STM JOURNALS

Journal of
VLSI Design Tools & Technology

ISSN: 2249-474X (Online)
ISSN: 2321-6492 (Print)
Volume 14, Issue 2, 2024
May–August
DOI (Journal): 10.37591/JoVDTT

https://journals.stmjournals.com/jovdtt

Review JoVDTT

Optimizing Routing and Placement of VLSI Circuits with Differential Algorithms and Neural Networks

Sree Krishna Raja K.[1,*], P. Jaya Krishna[2], D. Rajani[1]

Abstract

The performance of modern VLSI systems is heavily influenced by power constraints, necessitating precise power estimation and effective optimization techniques. Traditional methods, such as gate-level simulations, are often slow and computationally intensive. This study introduces DRPENN (Differential Algorithm for Routing and Placement Optimization using Neural Networks), an innovative solution that combines a Switching Activity Estimator (SAE) with a neural network-assisted differential algorithm. By leveraging toggle rates from simulations to train a Graph Neural Network (GNN), DRPENN circumvents the need for extensive gate-level simulations. This trained model accurately predicts switching activity, optimizing the placement and routing of circuit elements while minimizing computational demands. The methodology involves converting gate-level netlists into graphical representations and using a differential algorithm for back-propagation. Our experimental results show that DRPENN achieves lower error rates and faster inference times compared to conventional probabilistic SAE methods, offering a promising approach for efficient and accurate VLSI design optimization.

Keywords: VLSI design, routing optimization, placement optimization, differential algorithm, neural networks

INTRODUCTION

Modern computing systems face significant power constraints, affecting their overall performance. These power constraints have real world effects from the efficiency and capability of devices as small as a hand-held mobile device to a data center. Accurate power estimation plays a crucial role in all aspects of digital VLSI development. For example, architectural power analysis involves estimating average power consumption over extensive cycles, while power integrity signoff demands precise power analysis on gate-level netlists with physical annotations.

Dynamic power optimizations during logic synthesis, clock gate insertion, or place-and-route are essential for meeting power targets. To achieve accurate power analysis, the preferred way is to annotate toggle rates with vectors obtained from real workloads. This preference is due to its precision. However, this approach requires simulations to be done at gate-level, which is very slow, will take hours to days depending on the design's activity factor and size. However, to solve the above mentioned issues, usually a Switching Activity Estimator (SAE) is used [1]. The SAE gathers the average toggle rates for unit inputs and registers from RTL simulations, and then estimates internal toggle rates within the combinational logic which is synthesized using probabilistic methods. Even though SAE is fast, it comes at the cost of

***Author for Correspondence**
Sree Krishna Raja K.
E-mail: sreekrish@sreekrishnaraja.me

[1]Student, Department of Electronic & Communication Engineering, Nalla Malla Reddy Engineering College, Hyderabad, India
[2]Assistant Professor, Department of Electronic & Communication Engineering, Nalla Malla Reddy Engineering College, Hyderabad, India

Received Date: July 19, 2024
Accepted Date: July 31, 2024
Published Date: August 06, 2024

Citation: Sree Krishna Raja K., P. Jaya Krishna, D. Rajani. Optimizing Routing and Placement of VLSI Circuits with Differential Algorithms and Neural Networks. Journal of VLSI Design Tools & Technology. 2024; 14(2): 14–20p.

accuracy because of issues like signal correlation etc. Another popular technique used to solve the above-mentioned issues involves an optimization technique inspired by flock of birds known as "Particle Swarm Optimization" (PSO). PSO mimics the collective intelligence of a group of particles, here each particle represents a possible minor placement optimization. PSO is used to perform Placement Optimization, Floor Planning, Gate sizing, routing, Clock tree synthesis etc. However, using PSO to perform placement optimization and routing accurately is hindered by two issues. The first and the most impactful issue is Premature convergence. This results in a suboptimal solution. The other issue is Computational Overhead and the need for parameter tuning.

For accurate and rapid average power consumption estimation with less computational overhead, we propose a differential algorithm, assisted by a neural network, DRPENN (Differential Algorithm for Routing and Placement Optimization using Neural networks). DRPENN involves a Switching Activity Estimator (SAE) that eliminates the need for gate-level simulation. In the first phrase, DRPENN takes gate level netlists, its corresponding input ports and they register the toggle rates captured from simulations as input features [2]. The toggle rates per logic gate obtained from simulations are used to train the model. This trained model is assisted by a Differential algorithm for back-propagation. DRPENN shines especially for placement optimization done for circuit level elements, however with better trained models and a more efficient Differential algorithm, DRPENN can be improved to handle all kinds of elements and larger circuits.

BACKGROUND

Switching Activity Estimation and its application to power analysis for optimization of the placement and routing of elements in a circuit presents a challenge as the exact solution requires gate-level simulations which are very time consuming; these gate-level simulations take time to propagate traces from the register outputs and the unit inputs through a combinational logic netlist. In most of the scenarios, only average power is needed for a particular part of the circuit, this makes the implementation of event-driven simulators used for obtaining the final toggle rates α on gate outputs during dynamic power calculation inefficient.

Previously, researchers have tried various approaches to speed up the computation of average gate toggle rates. They have focused on five main strategies:

1. Particle Swarm Optimization based Techniques.
2. Machine Learning Based Techniques.
3. Accelerated gate-level simulation.
4. Statistical Sampling Methods.
5. Probabilistic Switching Activity Analysis.

However, recently, novel methods of simulation using parallel architectures like GPUs have been proposed to accelerate and achieve faster gate-level simulation. Some methods for gate level simulations like Particle swarm Optimization are done using three distinct methods. The first one being power optimization. PSO is used to optimize the timing characteristics of a digital circuit by adjusting the delays of the various gates in the circuit to meet the timing requirements. It also explores various combinations of delays to minimize the overall delay and to maximize performance. PSO is also used by researchers to optimize power consumption of a circuit by simulating adjustment of transistor sizes in different gates [3]. Thus, exploring different transistor sizes to minimize power consumption while maintaining the functionality. Another method is to optimize the placement and routing of gates and routing paths, thus minimizing the wirelength and delay. However, the above three methods are hampered by the lack of global search in particle swarm optimization algorithms. It heavily relies on localized information which results in efficient placement or power optimization in a localized area, but it might not translate to a larger circuit. Another hurdle is the need for parameter tuning. PSO involves various parameters such as cognitive coefficient, social coefficient and inertia weight to achieve the best performance. However, this process involves the audacious task of finding the right combination

of these parameters [4]. This is a challenging process as any improper settings can lead to suboptimal results. Some methods using parallel architectures use hybrid event-driven techniques, while other methods use parallelization of simulation across cycles and gates to achieve significant improvement in time taken for simulation. This is limited by GPU device limitations and the effects of Amdahl's law, to overcome this, often-manual speedup becomes necessary. For a large-scale circuit, it might become necessary to simulate millions of cycles on a gate-level netlist. An alternative to this is to take a statistical sampling approach which isolates a subset of cycles captured from FPGA emulation or RTL simulation and replays only that subset during gate level simulation.

The above approach is often considered desirable as it can reduce very long simulation to the tune of tens of millions of cycles to a small number of cycles which can be easily analyzed. The drawbacks of this method is that it does not solve the issue with slow gate level simulations and it might not cover many corner cases in power analysis. To overcome the slow gate-level simulations, another method is employed where instead of simulation, statistical probabilities derived from the propagating of average toggle rates from input to output designated as Boolean values for each gate. This approach is commonly used in commercial tools owing to its speed of execution. However, this method is wrought with inefficiencies as it does not take account of reconvergence correlations. It assumes gate level switching activity is independent and it does not consider any correlations caused by reconvergence, as switching activity is not always independent. These assumptions lead to inaccurate estimates in average toggle rates, especially in complex circuits where reconvergence is common. Neglecting these correlations often results in underestimating or overestimating the actual toggle rates, leading to inaccurate power estimations and not so optimal placement. To address the above issues, researchers have proposed various methods to incorporate the reconvergence correlations into the considerations of the probability computations. These methods aim to capture the interdependencies between the signals and gates to improve the average toggle rate estimation accuracy. By considering the correlations which arise from reconvergence the probabilities are adjusted to reflect the true behavior of the circuit, resulting in a more reliable power analysis and placement optimization. Accounting of these reconvergence correlations also help with complex intercommuting and feedback loops. Some recent techniques using Machine learning techniques have emerged as promising approaches for average power estimation, routing and placement optimization [5, 6]. A technique called GRANNITE conducted in NV100 Tesla V100 GPU by researchers in Nvdia, employs a baseline probabilistic SAE with less than 5.5% average errors. Another technique PRIMAL uses a Deep Learning model to deduce the combinational switching activity using RTL simulation traces with register outputs and unit inputs. PRIMAL has demonstrated its capability in scalability to accommodate complicated designs with approximately 100 thousand logic gates with exceptional accuracy.

METHODOLOGY

DRPENN is a Neural network powered, fast and transferable approach to SAE assisted by a differential algorithm. DRPENN is capable of running analysis on a completely new set of designs, the types of which are not present in training models.

Neural Network Overview

Graphical Neural networks GNN uses graph structures to capture the interconnections between circuit components enabling efficient analysis, placement optimization, routing for improved performance and power consumption.

The diagram in Figure 1 is the scaled down Graph Neural Network layer mechanism in DRPENN. Graphical Neural Networks aggregates information from the neighboring nodes in the graph to infer switching activity of each gate. Each gate in the GNN is defined as a node. The network updates the features of nodes iteratively by considering the information from its neighboring nodes. This captures signal correlations and dependencies effectively. This enables GNN to learn complex patterns and correlations present in digital circuits, leading to accurate estimates of average switching activity which can be used to infer best routing paths and placement optimization.

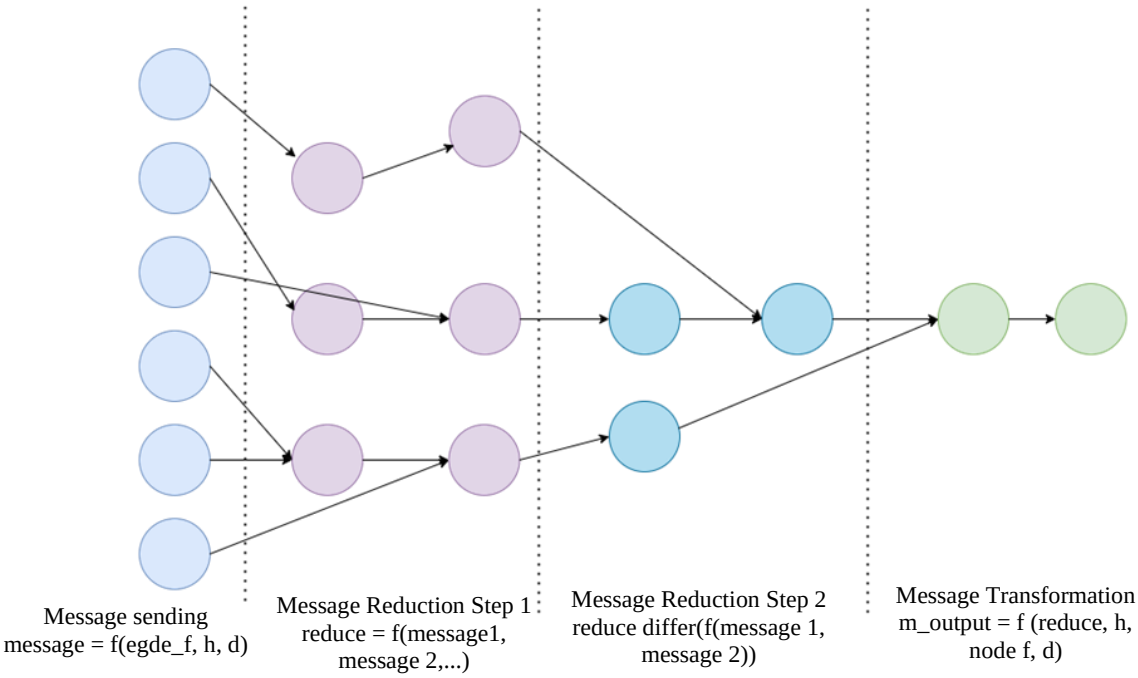

Figure 1. Scaled down graph neural network layer mechanism in DRPENN.

Creating input for Training

DRPENN is a graphical neural network assisted by a differential algorithm. The input for training the GNN is created by generated toggle rates of input ports and register outputs, which are generated by gate-level simulation during training or interference. The toggle rates are considered as the base for any prediction on switching activity within the combinational logic circuit [7]. The features are indicated in a five-dimensional array format with the probabilities of staying low, staying high, switching from low to high, switching from high to low and the differential algorithms inferred output for the same.

Training the input requires converting the gate-level netlist into a graphical representation of the netlist with each element denoted by the node and interconnections denoted by edge (Table 1). This is achieved by a custom script. The graph ensures the connectivity information, and the node information are preserved and converted to simpler formats for the GNN to be trained. This is a one-time process and will not be required once training is completed.

Table 1. Input for training.

Implementation	**DRPENN**
Message Sending	$(edge\ state)\ x \begin{bmatrix} e_0 & e_1 & e_2 & e_4 & e_5 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \cdots & \dots & \dots & \dots \\ \dots & \dots & \dots & e_{24} & e_{25} \end{bmatrix}$
Message Reduction	$\sum_i message$
Differ	$message_{reduced} = \frac{\sum_i message}{h}, i = 1, 2 \dots n$
Node Transform	$(node\ function)\ x \begin{bmatrix} reduce_0 \\ reduce_1 \\ \dots \\ \dots \\ reduce_{25} \end{bmatrix}$

ARCHITECTURE

The Fundamental Components of DRPENN Includes

Each node in the graph denotes the characteristics of an element in the combinational logic circuit. Edges in the graph processed by DRPENN denote the characteristics of the interconnect between two elements in the logic circuit. These two are similar to the traditional components of Graphical Neural Networks.

message =f (edge_f, h, d)

Message parsing in DRPENN is somewhat similar to Message Passing in traditional graphical neural networks, however they differ in a way that they also involve the differential algorithm, which calculates the variation or changes in the message when it is passed between nodes through the edges. The main functionality of the differential algorithm is to perform Back Propagation. This computes the gradients during the training phrase. It involves propagating the errors backward through the network and computing gradients layer by layer. The network parameters are adjusted by the differential algorithm to minimize the loss function.

reduce*(step1)* = f (message1, message2, …)
reduce*(step2)* = differ (f(message1, message2, …))

The differential algorithm also performs sensitivity analysis, which keeps track of how the changes in input data and the parameters affect the output. After getting the message from the previous node, each node will update the message to a new form based on the characteristics of the particular node. This is defined by the Nodal Simulator [8, 9]. The Nodal Simulator combines the node's current features with the aggregated messages to generate a refined representation. The Nodal simulator and the interconnects are further analyzed by DRPENN to perform graph pooling which summarizes the data from multiple nodes or subgraphs reducing the information into smaller graph size while preserving most of the information [10]. The updated node representation will be fed into an output layer depending on the specific sub graph. This involves activation functions and loss functions.

RESULTS AND DISCUSSION

The training and inference for DRPENN is done in a Nvdia GEFORCE GTX 4GB GPU assisted by Ryzen 716 GB RAM. The baseline SAE is constructed by a Graphical Neural network assisted by a Differential Algorithm. For training and testing datasets we used three benchmark circuits with stimulus random for all circuits as listed in Table 2. The results are compared with GRANNITE results.

Expected inference time of SAE is shown in Figure 2. To assess the transferability of the proposed GNN architecture, we trained the GNN model for the two circuits and tested it on new circuits not used during training, thus verifying unseen workloads performance. The Error percentage is calculated by comparing the DRPENN to known expected output. The inference time is greatly reduced to 50%.

The diagram in Figure 3 shows the Inference time of DRPENN. In comparison with baseline probabilistic SAE, DRPENN demonstrates significantly overall accuracy with lower error percentages. In all benchmarks, DRPENN’s error decreases from testing to testing to validation. This proves as the training data is scaled, the error margins will further reduce and will enable low error margins for more complex circuits. The overall performance of DRPENN is highly promising to accurately perform routing and placement optimization across various VLSI designs.

Table 2. Results are compared with GRANNITE result.

Dataset	Description	Gate count	Baseline SAE (Error %)	DRPENN (Error %)
qadd_pipe	32-bit fixed point adder	774	8.1	2.3
qmult_pipe	32-bit fixed point multiplier	1410	7.3	1.2

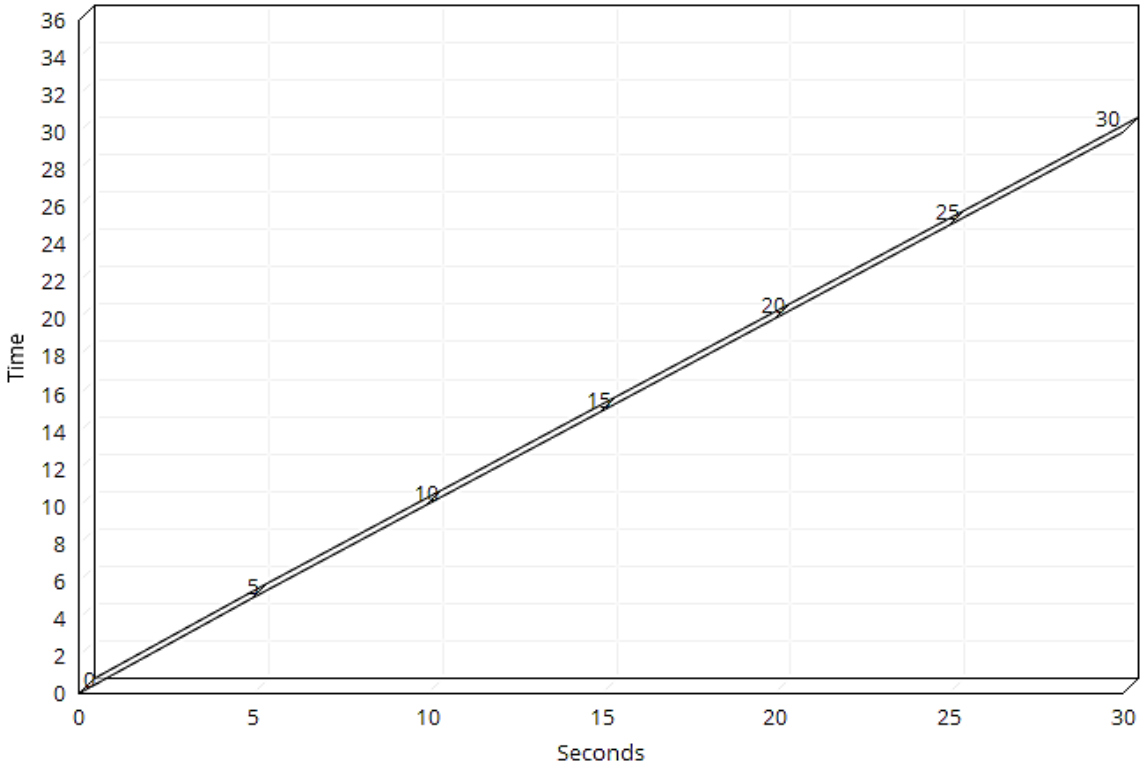

Figure 2. Expected inference time of SAE.

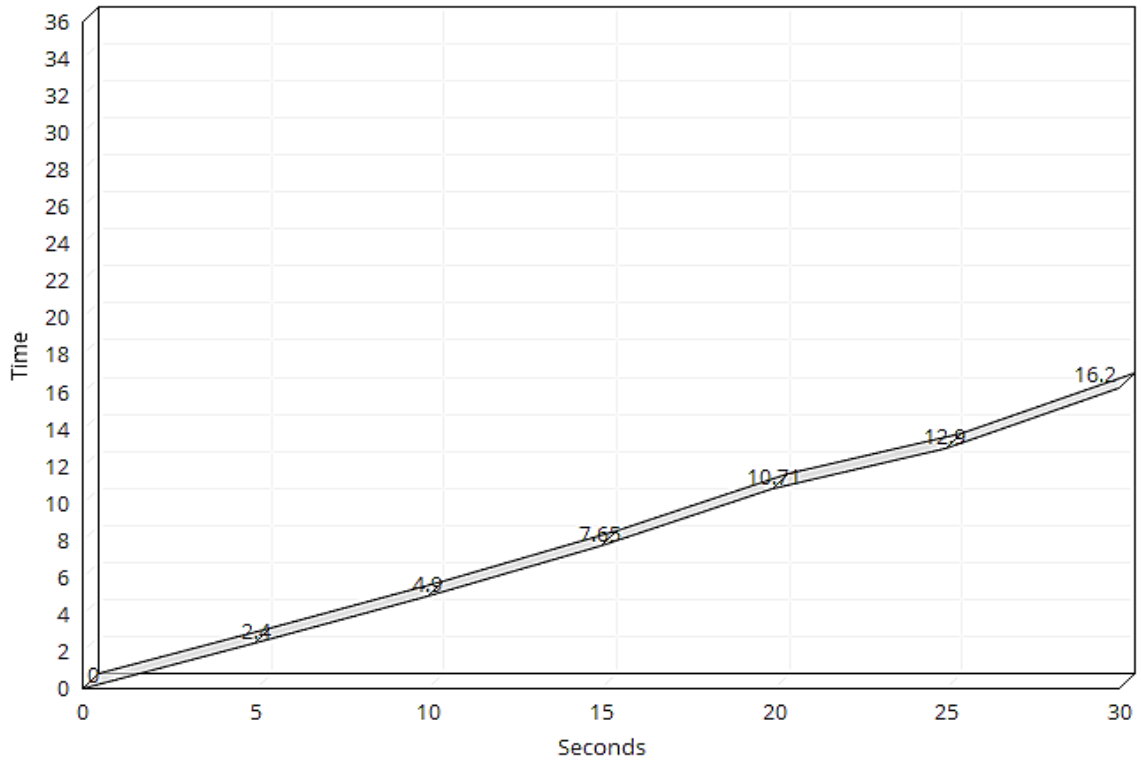

Figure 3. Inference time of DRPENN.

CONCLUSION

DRPENN presents a significant advancement in overall accuracy when compared to baseline probabilistic SAE. When testing the 32-bit fixed point multiplier, the error rate is 1.2% whereas the

baseline probabilistic SAE had an error rate of 7.3%. This is a significant improvement. The results are also similar for 32-bit fixed point adder. During validation, the accuracy improved as the training data was expanded. This proves that DRPENN is adaptable and can be enhanced to a greater extent by improving the training data. During development, tweaking the differential algorithm contributed heavily to the improvement in performance. Further improvement in the differential algorithm will result in even more accurate results. The above observations demonstrate the promise of DRPENN being developed as a highly efficient and accurate general tool for placement and routing optimization in VLSI Design.

REFERENCES

1. Aisha Fayomi, Hassan Amal S, Hanan Baaqeel, Almetwally Ehab M. Bayesian Inference and Data Analysis of the Unit–Power Burr X Distribution. Axioms. 2023; 12(3): 297.
2. Seyed Morteza Nabavinejad, Sherief Reda, Masoumeh Ebrahimi. BatchSizer: Power-Performance Trade-off for DNN Inference. ASPDAC '21: Proceedings of the 26th Asia and South Pacific Design Automation Conference Pages 819 - 824
3. Murata T, Ishibuchi H. Performance evaluation of genetic algorithms for flowshop scheduling problems. Proceedings of 1st IEEE Conference on Evolutionary Computation. 1994; 812–817.
4. Liu Y, Qingfu T, Ma KL. A novel differential evolution algorithm for solving integer programming problems. IEEE Trans Syst Man Cybern Part C (Applications and Reviews). 2010; 40(5): 714–724.
5. Kennedy J, Eberhart R. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. 1995; 1942–1948.
6. Dorigo M, Stützle T. Ant Colony Optimization. MIT Press; 2004.
7. Mitchell M. An Introduction to Genetic Algorithms. MIT Press; 1996.
8. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. Science. 1983; 220(4598): 671–680.
9. Shi Y, Eberhart R. A Modified Particle Swarm Optimizer. Proceedings of IEEE International Conference on Evolutionary Computation. 1998; 69–73.
10. Loshchilov I, Hutter F. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. Proceedings of the International Conference on Learning Representations (ICLR). 2016.
11. Yuan Zhou, Haxong Ren, *et al*. PRIMAL: Power Inference Using Machine Learning. In DAC. 2019;*2019 56th ACM/IEEE Design Automation Conference (DAC)*, Las Vegas, NV, USA, 2019, pp. 1-6.
12. Yanqing Zhang, Haoxing Ren, *et al*. GRANNITE: Graph Neural Network Inference for Transferable Power Estimation. In IEEE *57th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA. 2020; 1–6. 978-1- 7281-1085-1/20

