UCSB ECE 594BB Winter 2022
Modeling and Simulation of VLSI Circuits

Final Class Project

Instructor: Prof. Peng Li

1. Overview

First of all, this is NOT your typical homework assignment! The final class project is meant to offer students the opportunity to work on less well defined research-type problems on topics related to the themes of this course. Because of this reason, you shall be prepared to:

1) Define very specifically what your project goals are by using the handout as a basic guideline;

2) To achieve your goals, come up an effective and feasible approach with a solid implementation/experimental plan;

3) Be able to summarize your findings under a technical context professionally in the form of a project presentation and report.

Welcome to the real world. Be risk taking, stick to your goals, be strategic, and enjoy some hardship plus perhaps a whole lot of fun.

3. Suggested Project Topics

As mentioned before, you should clearly define what your specific goals are starting from the basic description provided for each topic. The references, if any, are only offered to get you started. They are not necessarily the only and/or the best materials you shall read for your specific project goals. More pointers to literature may be provided upon request.

You can choose on of the following topics to work on. It is also possible to come up with a topic on your own provided that: 1) the topic has a key simulation/modeling component, 2) you consult the instructor and obtain his permission.

- **Topic 1: General-Purpose Transistor-Level Circuit Simulator in C/C++ or other high-performance/maintainable programming languages**

The goal is to extend the experiences you gathered from the Matlab simulator development towards the realization of a more realistic general-purpose transistor-level simulator in C/C++ or some other programming language that is of high performance or can be maintained. This will allow you to more accurately measure the true simulation performance and target much larger benchmark circuits. The

developed simulator should be self-contained, i.e. it should have a (simple) parser and can be run on the standard Linux OS. The simulator can interface with simple level-like device models and make use of an on-the-shelf matrix solver.

References

[1]Sparse 1.3 matrix solver: http://www.netlib.org/sparse/

- **Topic 2: Optimized Parallel Matrix Solver for Large-scale Transistor-Level Circuit Simulation**

The goal here is to develop a highly efficient parallel sparse matrix solver in C/C++ for large-scale transistor-level circuit simulation. The targeted circuit size should be from medium to large (e.g. >100K nodes). Transistor-level schematics with extracted power nets/parasitics are good examples. The developed solver should be rather general without making any unrealistic assumptions about the matrix (such as assuming the matrix is symmetric). The solver should be applicable to typical matrices extracted from various types of circuits. The developed solver can be built upon a basic existing public domain solver, which, for example, provides the basic function of LU factorization. Your main focus would be to start from this basic solver, and add techniques on top of it, and eventually come up an efficient sparse parallel solver.

Your solver should be tested using matrices of various types and sizes. Runtimes and memory usages at different levels of parallelism (e.g. number of threads) should be carefully optimized and analyzed.

[References]

[1] Distributed voltage regulation in IBM Power8 Processor: http://semiengineering.com/the-good-kind-of-regulation/

[2] Peng Li, "Parallel Circuit Simulation: A Historical Perspective and Recent Developments," Foundations and Trends in Electronic Design Automation: Vol. 5: No 4, pp 211-318, 2011 (invited).

[3] Saad, *Iterative methods for sparse linear systems* (2nd edition): http://www-users.cs.umn.edu/~saad/books.html.

[4] X. Zhao, L. Han, and Z. Feng, "A Performance-Guided Graph Sparsification Approach to Scalable and Robust SPICE-Accurate Integrated Circuit Simulations", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 10, pp. 1639-1651, Oct. 2015.

- **Topic 3: Optimized Parallel Sparse Matrix Solver for Large-scale Linear Circuit Analysis**

There exist many large linear circuits (e.g. on-chip power grids) which must be accurately analyzed in the design process. Such circuits can be extremely large. For instance, extracted industrial power nets (modeled as RC circuits) can have more than a billion nodes.

In parallel to Topic 2, this topic concerns with developing efficient sparse parallel matrix solvers in C/C++ for large passive linear circuits (with >1 million nodes). Again, your solver can be built on top of a basic public domain LU-based matrix solver.

Note that the matrix for DC/transient analysis of linear circuits can be formulated to possess special properties, e.g. being symmetric or SPD (symmetric positive definite). You're encouraged to leverage these properties to improve the efficiency of parallel solution.

Again, your solver should be tested using matrices of various types and sizes. Runtimes and memory usages at different levels of parallelism (e.g. number of threads) should be carefully optimized and analyzed.

[References]

[1] Peng Li, "Parallel Circuit Simulation: A Historical Perspective and Recent Developments," Foundations and Trends in Electronic Design Automation: Vol. 5: No 4, pp 211-318, 2011 (invited).

[2] Saad, *Iterative methods for sparse linear systems* (2nd edition): http://www-users.cs.umn.edu/~saad/books.html.

[3] X. Zhao and Z. Feng, "An Efficient Spectral Graph Sparsification Approach to Scalable Reduction of Large Flip-Chip Power Grids", to appear in Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), November, 2014.

[4] Synopsys PrimeRail: http://www.synopsys.com/Tools/Implementation/SignOff/Pages/PrimeRail.aspx

Available at the ECE Department: https://engineering.tamu.edu/electrical/employee-resources/unixlinux-resources-helpdesk/sourcing-software

▪ Topic 4: Software Engineering for Large-Scale Circuit Simulation

The goal of this topic to help the students to acquire practical software engineering skills for developing real-life VLSI CAD tools. While the following two options are not compute-centric, the quality of software engineering and code development is strongly emphasized.

Option 1: Efficient Hierarchical Parser for Large Circuits

Parsing SPICE circuit netlists is a critical step in circuit analysis tools. Parsing very large circuits (e.g. with >100 million nodes/devices) can consume excessive CPU time and large amount of memory/disk storage. Circuit parsing becomes nontrivial for very big circuits, and requires efficient data structures and management of memory/disk space. In addition, practical circuits are often hierarchical in nature. The developed parser should support sub-circuits embedded in the netlist.

The expected quality and flexibility of the developed hierarchical parse would be consistent with the team composition (individual vs. group).

Option 2: Advanced Graphical User Interface (GUI)

The focus here is to develop a Linux-based GUI for supporting visualization of simulation results such as current and voltage responses and their correspondence to the input excitation.

The expected sophistication and usability of the developed GUI would be based on the team composition (individual vs. group).

▪ Topic 4: Computational Modeling and Analysis of Large Brain Models

Mammalian brains are large and complex dynamical systems comprising a huge number of neurons and interconnections (e.g. dendrites and axons) between the neurons. Computer-based modeling and simulation of the brain is very meaningful. It may help shed light on basic brain functions, for example, motor control, learning and memory. It may also offer a very useful *in silico* platform for understanding various brain disorders (e.g. epilepsy and Parkinson's disease) and aiding the development of therapeutic treatments.

The brain network is yet another circuit although its size is much larger than our typical integrated circuits. Neurons and axons are analogous to transistors and wires in an electronic circuit. The behavior of the brain is also a lot more complex and mysterious – it is highly nonlinear and oscillatory.

Option 1: This option aims to develop a generic brain simulation infrastructure that provides modeling of basic neuron, dendrite and axon models and allows simulation of general brain networks. The main emphasis is the generality and flexibility of the environment; the key technical focus is to identify, adopt and develop efficient algorithms and implementations for simulation of large brain models.

While C/C++ is highly recommended for coding, MATLAB is also an option.

Option 2: Different from Option 1, the key focus of this option is to develop models and simulation capability for understanding a meaningfully chosen behavior or disease of the brain. Generality is not an issue here; it will be sufficient to just model the specific part of the brain that is meaningful for your target. But you may want to demonstrate experimentally through simulation how the developed capability for understanding targeted behaviors such as oscillatory behaviors.

[References]

[1] Eugene M. Izhikevich and Gerald M. Edelman, "Large-scale model of mammalian thalamocortical systems," PNAS, March 2008.

[2] Y. Zhang et al, "Linking brain behavior to underlying cellular mechanisms via large-scale brain modeling and simulation," Neurocomputing, volume 97, 15 November 2012, pages 317-331.

[3] Brain 2.0 simulator: http://briansimulator.org/

- **Topic 5: Model Reduction of Large-Scale Linear Passive Circuits**

As discussed in the class, model order reduction (MOR) can go a long way to speed up analysis of complex linear circuits and systems. This topic allows you to implement a given MOR algorithm, verify the accuracy of the produced reduced order models, and demonstrate the use of such reduced models for speeding up time-domain simulation. Targeted algorithms can be AWE, PRIMA, or recent developments (preferred) in the field. Implementation in MATLAB is allowed, however, implementation in C/C++ or other high-performance programming languages is encouraged given the runtime efficiency consideration. **If choosing MATLAB, your implementation must provide new features/capabilities when comparing with your solutions to Homework 3.**

You can use the circuits in the first matlab project for basic benchmarking. Adding additional test circuits is encouraged.

4. What to submit and timelines

1) Milestone 1: (Due on 5pm, Wednesday, November 9. No partial credit after the deadline)

Submit a two-page proposal (both a hardcopy and softcopy (email); You need to clearly state the targeted problem, provide relevant backgrounds, and argue why this is a meaningful problem and state what will be done and how. The proposal must be written by using the standard IEEE journal template: https://www.ieee.org/publications_standards/publications/authors/author_templates.html.

Milestone 1 constitutes **15%** of the credit for the final project.

2) Milestone 2: (Due at 5pm, Monday, November 28. No partial credit after the deadline)

Milestone 2 constitutes the remaining **85%** of the credit for the final project. To complete this milestone, you need to submit ALL the following items by the deadline:

1. Report: You need to well document your project in a nicely written report. The report must be rewritten using the standard IEEE journal template: https://www.ieee.org/publications_standards/publications/authors/author_templates.html. This is the ONLY accepted form of your report. As in a technical paper, your report shall follow a structure similar to the following:

Abstract

Introduction

Major body

Experimental Results

Conclusion

A softcopy of the report shall be uploaded to Gauchospace by the deadline. **A hard copy should be submitted during the class project presentation later**.

2. Code, hardware design files, and benchmarks: You need to submit electronically your code in Matlab, C/C++ or other programming/hardware design languages, and benchmark circuits/networks and experimental setups that can be used to test your code/design. You shall provide instructions/documentations on how to install and run the code/design. These shall be uploaded to Gauchospace.

3. PPT presentation: Each individual/team needs to present the final project using **no more than 8 PPT content slides within 8 minutes**, followed by 2-minitue Q & A. The PPT presentation shall be uploaded to Gauchospace.

Mandatory class project presentations will be scheduled on November 29 or 30 (time & location TBA).

6. Grading

The final project is structured in such a way to provide the students the opportunity to apply and extend the techniques covered in the class, and engage in research exploration. As part of these objectives, each of you will be also provided the opportunity to document your work in the form of a project report and present your results to others in the form of a short presentation. The grading of **milestone 2** will be conducted by looking into the following aspects:

Soundness and quality of the results (45%): The proposed ideas and implementations should be technically sound and solid. The goals of your project should be very clearly specified and the claimed results should be backed up by convincing experiments.

Novelty (15%): We also look into novelty. So, let us know what's cool and what may be potentially considered as new contributions.

Quality of the report (15%): Think about how to nicely describe your work as in a technical paper. You should provide a basic context of your targeted problem, prior work and specific goals. Then, you should clearly present your proposed approach, followed by the achieved results and experimental validation.

Quality of the representation (10%): We look into clarity and professionalism. Think about how to best highlight your project within the time limit – you don't have much time and better use your time wisely! Well stick to your time budget. Going overtime will negatively impact your grade.

Level of challenge (15%): To be fair and also to encourage risk-taking spirit, your choice of work will be given a *challenge factor*, indicating its relative level of difficulty. This is based upon the excepted amount of effort required and how well defined each option is, etc. The final grade for the project is calculated as:

*Final project grade: = (soundness_quality + novelty + report + presentation + level_of_challenge) * 85% + grade_milestone1.*

Note that your performance under the above categories will be evaluated relatively to all other students, i.e. not just the students who will have chosen to work on the same option.