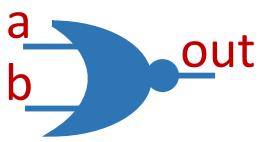
DeepSPICE: Accelerating Digital Cell Characterization Using Deep Learning

Ishwar Suriyaprakash
Homestead High School, Cupertino, CA

INTRODUCTION

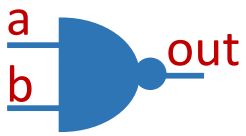
- Digital Integrated Circuits (IC or chips) implement complex arithmetic & logic functions in silicon
- Circuits created using network of building blocks called cells (or gates) from a collection (library)
- Cells implement smaller primitive Boolean functions such as NOR, NAND

NOR cell



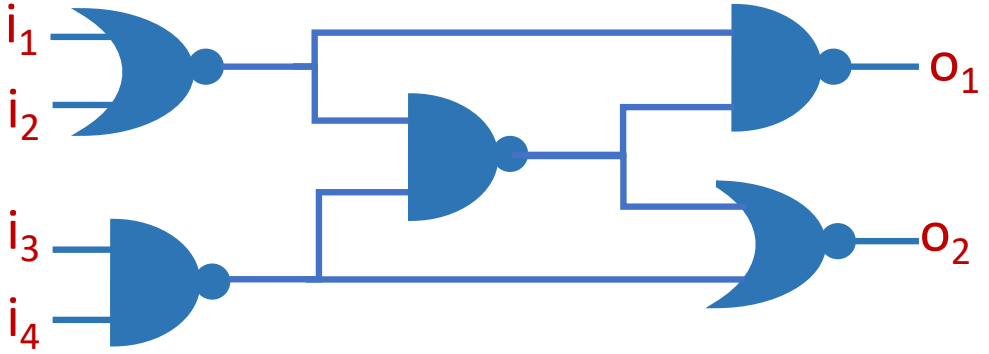
a	b	out
0	0	1
0	1	0
1	0	0
1	1	0

NAND cell



a	b	out
0	0	1
0	1	1
1	0	1
1	1	0

Circuit for a complex function implemented with cells



- Circuit validation includes determining properties such as power consumed and speed
- Circuit speed: Propagation times of events (0→1 or 1→0) from inputs to outputs
- Method to determine circuit speed is hierarchical
  1. Input to output propagation delays of each cell type determined using simulations
  2. Worst case circuit propagation times are calculated from pre-computed cell delays
- Cell simulations (Step 1 above) to determine delays are computationally very expensive

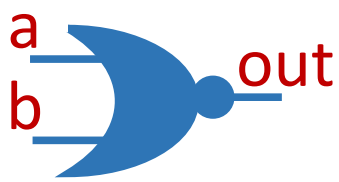
Goal: Compute delays with cell simulation time reduced by 2X using deep learning
Delay prediction error should be within 15% of average simulation delays
Method should work for cells with inputs ranging from 2 to 7

INTRODUCTION

Why are cell simulations expensive?

- Number of different cell types ~30K in a modern cell library
- Number of inputs per cell can range from 1 (e.g. inverter) to 20
- Determining cell input-to-output delays involves transistor-level simulations
- Event combinations at cell's inputs need to be simulated to determine delays

Events: 0→1, 0→1, 1→1, 1→0 x 0→1, 0→1, 1→1, 1→0



2-input cell
$2^2 \times 2^2 = 16$ combinations

Event 0→1 called rise transition
Event 1→0 called fall transition
Event 0→0 called steady 0
Event 1→1 called steady 1

- Number of simulations for k-input cell = $2^k \times 2^k = 2^{2k} = 4^k$
- Example: Simulations for 12 input cell = 4^{12} = 16.8 million
  - Assuming 1 second per simulation, this would take ~194 days!
  - In practice, large number of parallel machines are used to contain this cost

DeepSPICE approach: Learn from simulations on a small subset of input event combinations & predict the delays for the rest of the event combinations

- Recent research focused on learning delays from transistor behavior at one part of silicon wafer (on which circuits are manufactured) using full set of simulations and predicting those delays in another part of that silicon wafer
- Those works did not focus on reducing the number of input event combinations

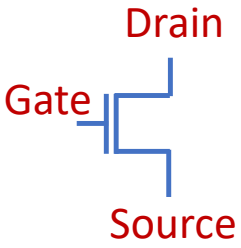
DeepSPICE is the first to estimate cell delay by learning from subset of input event combinations

INTRODUCTION

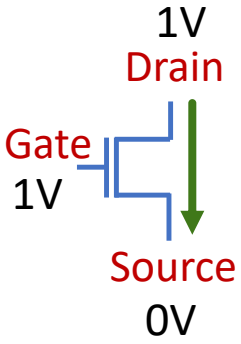
How are cells constructed?

- Cells implemented using Metal Oxide Semiconductor (MOS) 3-terminal transistor switches
- Complementary MOS (CMOS) technology uses two types of transistors: NMOS and PMOS

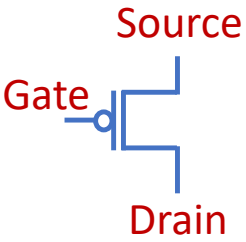
NMOS transistor



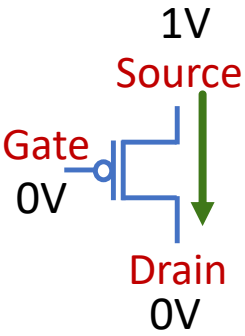
Conduction from Drain to Source established when V(Gate, Source) > V_{th}



PMOS transistor

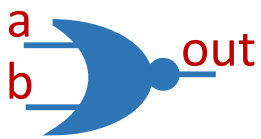


Conduction from Source to Drain established when V(Source, Gate) > V_{th}

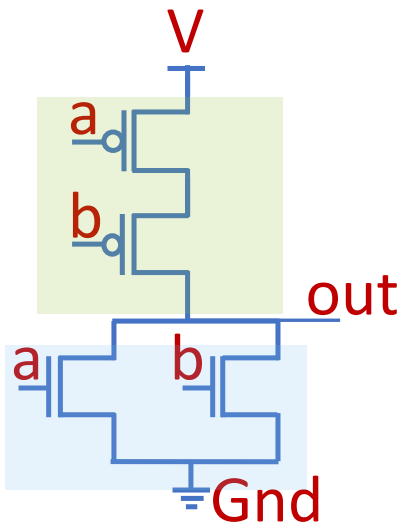


- Cells are implemented using PMOS and NMOS transistor networks
- PMOS network implements out = 1 by establishing charge path to out from power supply
- NMOS network implements out = 0 by establishing discharge path to ground from out

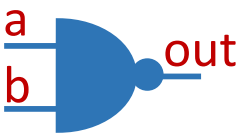
NOR cell



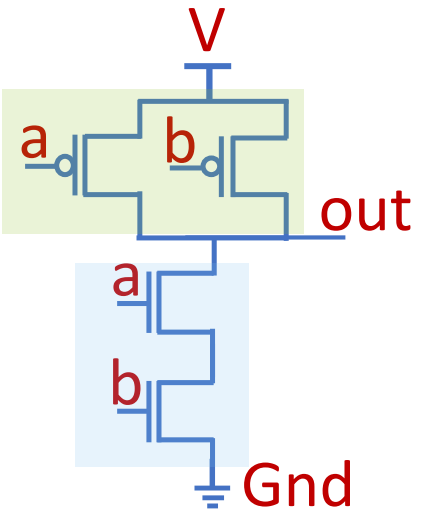
a	b	out
0	0	1
0	1	0
1	0	0
1	1	0



NAND cell

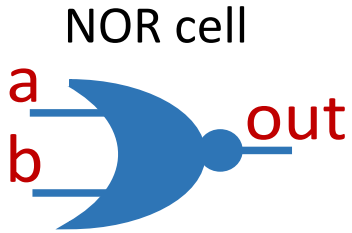


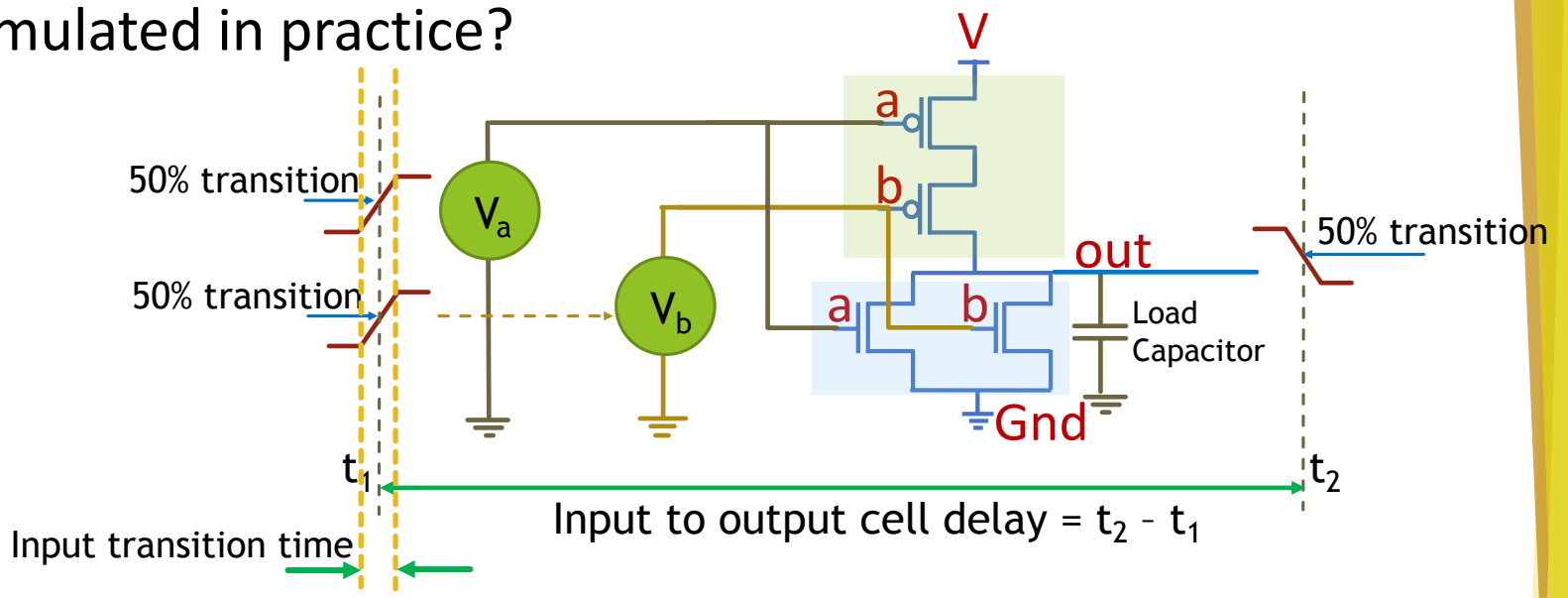
a	b	out
0	0	1
0	1	1
1	0	1
1	1	0



INTRODUCTION

How are cells simulated in practice?



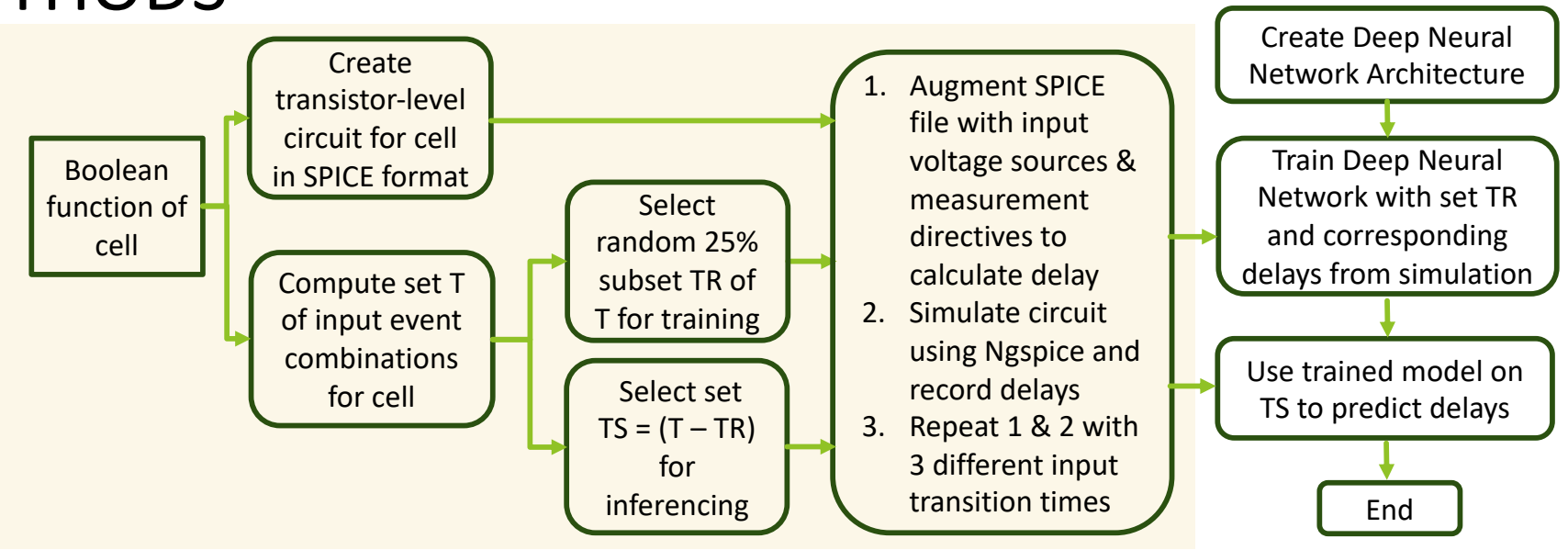


- Transistor-level circuit of cell, comprising of PMOS and NMOS network, is used for simulation
- Inputs are connected to voltage sources
- Voltage sources are setup to apply piece-wise linear (PWL) waveforms to apply events
  - Each input event combination is a different set of PWL waveforms at inputs
- Output is connected to a load capacitor
- SPICE[1] file format used to represent the connections of transistors, inputs and output
- Process technology file that describes transistor model parameters included for simulation
- Transistor-level simulator (HSPICE[2], SPECTRE[3], NGSPICE[4]) used for simulating each cell
  - Simulator solves equations (nonlinear, Kirchhoff's) to compute voltage, current values
  - Voltage & current values are computed for nodes within cell for each time step
- Input-to-output delay for each input event combination obtained through this simulation
- For each input event combination, delays obtained for different input transition (rise, fall) times

1 – Simulation Program with Integrated Circuit Emphasis

2 – Synopsys, 3 – Cadence, 4 - Public

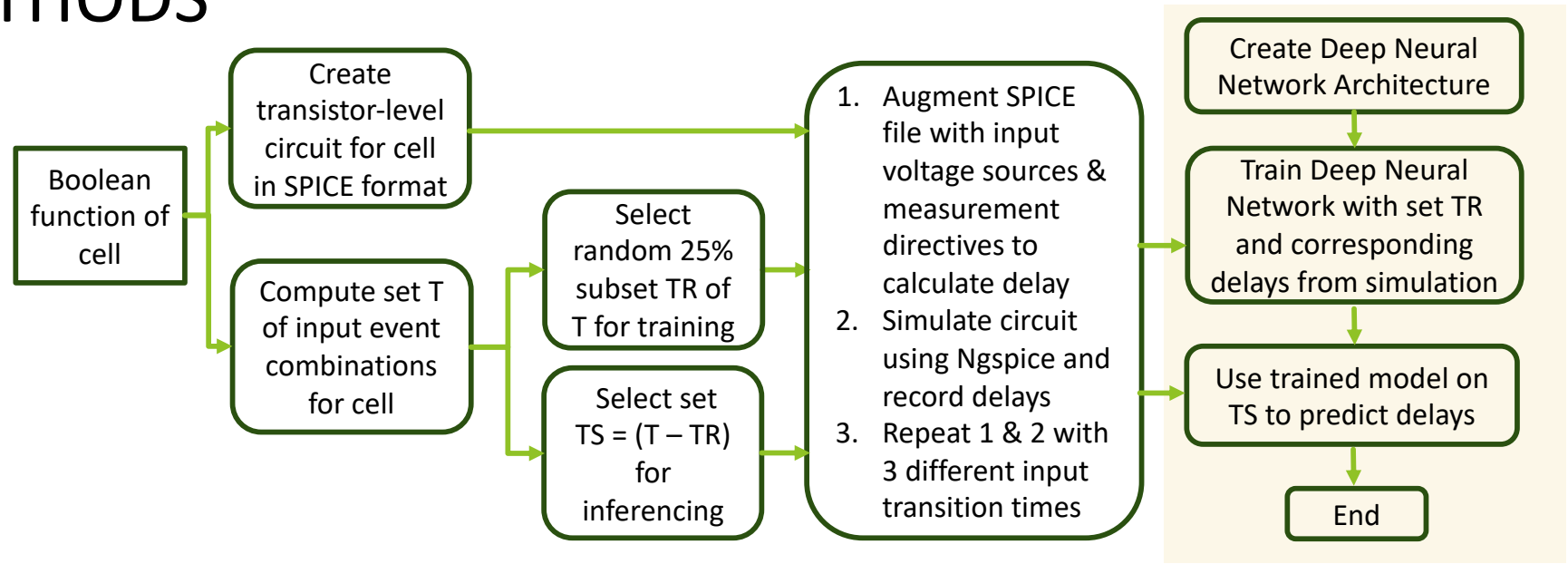
METHODS



Transistor-level simulations performed to create the baseline results

- 14 cells that implement Boolean functions created for experiment
  - 2 2-input, 3 3-input, 3 4-input, 3 5-input, 2 6-input and 1 7-input cells created
- CMOS transistor network for each cell implemented in SPICE file format
- Load capacitor of 5fF connected at each cell's output
- Temperature of 27C used as set-point for simulation
- Length of each transistor set at 0.18 micron, width adjusted based on cell type
- Three different input transition times, 10ps, 20ps and 30ps, chosen for experiment
- Simulations performed with publicly available NGSPICE transistor-level simulator
- Number of input event combinations, NIC = (Number of truth table rows with output 1) x (Number of truth table rows with output 0) x 2 - *multiplied by 2 for output rise & fall*
- Total number of baseline simulations, BS = NIC x 3 - *multiplied by 3 for 3 different input transition times*

METHODS



Deep Learning performed to learn and predict from subset of simulation results

- Deep Neural Network (DNN) built with dense layers in Keras
  - Neurons/layer & layers scaled based on number of cell inputs
  - ReLU function used as non-linearity in a neuron
- DNN trained with data from a subset of simulation results forming the training set TR
  - Training features are initial voltage, final voltage, and transition time value at each input
  - Training output is the measured input-to-output delay for each input combination from simulation
  - Trained model is used to predict input-to-output delays for input combinations in test set TS

Metrics for measuring DeepSPICE goodness

- **Error (%)** = 100 x NRMSE = $100 \text{ x } \frac{RMSE \text{ between simulated and predicted delays for TS}}{Mean \text{ delay of test set TS}}$

NRMSE = Normalized root mean square error

- Time for baseline = Time for simulating all combinations for a cell
- Time for DeepSPICE = Time for simulating training subset TR + Training time + Prediction time for TS
- **Acceleration Factor** = $\frac{Time \text{ for baseline}}{Time \text{ for DeepSPICE}}$

RESULTS

- For each cell, DeepSPICE was performed with the following options
  - Training with (a) 25% subset of simulation results, and (b) with 30% subset of simulation results
  - For each training option above, 3 trials to select a different random training set
  - For each trial above, 3 DNN training & prediction runs with different initial DNN starting states
    - Only the mean results from the 3 DNN runs for each trial is reported in the table below
- Cells are ordered in the table below in non-decreasing order of number of inputs

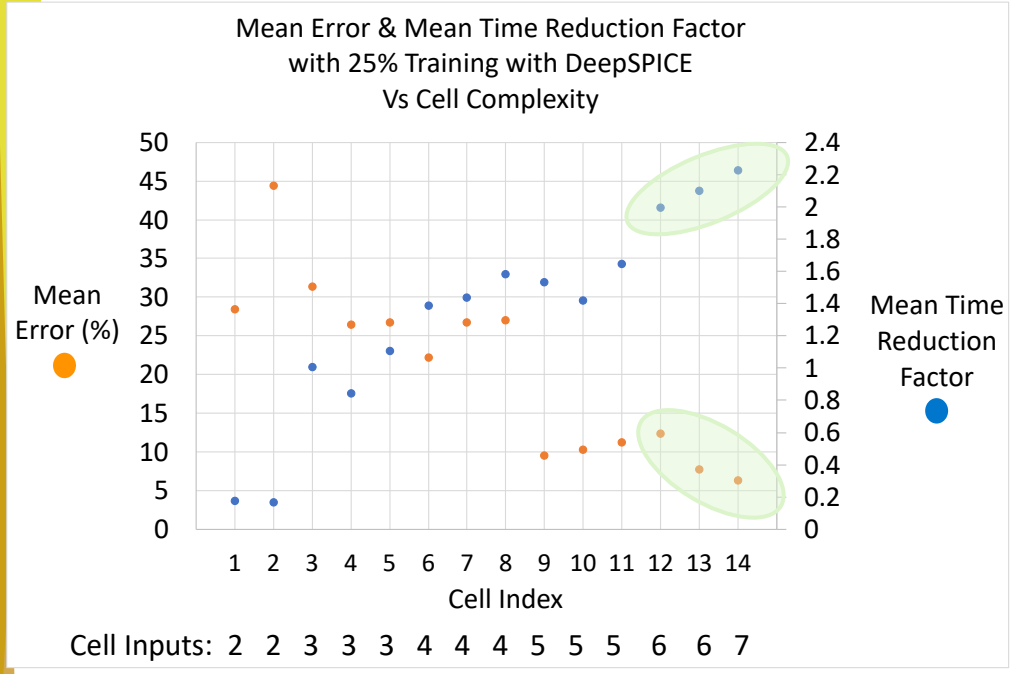
Cell properties & Baseline time					Training with 25% of simulation results						Training with 30% of simulation results					
					Trial 1		Trial 2		Trial 3		Trial 1		Trial 2		Trial 3	
Cell	Inp	Tran	BS	Time for baseline (sec)	Time DS (sec)	Error (%)	Time DS (sec)	Error (%)	Time DS (sec)	Error (%)	Time DS (sec)	Error (%)	Time DS (sec)	Error (%)	Time DS (sec)	Error (%)
1	2	4	18	0.40	2.20	25.25	2.52	27.37	2.26	32.77	2.26	33.51	2.18	40.74	2.51	38.39
2	2	4	18	0.36	2.10	43.89	2.13	40.97	2.14	48.46	2.28	38.78	2.11	58.57	2.25	35.35
3	3	6	90	2.97	2.92	33.69	2.95	28.95	2.97	31.39	3.06	32.75	3.17	28.22	3.08	27.59
4	3	6	90	2.22	2.74	30.39	2.56	26.70	2.61	22.28	2.86	30.08	2.61	27.86	2.79	24.31
5	3	10	96	2.89	2.55	28.58	2.66	25.50	2.62	26.16	2.73	28.48	2.84	25.30	3.01	26.47
6	4	10	384	11.21	7.94	22.43	8.03	17.74	8.30	26.39	9.96	13.83	9.99	18.95	9.95	19.32
7	4	12	378	10.62	7.75	25.36	7.13	27.16	7.33	27.64	9.61	26.60	9.99	24.24	10.15	21.91
8	4	12	360	12.80	8.02	26.05	7.71	25.76	8.55	29.13	9.77	25.12	10.46	16.49	10.16	26.94
9	5	10	1530	45.16	29.54	9.64	29.47	10.10	29.28	8.69	36.33	8.82	35.27	8.58	35.93	7.88
10	5	12	1512	39.61	27.95	9.70	27.82	9.95	27.85	11.24	34.64	8.93	34.23	8.11	35.46	8.82
11	5	16	1536	49.52	29.70	13.14	29.55	9.76	30.97	10.68	38.03	9.15	37.03	8.98	37.34	9.30
12	6	22	5760	246.55	122.53	16.07	125.60	5.30	122.41	15.63	146.91	4.94	148.29	4.59	146.69	4.68
13	6	30	6120	309.54	147.19	7.11	148.01	8.99	147.14	6.95	179.00	6.55	176.51	8.08	177.44	8.33
14	7	30	24426	1404.80	631.54	6.57	632.72	6.53	628.36	5.82	753.06	5.74	758.49	6.85	752.47	6.24

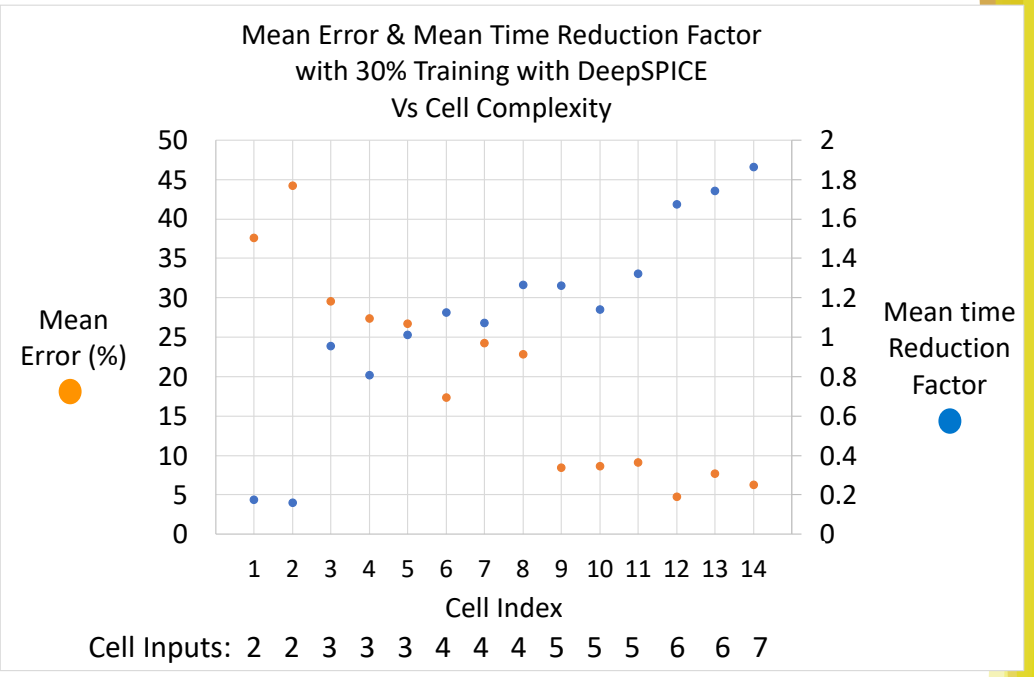
Inp - Number of inputs Tran - Number of transistors BS - Total number of baseline simulations

Time DS – Time for DeepSPICE

RESULTS

- Comparison of the mean results over the 3 trials for each training option is shown in the 2 graphs below





Results with 25% training subset

- For 3 largest cells (12, 13, 14) with 6 and 7 inputs, DeepSPICE achieves goal of >2X acceleration & < 15% error
- For 5-input cells, DeepSPICE achieves <15% error goal but achieves only under 1.7X acceleration
- For 3 & 4-input cells, DeepSPICE, most have >20% error with limited acceleration on 5/6 cells
- For 2-input cells, DeepSPICE takes at least 5X more time than baseline and has >25% error

Results with 30% training subset

- For 3 largest cells DeepSPICE achieves <10% error but acceleration is below 2X goal
- For 5-input cells, DeepSPICE achieves <10% error but achieves acceleration of >1.2X only on 2/3 cells
- Overall, the trend is similar to 25% training but with less error and less acceleration (due to increased training)

DISCUSSION

- DeepSPICE shows promising results especially for large cells for which simulations are expensive
- Results on small cells are poor – this is somewhat expected
  - Small cells have few baseline simulation combinations to begin with
  - Training with an even smaller set of simulations can lose significant information
  - Baseline approach is preferred for small cells and is computationally feasible
- Intuition behind DeepSPICE is validated
  - Possible to learn about charging and discharging paths within cells from few simulations

- Other research on accelerating cell simulations don't aim to reduce input simulation combinations

- Several challenges encountered that were overcome
  - Finding a free transistor-level simulator (NGSPICE) and learning how to use a circuit simulator
  - Creating Boolean functions for cells that can best validate DeepSPICE
    - For a given number of inputs for a cell, it took time to find Boolean functions that maximize baseline simulation combinations
    - Boolean function should also be implementable without any additional inverters at inputs
  - Creating transistor-level circuits for cells and setting up simulations in NGSPICE
  - Manually running the simulations and Deep Learning was difficult initially
    - So the entire flow was automated using Python to perform simulations, training and predictions

CONCLUSIONS

- Exciting to observe DeepSPICE performing well at predicting delays of larger cells
- Plan to investigate modifications to approach to reduce error further without increasing time
  - Ex: Use time series of input waveforms as training data in place of initial and final voltages & transition time
- DeepSPICE goodness has to be evaluated for cells that include resistances and capacitances
  - Current evaluation was on cells that contain only transistors with interconnecting wires as ideal conductors
  - Actual cell implementations have wires having resistances and node-pairs within cells having capacitances
- Approach can be extended to different applications
  - To predict power consumption of cells
  - To predict delays for large circuits composed of interconnection of cells
- Plan to explore Recurrent Neural Networks in place of DNN to determine improvement in goodness

- Project was an exciting learning experience on multiple concepts
  - Digital design process, Cells and CMOS networks, Transistor-level simulation
  - Deep Learning with Keras

ACKNOWLEDGMENTS

- Many thanks to my math teacher, Mr. Greg Burroughs, for his feedback and suggestions.

REFERENCES

1.Batten, Christopher. ECE 5745 Complex Digital ASIC Design.
https://www.csl.cornell.edu/courses/ece5745/handouts/ece5745-T05-methodology-auto.pdf.

2.Černý, David, and Josef Dobeš. “Deep Learning Neural Network Algorithm for
Computation of Spice Transient Simulation of Nonlinear Time Dependent Circuits.” Electronics, vol. 11, no. 1, 2021, p. 15., https://doi.org/10.3390/electronics11010015.

3.Lee, Chuan-Zheng. Transistors.
https://web.stanford.edu/class/archive/engr/engr40m.1178/slides/transistors.pdf.

4.NGSPICE: Circuit Simulator - Oregon State University.
https://web.engr.oregonstate.edu/~traylor/ece391/smith_NGSPICE_USERGUIDE_ECE391.pdf.

5.“PrimeSim HSPICE the Gold Standard for Accurate Circuit Simulation.” The Gold
Standard for Accurate Circuit Simulation, https://www.synopsys.com/implementation-and-signoff/ams-simulation/primesim-hspice.html.

6.Rosenbaum, E., et al. “Machine Learning for Circuit Aging Simulation.” University of
Illinois Urbana-Champaign, Institute of Electrical and Electronics Engineers Inc., 12 Dec. 2020, https://experts.illinois.edu/en/publications/machine-learning-for-circuit-aging-simulation.

7.“Spectre Simulation Platform.” Cadence,
https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-simulation-platform.html.

8.Tan, Wei-Lii. “Machine Learning Overcomes Library Challenges at the Latest Process
Nodes.” Tech Design Forum Techniques, https://www.techdesignforums.com/practice/technique/machine-learning-overcomes-library-challenges-at-newer-process-nodes/.

9.“What Is Library Characterization? – How It Works & Techniques.” Synopsys,
https://www.synopsys.com/glossary/what-is-library-characterization.html.