scratch. Submissions based on others code will receive a grade of **zero** in the entire project (even if you understand every code line and even if the code is heavily re-factored/modified). Examples of such sources include (but is not limited to) code coming from the following sources: other teams, previous course offerings, open-source software, tutors, Internet, etc.

Grading Criteria

- Deadline: Thursday 20th of May 2021 at 11:59 PM
- The project is worth 15% of the course marks.
- Bonuses: Each bonus feature will count for 1.5 bonus marks with a maximum of 2 bonuses worth 3 marks. Please do not be tempted to implement more than 2 bonus features, as this will cost you too much time.
- Deductions:
  - -5% for late submissions (1 day only).
  - -100% for plagiarized submissions.