**UCSB ECE 594BB – Fall 2022**

# Project Assignment: MATLAB Transient Circuit Simulator

Instructor: Prof. Peng Li

**Total: 100 points**

**Due: 10pm, Monday, November 7 – Upload to Gauchospace**

## 1. Overview

There are two parts in this project:

**Part 1: (55pts)** a linear transient simulator for circuits containing linear circuit elements (R's, L's, C's and ideal current/voltage sources).

**Part 2: (30pts)** extend your linear transient simulator to a nonlinear transisent simulator to handle nonlinear MOS transistors.

Follow the general flow for nonlinear transient analysis discussed in class. Use the multi-dimensional Newton-Raphson method in the inner loop to do the nonlinear iterations. To get the correct solution at t=0, you will need to do a nonlinear DC analysis.

## 2. Matlab Circuit Parser

A Matlab circuit parser is provided as a zip file (**matlab_parser.win.2022b.zip**) that works on Matlab version 2022b on 64-bit Windows OS. Additionally, **matlab_parser.2019a.zip** has parsers that work for Matlab windows version 2019.a and an older version of Matlab on Linux. See **parser_manual.pdf** for a detailed user's manual.

## 3. MOS Transistor Models (nonlinear transient simulation)

For this assignment we use a simplified version of SPICE level 1 MOS model. The model formulae are as follows:

NMOS model:

Cut-off: $V_{gs} \le V_T$,

$$I_{ds} = 0$$

Linear: $V_{gs} > V_T \; and \; V_{ds} \le V_{gs} - V_T$,

$$I_{ds} = \mu C_{ox} \frac{W}{L}((V_{gs} - V_T)V_{ds} - \frac{1}{2}V_{ds}^2)(1 + \lambda V_{ds})$$

Saturation: $V_{gs} > V_T \; and \; V_{ds} > V_{gs} - V_T$,

$$I_{ds} = \frac{1}{2}\mu C_{ox} \frac{W}{L}(V_{gs} - V_T)^2 (1 + \lambda V_{ds})$$

PMOS model:

Cut-off: $V_{sg} \le -V_T$,

$$I_{sd} = 0$$

Linear: $V_{sg} > -V_T \; and \; V_{sd} \le V_{sg} - (-V_T)$,

$$I_{sd} = \mu C_{ox} \frac{W}{L}((V_{sg} - (-V_T))V_{sd} - \frac{1}{2}V_{sd}^2)(1 + \lambda V_{sd})$$

Saturation: $V_{sg} > -V_T \; and \; V_{sd} > V_{sg} - (-V_T)$,

$$I_{sd} = \frac{1}{2}\mu C_{ox} \frac{W}{L}(V_{sg} - (-V_T))^2 (1 + \lambda V_{sd})$$

Parasitic capacitor models:

To simplify the problem, we ignore the nonlinearity of the parasitic capacitors in the MOS model. The formulae are as follows:

Gate-source capacitance:

$$C_{gs} = \frac{1}{2}C_{ox}WL$$

Gate-drain capacitance:

$$C_{gd} = \frac{1}{2}C_{ox}WL$$

Source/drain to ground junction capacitance:

$$C_d = C_s = C_{j0}$$

The syntax for MOSFET's in our MATLAB parser is:

```
Mxxx <ND> <NG> <NS> <MOS_TYPE> <WIDTH> <LENGTH> <MODEL_ID>
```

The MOS models are specified in .MODEL cards:

.MODEL <MODEL_ID> VT <$V_T$> MU <μ> COX <$C_{OX}$> LAMBDA <λ> CJ0 <$C_{J0}$>

$V_T$ is the threshold voltage, μ is the mobility, λ is the channel-width modulator, and $C_{J0}$ is the junction capacitance.

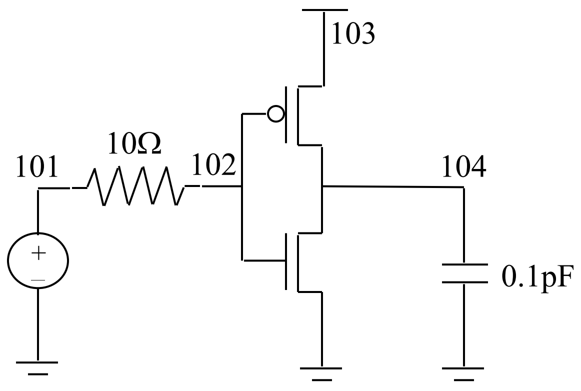
More detailed explanations of the MOSFET and MOS model syntax can be found in the MATLAB parser user's manual.

## 3. Benchmark Circuits

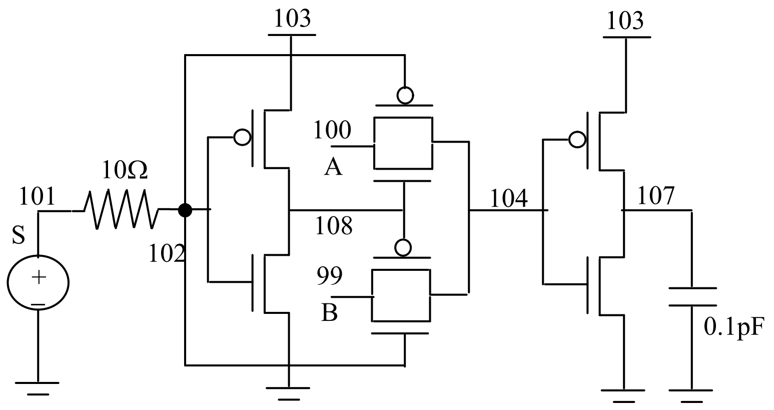
A suite of benchmark circuits are provided in the file: **benchmark_circuits.zip**. In addition, all benchmark circuits were parsed and the resulting internal Matlab data structures are provided to you as a zip file (**parsed_benchmarks_matlab_data.zip**). **In the event that the provided Matlab parsers do not work for your version of Matlab, you may directly load the pre-parsed data into Matlab without needing for running the parser**.

You are encouraged to use simple RC circuits of your own to test out your simulator and include the results in the report. For part I, you are required to verify your simulator using the linear benchmark circuits (rcmesh20.ckt, rc_line.ckt, rlc_line.ckt) found from the provided benchmark suite, which also includes the optional benchmarks for nonlinear transient anlalysis, which should be used to verify your nonlinear transient simulation.

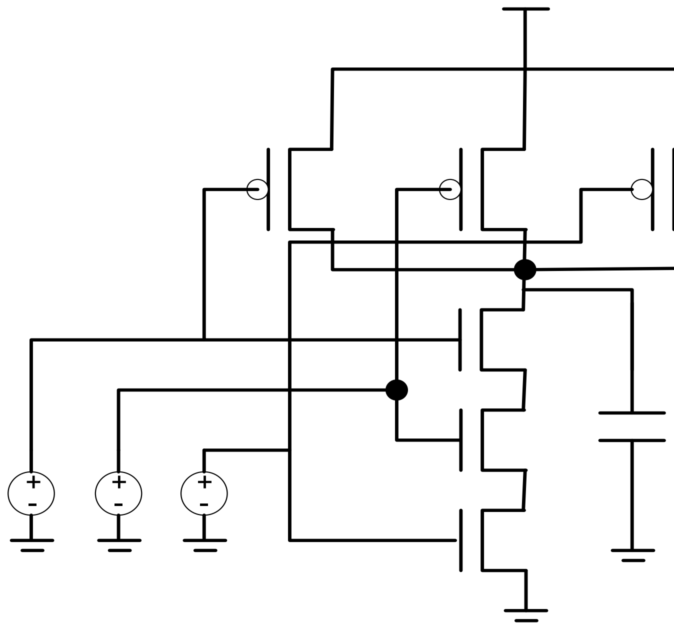
3.1 Inverter



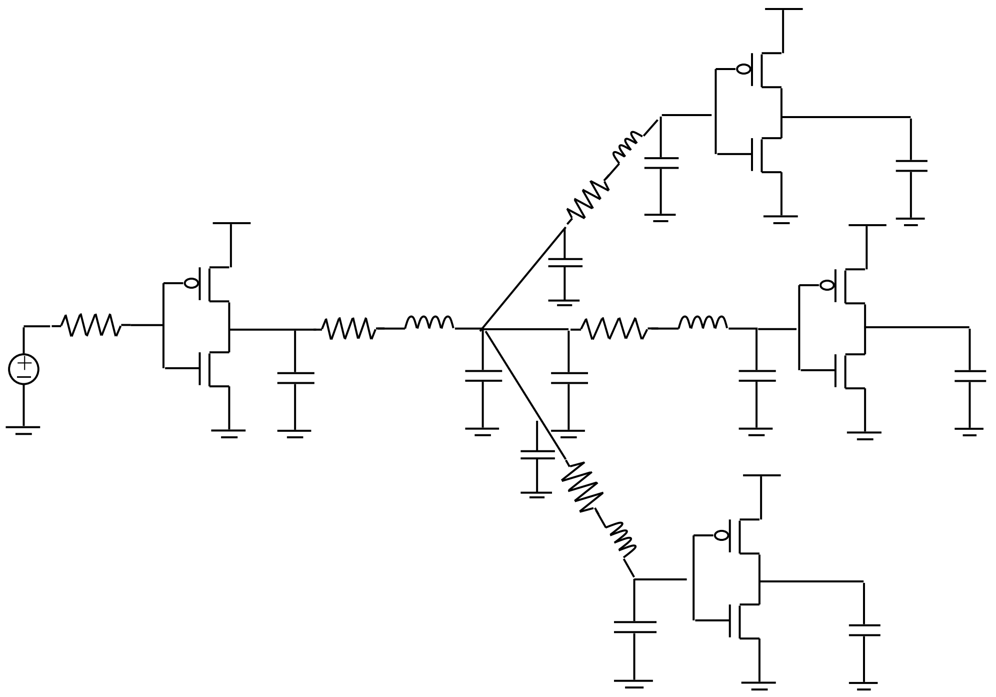
3.2 Transmission gate Mux



3.3 NAND Gate



3.4 Clock Tree



Your program should be able to successfully simulate these circuits and produce accurate results. We'll check your accuracy and runtimes.

## 4. Dynamic Time Step Control (bonus 5pts)

Dynamic time step control is optional. You can choose to use a predetermined fixed time step if you would like to skip this part. However, you're strongly encouraged to explore it as described as follows.

Dynamic adjustment of time step is used for two purposes: First, fixed time steps can cause large local truncation errors in some cases, or lead to inefficiency in others. By controlling local truncation error, the simulator can "intellegently" choose suitable time steps to improve the effeciency and accuracy.

Secondly, large timesteps can lead to convergence problems for the nonlinear circuit solution (this is due to the local convergence property of Newton-Raphson method.) If nonconvergence is detected, re-simulating the circuit with a smaller timestep can help to achieve convergence.

You are required to implement your own timestep control scheme. In order to estimate the timestep, you may need to calculate the second (or the third) derivative of capacitor voltages and inductor currents. (Hints: you may use the solution of previous timepoints to approximate these derivatives). In order to evaluate your implementation, you can fix the time step (with the value given by the .tran card) in your simulator and compare the performance between the fixed time step simulation and your dynamic time step simulation.

## 5. What to Hand In

### 5.1. Your MATLAB code (linear: 55pts / nonlinear: 30pts)

Email me a zipped/tarred file of your complete MATLAB code plus all the benchmarch circuits **before our class on the due day**. Once I unzip it, I would expect to be able to type just the following:

```
>> my_simulator_name    bechmark_circuit_name
```

under the MATLAB command line to run a simulation and your program should generate simulation plots that are specified in .PLOTXX cards in the netlist. **This is the only accepated program interface.** In other words, for example, if you ask me to run five other scripts before I can test your code on a circuit, your code will be treated as a failure.

In your zip file, please also include a README to tell what I shall specifically type (as in the above) to run your program.

### 5.2. Project report (soft & hard copies) (15 pts)

A clean report (3 pages minimum) summarizing your overall approach and hightlighting anything you've done uniquely is required. Simulation waveforms for the circuits you've successfully simulated should be provided in the report. You should clearly say what features (nonlinear DC, linear transient, and nonlinear transient) you have successively achieved, and what you could not do. If there is any issue of running your simulation on any benchmark circuit, you should report it as well. You are required to use the IEEE conference proceedings paper template (MS word or latex (preferred)) to write your report: http://www.ieee.org/conferences_events/conferences/publishing/templates.html.

Up to 15 points will be given to the report based on its overall quality.

The hardcopy of the report should be submitted in class on the due day. The softcopy should be emailed as part of the above zip/tar file.

## 6. Grading Policy

### 6.1. Factors considered for grading

- **Quality/success of your code**
- **User interface**: You do not have to implement any fancy (graphic) interface. However, you should provide introductions on how to run your code. You should provide an easy interface through which your code can be conveniently run as specified earlier.
- **Quality of report**: a clean, well-written report is expected.
- **Frankness**: In the report, clearly and frankly state whatever you couldn't achieve. For example, if your simulator can only do linear transient analysis but not nonlinear DC & transient, say it. If your simulator doesn't yield the correct results or cannot converge for some benchmarks, say it too. Remember, your code will be tested so I will find out these problems one way or another.

### 6.2. Late/incomplete submission

**25% penalty per calendar day** after the due date. An incomplete submission is considered as "late" till it becomes complete.