International Institute Of Information Technology, Hyderabad

Project Report Team MP-36

Mentor: Prof. Zia Abbas

Kushagra Agarwal - 2018113012 (CND)

Date: 19/07/2020

Aryamaan Jain - 2019121002 (LCD)

MOSFETs Delay And Leakage Values Optimisation

ABSTRACT

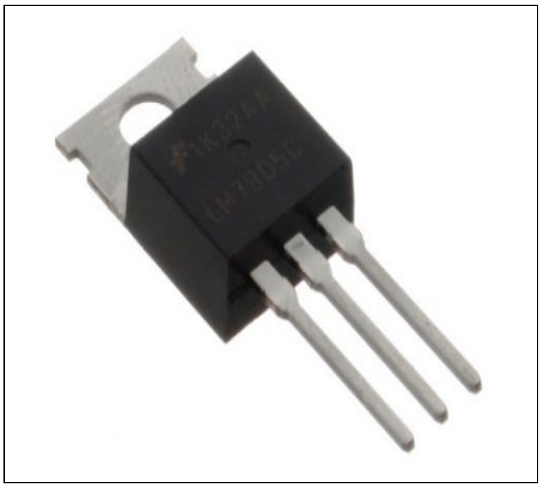
Through Simulation softwares, the Delay and Leakage values for a Full Adder circuit of MOSFETs were computed for 50000 different Length and Width parameters for all the 28 transistors. Using these values as training data, we created a Neural Net Model implemented on the Keras framework to predict the delay and leakage values for unknown L and W parameters. Then we applied Genetic Algorithms and Particle Swarm Optimisation techniques to find the best suitable 56 values (28 L and 28W) for which the Delay and Leakage values were the least.

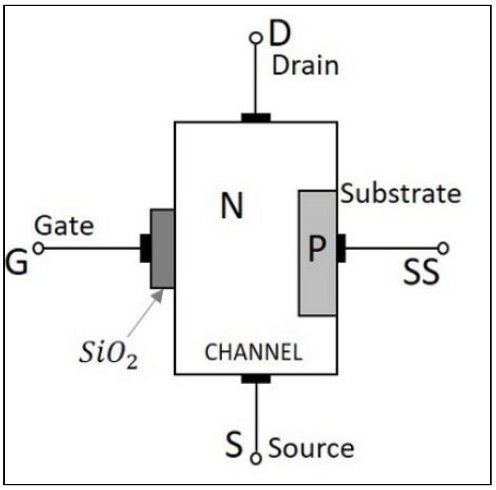
INTRODUCTION

What are MOSFETs and why are they important?

FETs have a few disadvantages like high drain resistance, moderate input impedance and slower operation. To overcome these disadvantages, the MOSFET which is an advanced FET was invented.

MOSFET stands for Metal Oxide Silicon Field Effect Transistor or Metal Oxide Semiconductor Field Effect Transistor. This is also called IGFET meaning Insulated Gate Field Effect Transistor. The FET is operated in both depletion and enhancement modes of operation. The following figures show how a MOSFET looks.





How is a MOSFET constructed?

The construction of a MOSFET is a bit similar to the FET. An oxide layer is deposited on the substrate to which the gate terminal is connected. This oxide layer acts as an insulator (SiO_2 insulates from the substrate), and hence the MOSFET has another name as IGFET. In the construction of MOSFET, a lightly doped substrate is diffused with a heavily doped region. Depending upon the substrate used, they are called as P-type and N-type MOSFETs.

What is Delay in a circuit?

In electronics, digital circuits and digital electronics, the propagation delay, or gate delay, is the length of time which starts when the input to a logic gate becomes stable and valid to change, to the time that the output of that logic gate is stable and valid to change. Often on manufacturers' datasheets this refers to the time required for the output to reach 50% of its final output level when the input changes to 50% of its final input level. Reducing gate delays in digital circuits allows them to process data at a faster rate and improve overall performance. The determination of the propagation delay of a combined circuit requires identifying the longest path of propagation delays from input to output and by adding each tpd time along this path.

What are leakage values?

In electronics, leakage is the gradual transfer of electrical energy across a boundary normally viewed as insulating, such as the flow of current across a transistor in the "off" state or a reverse-polarized diode. In semiconductor devices, leakage is a quantum phenomenon where mobile charge carriers tunnel through an insulating region. Tunneling leakage can also occur across semiconductor junctions between heavily doped P-type and N-type semiconductors. Other than tunneling via the gate insulator or junctions, carriers can also leak between source and drain terminals of a Metal Oxide Semiconductor (MOS) transistor. The primary source of leakage occurs inside transistors, but electrons can also leak between interconnects. Leakage increases power consumption and if sufficiently large can cause complete circuit failure.

So, with all the information on MOSFETs and delay and leakage values, it is easier to agree that the lesser the delay and leakage values in a given circuit, the better its performance becomes. Deep Neural Nets allow us to remove the dependence on simulation which is in itself a time consuming network. One way to get the 56 parameters resulting in least delay and leakage values would be to randomly change their values and store the best observation. This method has its flaws which need no mention and to do away with this primitive method of finding the minima, we deployed 2 optimisation techniques namely Genetic Algorithms and Particle Swarm Optimisation to converge to the minima in the 56 dimensional space.

LITERATURE REVIEW

Deep Learning

Deep learning (DL) is a powerful machine learning field that has achieved considerable success in many research areas. Especially in the last decade, the-state-of-the-art studies on many research areas such as computer vision, object recognition, speech recognition, natural language processing were led to the awakening of artificial intelligence from deep sleep.

[1] This was the first paper that introduced the revolutionary concept of Deep Learning to the world and Ivakhnenkho, the Father of Deep Learning called it the Group Method of Data Handling (GMDH). The attractive features of GMDH are: (i) It determines the structure of nonlinear systems. (ii) It separates fitting and checking sets to avoid the pitfalls of overfitting. (iii) It can employ multi criteria objective function for model validation. These and other features of GMDH, such as networks of polynomials and layering concepts gave us insights into Non-linear classifications.

[2] The work combines supervised learning with unsupervised learning in deep neural networks. The proposed model is trained to simultaneously minimize the sum of supervised and unsupervised cost functions by backpropagation, avoiding the need for layer-wise pretraining.

[3] The combination of adaptive hybrid leveraging on machine intelligence is used to boost a metaphor-to-metaphorless optimizer called PSO-Rao (Particle Swarm-Rao Optimizer) for solving the energy schedule/delivery problem. Simulation results show that the proposed adaptive hybrid solution can serve as a useful consensus estimator for adequate power systems planning interventions.

[4] PyTorch Geometric achieves high data throughput by leveraging sparse GPU acceleration, by providing dedicated CUDA kernels and by introducing efficient mini-batch handling for input examples of different size. In this work, the researchers present the library in detail and perform a comprehensive comparative study of the implemented methods for homogeneous evaluation scenarios.

Genetic Algorithms

Genetic Algorithm (GA) a heuristic search and optimisation technique is one of the most well-regarded evolutionary algorithms in history. This algorithm mimics Darwinian theory of survival of the fittest in nature. It is analogous to biology for chromosome generation with variables such as selection, crossover and mutation together constituting genetic operations which would be applicable on a random population initially. GA aims to yield solutions for the consecutive generations. The extent of success in individual production is directly proportional to fitness of solution which is represented by it, thereby ensuring that quality in successive generations will be better.

[5] GAs were first proposed by John Holland in this paper as a means to find good solutions to problems that were otherwise computationally intractable. Holland's Schema Theorem, and the related building block hypothesis, provided a theoretical and conceptual basis for the design of efficient GAs. It also proved straightforward to implement GAs due to their highly modular nature.

[6] This paper introduces genetic algorithms (GA) as a complete entity, in which knowledge of this emerging technology can be integrated together to form the framework of a design tool for industrial engineers. An attempt has also been made to explain "why" and "when" GA should be used as an optimization tool.

[7] This paper is intended as an introduction to GAs aimed at immunologists and mathematicians interested in immunology. The authors describe how to construct a GA and the main strands of GA theory before speculatively identifying possible applications of GAs to the study of immunology.

[8] In the field of computing, combinatorics, and related areas, researchers have formulated several techniques for the Minimum Dominating Set of Queens Problem (MDSQP) pertaining to the typical chessboard based puzzles. However, literature shows that limited research has been carried out to solve the MDSQP using bio inspired algorithms. To fill this gap, this paper proposed a simple and effective solution based on genetic algorithms to solve this classical problem. It reports results which demonstrate that near optimal solutions have been determined by the GA for different board sizes ranging from 8 × 8 to 11 × 11.

Particle Swarm Optimisation

Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is originally attributed to Kennedy, Eberhart and Shi [9] and was first intended for simulating social behaviour, as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization.

[10] The work presents the prediction performance results of three algorithms namely: Artificial Neural Network (ANN), Artificial Neural Network trained with Particle Swarm Optimization (PSO) and Adaptive Neuro-Fuzzy Inference System (ANFIS) models. ANFIS and ANN trained by PSO are applied to predict the power and torque values of a Stirling heat engine with a level-controlled displacer driving mechanism.

[11] In this study, the authors implemented a Particle Swarm Optimization (PSO)-based method in parallel by using a parallel MATLAB with one, two, three, and four threads to solve the Job-Shop Scheduling Problem (JSSP). The resulting parallel PSO algorithm is evaluated by applying it to some job shop benchmark problems. The obtained results indicate that implementing PSO in parallel is an effective method for the JSSP that significantly increases the speedups especially for large-scale problems.

REASON FOR HYPOTHESIS

This was an ongoing research project in CVEST, IIIT Hyderabad under Dr. Zia Abbas. This topic was picked to better the work already done and learn more about how circuit delays and leakages are optimised.

What motivated the enquiry

The enquiry was motivated because of non optimum power consumption by electrical circuits like adders because optimal width and length sizes are not known. Growing field of artificial intelligence with the help of neural networks helped us to first build a model and then optimize using algorithms such as genetic algorithms and particle swarm optimization.

Question trying to answer

We tried to answer and find the best and most efficient methods to optimize given circuits. We were able to see that methods such as linear regression were not good and scalable and hence shifted to methods such as neural networks using Keras. We were able to optimize using algorithms such as genetic algorithms and particle swarm optimization algorithms.

METHODS

Our solution was divided into two parts. The first one was to build a good model. We used neural networks to build a model after the initial failure to do so by using linear regression. We were not able to use linear regression because of its huge memory consumption. Neural network did not suffer from this problem and was able to give a good R2 score of 0.99 whereas linear regression gave a score of only about 0.75. After building a model we had to optimise the model. We optimised our model using two algorithms. The first one was genetic algorithms and the second one particle swarm optimisation. Both algorithms were able to converge on delay but were not able to do very well on the leakages. All this was done using popular libraries and frameworks like Keras, Scikit, Pandas, etc.

MATERIALS

The materials used by us were libraries such as keras which have very efficient neural network libraries and functions built in. We used Jupyter notebook and google colab to write out codes. Google colab used tensorflow in its backend. We also used Microsoft teams to have regular meetings with Dr. Zia Abbas and his research students who were working on the same problem. Through Microsoft Teams we were able to communicate flawlessly to gain all information of what was to be done and how it was supposed to be done. We used outlook webmail to have an active line of communication open for mail. We regularly interacted via mail and received the datasets via mail too. For data reading and cleaning we used Pandas. After that we built our model using library Keras. We used Scikit evaluation functions to see how well our model was performing.

PROCEDURE

The procedure used was the following:

1. We read, understood and cleaned our data with popular data management library Pandas. The data was split 80:20 for training and testing.

2. We built our neural network model using Keras with Tensorflow backend.
   To build the Neural Net, Deep Learning was used. The Neural consisted of 9 layers. The input layer had 56 nodes. The hidden layers had 43, 35, 29, 25, 20, 26, and 12 nodes while the output layer had 6 nodes for the delay model and 1 node for the leakage model. All the layers but the last one used ReLu activation function and the last layer used a linear regressor alone. Adam optimization technique was used to converge faster.

3. The two models were trained independently on data consisting of 40000 rows for delay and leakage values separately.

4. Both the models were tested using both R2 scores and MSE to get an idea of their performance on an unseen test set of 10000 rows.

5. We optimised our model parameters given initial sizing using genetic algorithms and particle swarm optimization. To do this the cost function was defined such that it took into account both the delays and leakages. Hence both Neural nets were called simultaneously to provide the values for a test set of 56 parameters.

6. Then the results were compared to initial sizing to find out the efficiency of the framework.

RESULTS

Genetic algorithm

optimised input variables:

[2.2000000000000002e-08, 1.1929617969488659e-07,
2.214632957268874e-08, 9.473937663999493e-08,
2.2000000000000002e-08, 1.8301599500700413e-07,
2.2000000000000002e-08, 6.807187083505735e-08,
2.2000000000000002e-08, 6.688119754199616e-08,
2.5051183388151332e-08, 3.670606054735441e-08,
3.037867102554123e-08, 8.52690548170464e-08,
3.04698630890197e-08, 9.520844006522783e-08,
2.2000000000000002e-08, 2.5686190231096227e-08,
4.242456207777053e-08, 4.5485571811849865e-08,
2.4078120960243408e-08, 9.764564015153045e-08,
4.269953465454261e-08, 9.200469396171522e-08,
2.218437295387325e-08, 6.087796721702229e-08,
3.2146693698029166e-08, 8.807413204269922e-08,
2.4983250094327382e-08, 6.149186285471231e-08,
2.2000000000000002e-08, 5.1703996686861104e-08,
2.2000000000000002e-08, 5.193489212147286e-08,
2.2000000000000002e-08, 5.734993955043629e-08,
2.2404418392033884e-08, 3.248036368215727e-07,
2.7465785634176708e-08, 1.9269274751661492e-07,
4.0059850408802904e-08, 9.486745990440373e-08,
3.833286455248875e-08, 6.144028830180244e-08,
2.2000000000000002e-08, 5.221184625196659e-08,
2.5316551198534223e-08, 6.291774778727786e-08,
2.3067630844742775e-08, 8.958099572593591e-08,
2.4419699093951144e-08, 3.811231130051933e-08,
2.2000000000000002e-08, 9.232315027940973e-08,
4.047352451820549e-08, 4.7482120794649604e-08]

```
optimised delays =
[4.8374468e-12 8.0630624e-12 8.1829803e-12 1.4586535e-11
9.5415099e-12 1.1679768e-11]

optimised leakage = 1.5381e-06
```

Particle swarm optimization

```
optimised input variables:

[2.28494026e-08 1.92036635e-07 2.30192760e-08 1.03397490e-07
 2.21858353e-08 1.32540606e-07 2.44125279e-08 1.46028801e-07
 2.21589445e-08 1.34588414e-07 2.65556615e-08 3.33928746e-08
 2.49412220e-08 3.97777558e-08 2.64460065e-08 4.22147822e-08
 2.46332057e-08 4.95334191e-08 2.82114508e-08 3.74180079e-08
 2.54882745e-08 2.75066085e-08 2.69814787e-08 2.99293270e-08
 2.35011707e-08 6.44580710e-08 2.90479480e-08 7.89820256e-08
 2.40245742e-08 3.11372618e-07 2.27678599e-08 2.33155573e-07
 2.38150790e-08 1.19245937e-07 2.33142312e-08 8.17103475e-08
 2.43343689e-08 2.92337312e-07 3.10056271e-08 1.00289008e-07
 2.85193740e-08 2.02267111e-07 2.94393613e-08 1.51544589e-07
 2.69037428e-08 3.68934320e-08 2.82816654e-08 2.10395719e-07
 3.04585782e-08 9.99505879e-08 2.49598738e-08 1.99162562e-07
 2.26850909e-08 9.62158792e-08 2.90021702e-08 4.37342467e-08]


optimised delays =

[5.592054597158835e-12, 1.1030185445570773e-11,
5.4595087131681375e-12, 1.3326664424773149e-11,
5.779232994940209e-12, 9.799409547706084e-12]


optimised leakage = 1.3509002e-06
```

DISCUSSION

This mini project helped us to know about various techniques used in circuit optimization. We enjoyed our collaboration with Dr. Zia Abbas and his research students. We were able to apply many of the topics we had previously learned in courses such as Machine, Data and Learning to a real world problem and solve it to some extent. Now we will be able to solve any problems of similar structure given to us using the framework and ideas of this mini project.

CONCLUSION

We were able to optimize our given problem to some extent. We learnt new tricks and techniques which can be used further on in future work. We achieved a model with R2 score 0.99 and were able to converge our model for delays but not so much for leakages. The trained Neural Net model had an average MSE of 2%. Both the algorithms Genetic and Particle Swarm Optimisation Technique were able to converge. The delay values reduced considerably by 34% as compared to the initial sizing and the leakage went up by only 8%. The inverse relation between the two parameters resulted in such a behaviour nevertheless this skewness of 34 versus 8 gives weight to the credibility of our proposed framework and further improvements could lead to even better results.

Recommendations For Further Research

1. The accuracy of the Neural Net could be further improved by building a deeper net or deploying different Machine Learning Architectures.

2. Using more training data could also improve the accuracy. A divide of 90:10 could also be used in case of absence of new data.

3. Different optimization algorithms apart from GA and PSO could be used for the convergence of the framework like Glow worm Optimisation.

4. Efficient methods for hyperparameter tuning should be found. The hyperparameters used for PSO and Genetic Algorithms can be tuned further like the social parameters a, b and w in PSO and the mutation probability etc in GA.

5. The cost function could be changed to achieve a different descent if the model is suspected to be stuck in a suboptimal local minima.

6. This idea of finding minima through model building followed by optimisation algorithms can be extended beyond circuits in domains such as finance, physics, etc.

CITATIONS AND BIBLIOGRAPHY

1. Group method of data handling (GMDH): Alexey Ivakhnenko and Lapa

2. Semi-Supervised Learning with Ladder Network: Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, Tapani Raiko

3. Adaptive Hybrid PSO-RAO -Type for Predictive Economic Load Dispatch: Biobele Alexander Wokoma

4. Fast Graph Representation Learning With PyTorch Geometric: Matthias Fey & Jan E. Lenssen

5. Adaptation in Natural and Artificial Systems: J.H. Holland

6. Genetic algorithms: concepts and applications [in engineering design]: K.F. Man ; K.S. Tang ; S. Kwong

7. Genetic algorithms for modelling and optimisation: JohnMcCall

8. Genetic Algorithm Approach for Solving Minimum Dominating Set of Queen Problem: Saad Alharbi and Ibrahim Venkat

9. Particle Swarm Optimization: Kennedy, J.; Eberhart, R.

10. Evaluation of the Stirling Heat Engine Performance Prediction Using ANN-PSO and ANFIS Models: Modestus Okwu

11. Implementation of the PSO algorithm for solving the JSSP in parallel: Wael Abdel-Rehim