2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD) | 978-1-6654-9729-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/MLCAD55463.2022.9900103

Driving Early Physical Synthesis Exploration through End-of-Flow Total Power Prediction

Yi-Chen Lu
yclu@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Wei-Ting Chan
wechan@synopsys.com
Synopsys
Hillsboro, Oregon, USA

Vishal Khandelwal
vishalk@synopsys.com
Synopsys
Hillsboro, Oregon, USA

Sung Kyu Lim
limsk@ece.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

Leading-edge designs on advanced nodes are pushing physical design (PD) flow runtime into several weeks. Stringent time-to-market constraint necessitates efficient power, performance, and area (PPA) exploration by developing accurate models to evaluate netlist quality in early design stages. In this work, we propose PD-LSTM, a framework that leverages graph neural networks (GNNs) and long short-term memory (LSTM) networks to perform end-of-flow power predictions in early PD stages. Experimental results on two commercial CPU designs and five OpenCore netlists demonstrate that PD-LSTM achieves high-fidelity total power prediction results within 4% normalized root-mean-squared error (NRMSE) on unseen netlists and a correlation coefficient score as high as 0.98.

CCS CONCEPTS

• **Hardware → Physical design (EDA)**; **Methodologies for EDA**.

KEYWORDS

power prediction, flow-based modeling, design space exploration

ACM Reference Format:
Yi-Chen Lu, Wei-Ting Chan, Vishal Khandelwal, and Sung Kyu Lim. 2022. Driving Early Physical Synthesis Exploration through End-of-Flow Total Power Prediction. In *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '22), September 12–13, 2022, Snowbird, UT, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3551901.3556476

1 INTRODUCTION

Modern low-power physical design (PD) implementation flows require designers to perform design space exploration (DSE) in search of the tool configurations (i.e., input parameters of each design stage) that lead to desired end-of-flow power targets [11]. However, with the ever-increasing design complexity driven by Moore's Law, leading-edge industrial designs in advanced technology nodes are pushing PD full-flow runtime into several weeks, which prohibits designers from performing effective power, performance, and area (PPA) exploration. Therefore, a methodology that performs accurate end-of-flow PPA predictions in early design stages is urgently needed, which allows designers to perform efficient DSE by terminating the runs that are doomed to fail early [5].

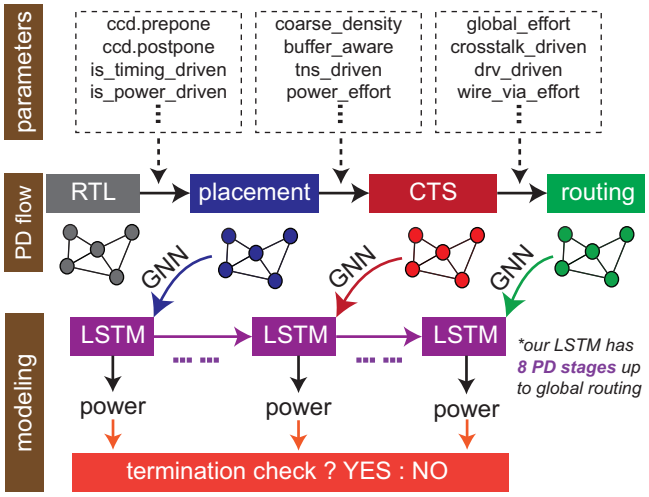

Figure 1: Overview of our sequential modeling approach.

Recently, the authors of [9] have attempted to tackle the PD doomed run prediction problem by predicting end-of-flow design total negative slack (TNS). However, the literature only focuses on building a prediction model to capture the effect of sweeping target frequency and utilization rate (i.e., two parameters) that are set at the beginning of a flow, where all the other tool parameters are fixed. This makes previous work [9] impractical because modern PD implementation tools such as *Cadence Innovus* and *Synopsys IC Compiler II (ICC2)* offer hundreds of tool parameters throughout the PD flow for designers to explore.

To overcome the above issue and truly build a doomed run prediction framework that will benefit PD engineers, in this paper, we specifically focus on the aspect of power and develop an end-to-end learning-based model named PD-LSTM using graph neural networks (GNNs) and long short-term memory (LSTM) networks [3]. Our framework predicts end-of-flow total power consumption in early design stages by sequentially encoding the input parameters specified for each intermediate PD stage. The goal of this work is to develop an early-stage power prediction framework that outputs an end-of-flow total power prediction accurately at each intermediate PD stage by incorporating DSE through sequential modeling.

Figure 1 demonstrates a high-level view of the proposed modeling approach based on a reference commercial PD tool *Synopsys ICC2*. As shown in the figure, unlike previous work [9] assuming

MLCAD '22, September 12–13, 2022, Snowbird, UT, USA

ACM ISBN 978-1-4503-9486-4/22/09.
https://doi.org/10.1145/3551901.3556476

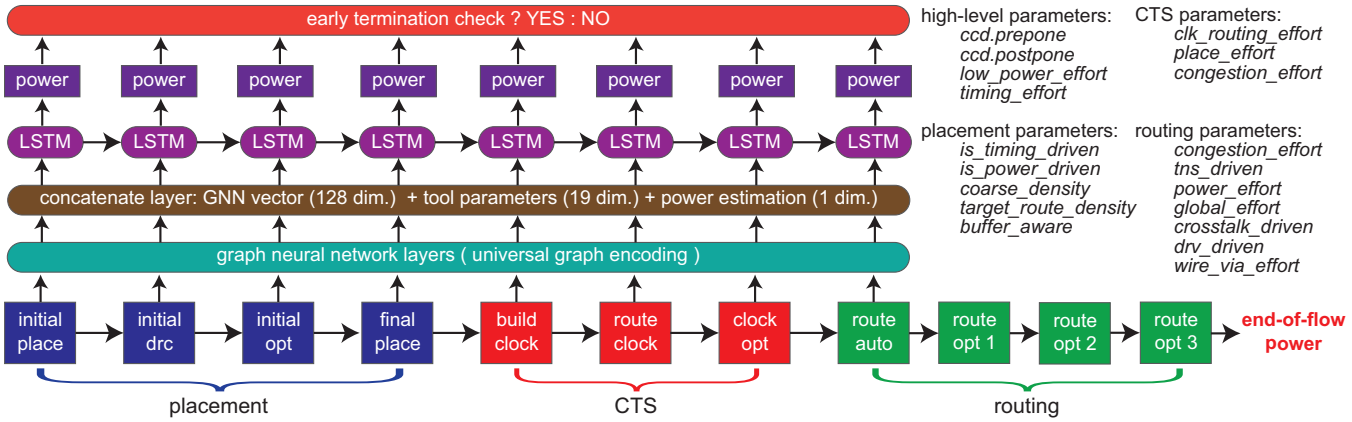

Figure 2: Overview of our PD-LSTM framework and the modeling parameters offered by *Synopsys ICC2*. For each stage, our framework will output an end-of-flow total power estimation which can be taken as the criterion of a doomed PD run.

the underlying implementation is fixed, in this work, we accept the fact that designers may explore various parameters at each intermediate PD stage. Note that due to the space limit, Figure 1 does not show all intermediate PD stages that we select for modeling. In this work, we select 8 design stages offered publicly by *ICC2* to perform sequential modeling using GNNs and LSTM.

At each modeling stage, our framework PD-LSTM will strive to directly predict the end-of-flow total power value by leveraging all the information obtained up to the current stage along with the tool parameters that designers plan to explore in the future stages (i.e., a look-ahead mechanism). With the proposed framework, we envision designers to easily perform the following two operations that are not imaginable before: (1) on-the-fly changing input parameters of future PD stages, which enables a more efficient PPA exploration, and (2) performing early termination on the implementations that are doomed to miss the power targets.

Ideally, we only want to perform the end-of-flow prediction as early as possible in the PD flow. However, there is an accuracy and runtime trade-off between a machine learning (ML) model's prediction and its input features collection. With more features collected from late stages, ML models are expected to make better predictions in terms of fidelity and correlation. In this work, we properly balance this trade-off with sequential modeling techniques by iteratively predicting power at each modeling stage. Note that although at each PD stage, the commercial tool will originally output a power prediction of the underlying design, this estimation from the tool is not accurate. In the experiments, we demonstrate that the proposed framework, PD-LSTM, consistently outperforms commercial tool's early stage power estimations and is generalizable to unseen netlists that are not utilized during the training process.

2 RELATED WORKS OF ML FOR DSE

Novel DSE methods of modern electronic design automation (EDA) flows mainly focus on devising ML-based techniques that address the PPA exploration of a single PD stage [4]. Starting from the placement stage, previous work [1] develops a placement prediction model that takes placement parameters as inputs and output design quality predictions. At the clock tree synthesis (CTS) stage, previous works [6, 8] develop ML models to predict and optimize CTS outcomes under different CTS constraints and targets. As for the routing stage, the authors of [7, 13] develop learning-based techniques to estimate the routability and design rule violations (DRVs). Engineering Change Order (ECO) is a key feature provided by modern PD tools. Previous work [12] presents a gradient-boosted tree model to predict the path-based timing analysis (PBA) results using the context of global-based analysis (GBA). The main idea behind these works is to improve chip design productivity through data-driven modeling approaches. However, all these previous works only focus on the PPA estimation of a single PD stage. To truly improve DSE of a full PD flow, a framework that performs end-of-flow PPA predictions at early design stages is needed.

3 OVERVIEW: PD FLOW MODELING

It has been widely acknowledged that GNNs are powerful ML models that encode graph knowledge into meaningful representations. Since VLSI netlists can be naturally represented as hypergraphs, in this work, we leverage GNNs to distill netlist information at each intermediate PD stage. Given that a netlist under a PD implementation is dynamically changing from stage to stage due to buffer insertion or removal, logic restructuring, fanout re-design etc., the goal of applying GNNs in this work is to encode these netlist updates effectively, where the encoded information is further taken as the input of the LSTM framework to perform power estimation.

Figure 2 presents a high-level overview of our PD-LSTM framework. The key idea behind is to model the PD flow as a sequential process, and perform on-the-fly end-of-flow total power estimation at each targeted modeling stage. Intuitively, with the proposed framework, designers can perform early termination of an implementation without waiting several weeks to obtain the end-of-flow power results. Furthermore, to enable a more fine-grained DSE for better PPA exploration, we train the proposed framework to incorporate the tool parameters specified by designers. That is, parameter configurations are taken as the inputs of the framework. Unlike previous work [9] which assumes the underlying parameters are fixed, in this work, the proposed framework accepts on-the-fly tuning of the input parameters at each intermediate PD stages.

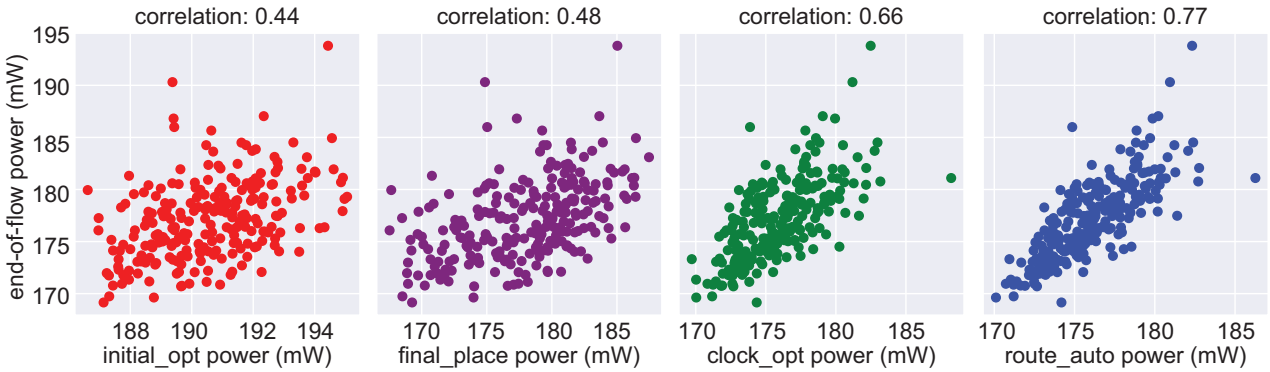

Figure 3: *ICC2* correlation analysis of sequential PD stages on a commercial CPU design. We select 4 intermediate PD stages and plot the power estimation of *ICC2* at each stage to the final achieved power value. We observe that power estimations in early design stages are poorly correlate with the end-of-flow power values.

Numerous parameters are offered by *ICC2* for PPA exploration. In this work, we select 19 parameters by design expertise to perform PD flow modeling which are shown in the right of Figure 2. The high-level parameters are known to have profound impact to the overall PD flow. Specifically, "CCD" stands for "concurrent clock data optimization", which is a key feature of *ICC2* that optimizes clock and data paths for PPA optimization. Finally, we would like to mention that one of our input features to the LSTM framework is the power estimation made by the commercial tool *ICC2*. Although it is known that this power estimation made by the tool is not accurate, we reckon that it may act as a baseline for the model to improve from. In this paper, the main objective of PD-LSTM is to provide better end-of-flow power estimations than the commercial tool *ICC2* in early design stages.

4 DESIGN OF EXPERIMENTS

4.1 Database Construction

The proposed framework adopts supervised learning, which requires a database to be pre-generated for the model to be trained upon. To build the database, we leverage *Synopsys Design Compiler* to synthesize RTL into gate-level netlists, and utilize *Synopsys ICC2* to perform physical implementations. In this work, we utilize 2 commercial multi-core CPU designs and 5 OpenCore designs to perform the experiments. All the designs are synthesized under *TSMC 28nm* technology node at their best achievable frequency. For each synthesized netlist, we generate 200 PD implementations by randomly sampling 19 tool parameters as shown in Figure 2. These parameters govern the tool behaviour of various PD stages such as placement, clock tree synthesis (CTS), and routing, which directly impact the final design quality-of-results (QoR).

4.2 Database Analysis

4.2.1 Correlation Analysis. Since the goal of this work is to perform high-fidelity end-of-flow power estimation in early stages of the PD flow, the power estimation from the commercial tool of each intermediate PD stage becomes a natural and meaningful baseline for us. Figure 3 demonstrates a correlation analysis between the tool estimated power value at selected PD stage and the end-of-flow

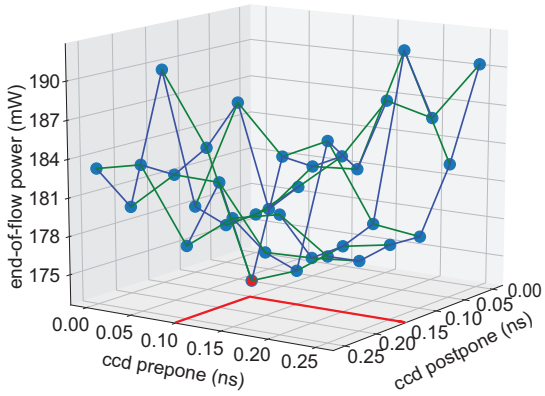

Figure 4: CCD parameter sweeping experiment. We observe that the end-of-flow power values can vary as much as 15% by only sweeping two parameters, and the best achieved power occurs under a non-intuitive combination.

achieved power value. Note that each dot in the figure represents an actual PD implementation. We observe that as moving toward the end of the design flow, the tool provides more accurate power estimations with higher correlation coefficient (Pearson). However, in the early stages of the design flow (e.g., placement), the power estimations of the tool correlate poorly with the final achieved value. This motivates us to build a framework that can provide accurate power estimation in early stages of the design flow.

4.2.2 CCD Parameter Sweeping. As aforementioned, in *ICC2*, CCD optimization is a key methodology to improve design PPA during many optimization phases throughout the entire design flow. In this work, we select two CCD related parameters: "prepone" and "postpone", which are numerical numbers denoting the maximum reduction and increment, respectively, of the clock latency to registers. These two values are extremely critical, since the change of clock latency will have a direct impact on clock skew that governs the setup and hold margins of timing paths. Ultimately, the power consumption will be affected by the tightness or looseness of the timing margins. For example, buffer insertion is usually applied to fix timing violations, which inevitably increase the internal power.

Table 1: Initial node features defined for each design instance.

feature name	description
min slack	min_data_delay - max_clock_delay
max slack	max_data_delay - min_clock_delay
wst output slew	maximum transition of output pin
wst input slew	maximum transition of input pin
drv net power	switching power of driving net
switching power	cell switching power
int power	cell internal power
leakage	cell leakage power

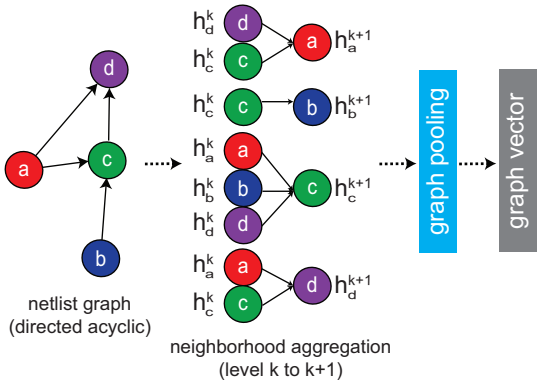

Figure 5: Netlist-to-vector encoding using GNNs. The GNN aggregation is performed from $k = 0$ (initial features) to $k = K$ (learned representations) where K is the number of layers.

Figure 4 demonstrates a CCD parameter sweeping experiment on a commercial multi-core CPU design. In this experiment, we only sweep around the two CCD parameters introduced earlier: prepone and postpone, from 0 (ns) to 0.25 (ns) with an interval of 0.05, while fixing all other input parameters (shown in Figure 2). We observe that the best achieved power occurs when prepone and postpone values are set to 0.1 and 0.15 respectively, which is non-intuitive. Also, it is shown that the achieved power can vary as much as 15% only by simply sweeping these two parameters. Therefore, we believe a framework that predicts end-of-flow power estimation while considering the effect of tool parameters sweeping is highly needed to perform efficient PPA exploration.

5 PD-LSTM ALGORITHMS

In this work, we develop a flow-based ML-powered framework named PD-LSTM that performs on-thy-fly end-of-flow total power prediction at each intermediate PD stage by incorporating the dynamic netlist evolution and various parameter specifications. The goal of our framework is to perform early and accurate power predictions by leveraging graph learning and sequential flow modeling. Specifically, there are two main components in our framework, which are the GNN model and the LSTM network that are responsible for netlist encoding and time-series modeling, respectively.

5.1 Initial Node Features for Graph Learning

To fully benefit from the graph representation learning conducted by GNNs, prior to the learning process, we have to hand-craft related features for each design instance. Table 1 summarizes the features we utilize for GNN modeling. As shown in the table, besides the power features that are directly related to our power prediction task, for each design instance, we also carefully monitor its timing information by introducing timing-related features. The key reason is that timing and power are highly-related with each other. As aforementioned, buffers or inverters often need to be inserted to fix timing violations, and on the other hand, if a design has enough timing budget, power can often be improved by relaxing timing margins [12]. During graph learning, these initial features of a cell will be transformed into meaningful high-dimensional representations by recursively aggregating neighborhood information.

5.2 Graph Embedding

The goal of graph representation learning is to obtain a graph embedding vector that accurately characterizes the underlying netlist. GNNs perform graph representation learning through a messaging passing scheme, where the initial representation vector of a node (i.e., a design instance) can be viewed as a message being recursively transformed and passed onto its neighboring nodes. This message passing process will capture the structural information of the graph and the complex interactions among nodes.

The GNN module is consisted of a set of neuron layers and each of them is responsible to perform aggregation at a specific level k. Figure 5 summarizes the netlist to graph vector encoding process using GNN, where for each node we recursively aggregate its neighborhood information from previous level k to obtain the representation in the next level $k+1$. Let h_v^k denote the representation vector of node v at level k, and h_v^0 represent the initial features defined in Table 1. Then, following from [2], we design our GNN model to transform the features from level k to level $k+1$ as:

$$\begin{aligned} h_{N_k(v)}^{k-1} &= reduce\_mean\left(\{\mathbf{W}_k^{agg} h_u^{k-1}, \forall u \in N_k(v)\}\right), \\ h_v^k &= \sigma\left(\mathbf{W}_k^{proj} \cdot \text{concat}\left[h_v^{k-1}, h_{N_k(v)}^{k-1}\right]\right), \end{aligned} \tag{1}$$

where σ denotes the sigmoid function, $N_k(v)$ denotes the neighboring nodes of node v, W_k^{agg} and W_k^{proj} denote the aggregation and projection matrices at level k respectively, which together represent the neuron layer at level k. Finally, after the last transformation at level $k = K$, we take global mean pooling of $h_v^{k=K}$ across every node in the graph to obtain the final vector g_t in graph-level at timestep t (i.e., an intermediate PD stage) as:

$$g_t = \text{concat}\left[mean\_pooling\left(\left\{h_v^{k=K}\right\}\right), estPower, params\right], \tag{2}$$

where "$estPower$" denotes the commercial tool's estimation power at the current stage, and "$params$" represents the tool parameters that the underlying PD implementation explores, which includes both past (i.e., up to timestep t) and future (i.e., after timestep t) exploration. The vectors $\{g_t|_{t=0}^{t=7}\}$ across 8 intermediate PD stages are further taken as the inputs of the downstream LSTM framework.

5.3 PD Sequential Modeling using LSTM

Since PD is a sequential process where the output of an intermediate stage is the input of the next stage, the encoded graph vectors $\{g_t|_{t=0}^{t=7}\}$ are highly related with each other. Therefore, in this work,

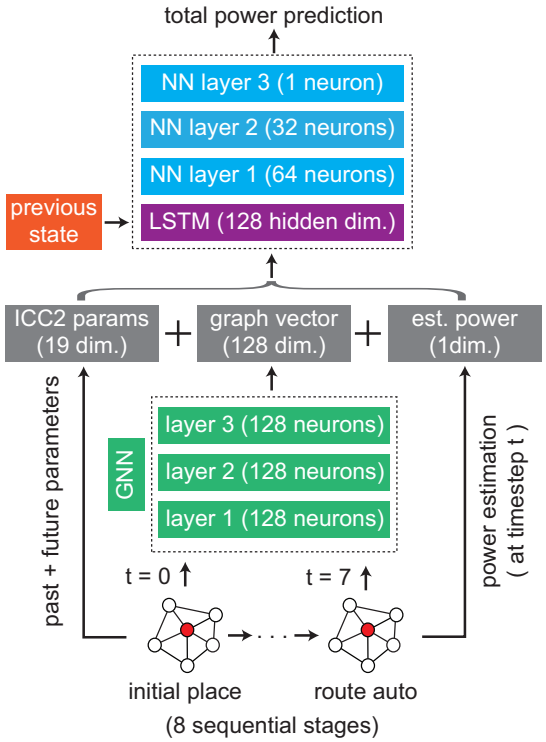

Figure 6: Architecture of the proposed PD-LSTM framework.

we leverage a LSTM network [3] to model such dependency by considering the encoded vectors across various stages as time-domain dependent information. Figure 6 demonstrates the detailed architecture of our framework PD-LSTM.Combined with the GNN model presented above, here, we present an end-to-end framework that leverages LSTM architecture to predict end-of-flow total power value at each timestep t based on the encoded graph vectors.

Basically, the LSTM network is a variant of recurrent neural networks (RNNs) that has a backward connection. That is, at each timestep t, the LSTM network will receive not only the inputs from the current time step, but also the outputs from the previous timestep $t-1$ as shown in Figure 6. Note that since at the beginning there is no previous output, the state vector is usually set to $\mathbf{0}$. The key idea of LSTM is that the network possesses long-term and short-term memories to learn about which information to be disposed of and which to be kept track of. Specifically, a LSTM cell is consisted of three gates, which are input gate i, forget gate f, and output gate o subject to a timestep t. Given an input sequence g_t, the LSTM performs the encoding procedure as follows

$$i_t = \sigma(W_i \cdot [s_{t-1}, x_t] + b_i), \quad (3) \qquad f_t = \sigma(W_f \cdot [s_{t-1}, x_t] + b_f), \quad (4)$$

$$o_t = \sigma(W_o \cdot [s_{t-1}, x_t] + b_o), \quad (5) \qquad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t}, \quad (6)$$

$$s_t = o_t \odot tanh(c_t), \quad (7) \qquad \tilde{c_t} = tanh(W_c[s_{t-1}, x_t] + b_c), \quad (8)$$

where $\{W\}$ and $\{b\}$ denote the weights and biases, σ denotes the sigmoid activation function, s_{t-1} denotes the output from the previous time step, and $\odot$ denotes the element-wise multiplication. As shown in Figure 6, unlike previous work [9] that trains the LSTM framework to predict the target value only at the final time step, in this work, our LSTM model outputs a prediction representing the end-of-flow total power estimation at every time step. Finally, in this work, we take the mean-squared-error (MSE) as the loss function to train the model. Note that the proposed framework PD-LSTM is end-to-end differentiable, which means the parameters in both GNN module and the LSTM network are jointly updated in the same computational graph by optimizing the MSE at each timestep t through a gradient descent optimizer.

Table 2: Our benchmarks and their attributes in *TSMC 28nm*.

Design Name	# Nets	# FFs	# Cells	Usage
CPU-A	206,224	22,366	202,791	training
ECG	85,058	14,018	84,127	
VGA	56,279	17,054	56,194	
JPEG	231,934	37,642	219,064	
CPU-B	542,391	47,552	597,085	testing
AES	90,905	10,688	113,168	
LDPC	42,018	2,048	39,377	

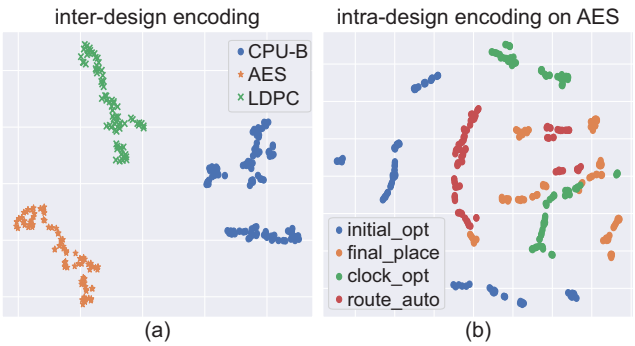

Figure 7: t-SNE visualization of GNN netlist encoding. (a) Each dot represents a complete PD run of an unseen netlist. (b) Each dot denotes a netlist graph at a specific PD stage.

6 EXPERIMENTAL RESULTS

In this section, we demonstrate the achievements of our PD-LSTM framework, which is implemented in *Python3* and the *PyTorch* library. Specifically, we validate our framework on two commercial multi-core CPU designs and five OpenCore benchmarks with a train/test split ratio of 4:3. As aforementioned, for each design, we generate 200 complete PD implementations by randomly sampling the parameters shown in Figure 2. The characteristics of these designs after synthesizing under *TSMC 28nm* are shown in Table 2.

6.1 GNN Netlist Encoding Results

Graph encoding is the key to the success of the proposed PD-LSTM framework. Here, we leverage the t-distributed stochastic neighboring embedding [10] (t-SNE) algorithm to visualize the embedding results in R^2. The visualization results are shown in Figure 7. In Figure 7 (a), for each unseen design, we concatenate the encoded graph vector of each modeling stage and utilize t-SNE to visualize the concatenated vector ($128 * 8$ dimensions) in R^2. As for Figure 7 (b), within the AES benchmark, we visualize the distribution of the encoded graph vector in 128 dimensions extracted from four selected modeling stages. In the figure, we observe that our GNN module

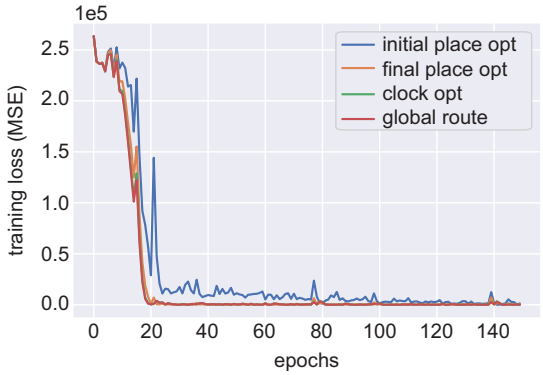

Figure 8: Training loss iteration.

not only clearly differentiates the characteristics of different designs, but also comprehends features from various modeling stages. Hence, we conclude that the proposed GNN model is generalizable.

6.2 Sequential Learning Results

Figure 8 demonstrates the training loss iteration of the selected modeling stages. We observe that the losses of early design stages require more iterations to reach convergence. Table 3 demonstrates the prediction results of the proposed PD-LSTM framework on the unseen netlists that are not utilized in the training process. In this work, our PD-LSTM has 8 modeling stages and for each stage, the framework will output an end-of-flow power estimation as *ICC2*. NRMSE denotes the normalized root-mean-squared error and is calculated by normalizing the RMSE that inherently comes with a "unit" (e.g., *mW*) by the difference between the maximum and minimum ground truth values (i.e., $NRMSE = \frac{RMSE}{power_{max} - power_{min}}$). NRMSE is a popular comparison metric that removes the effect of unit scale. As shown in the table, we observe that the predictions made by PD-LSTM consistently outperform the ones made by *ICC2* starting from early stages of the design flow in terms of correlation coefficient (CC), which directly proves that the proposed framework delivers better end-of-flow total power estimation. Finally, as moving closer to the end of the design flow, we see that the power predictions become more accurate for both *ICC2* and the proposed framework. This is expected because with more features collected from latter stages of the design flow, ML models are expected to make better predictions in terms of fidelity and correlation.

7 CONCLUSION AND FUTURE WORK

In this paper, we have proposed PD-LSTM, a flow-based framework that leverages graph learning and sequential modeling to perform end-of-flow total power estimation starting from early PD stages. The proposed framework consistently demonstrates better power estimation results across various intermediate modeling stages than the reference commercial PD tool *ICC2*. In spite of the superior prediction results achieved, in the future, we aim to explore the possibilities to leverage PD-LSTM to perform on-the-fly PPA optimization by dynamically searching for optimized parameters.

Table 3: PD-LSTM end-of-flow prediction results on "unseen" designs. CC denotes the Pearson correlation coefficient. NRMSE denotes the accuracy of our model. All metrics are computed against end-of-flow total power values.

PD stage (avg time)	unseen design	NRMSE (%)	ICC2 CC	our CC
initial place (3%)	CPU-B	29.8	0.42	0.46
	AES	24.7	0.26	0.5
	LDPC	21.2	0.18	0.37
initial drc (4%)	CPU-B	22.1	0.43	0.58
	AES	28.6	0.25	0.52
	LDPC	27.3	0.18	0.38
initial opt (7%)	CPU-B	18.5	0.42	0.72
	AES	12.1	0.32	0.68
	LDPC	12.9	0.31	0.66
final place (22%)	CPU-B	11.2	0.45	0.81
	AES	9.7	0.35	0.86
	LDPC	9.2	0.32	0.72
build clock (6%)	CPU-B	8.2	0.41	0.89
	AES	7.1	0.47	0.9
	LDPC	8.7	0.43	0.88
route clock (7%)	CPU-B	5.9	0.42	0.94
	AES	6.4	0.76	0.92
	LDPC	5.8	0.74	0.93
clock opt (12%)	CPU-B	5.2	0.65	0.95
	AES	6.4	0.96	0.96
	LDPC	3.9	0.92	0.95
route auto (8%)	CPU-B	4.1	0.75	0.98
	AES	5.3	0.96	0.97
	LDPC	3.7	0.94	0.97

*remaining routing optimization stages take 31% of runtime.

REFERENCES

[1] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath, and K. Samadi. Beol stack-aware routability prediction from placement using data mining techniques. In 2016 IEEE 34th International Conference on Computer Design (ICCD), pages 41–48.

[2] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, 2017.

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[4] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, et al. Machine learning for electronic design automation: A survey. arXiv preprint arXiv:2102.03357, 2021.

[5] A. B. Kahng. New directions for learning-based ic design tools and methodologies. In 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pages 405–410. IEEE, 2018.

[6] A. B. Kahng, B. Lin, and S. Nath. High-dimensional metamodeling for prediction of clock tree synthesis outcomes. In 2013 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP), pages 1–7. IEEE, 2013.

[7] R. Liang, Z. Xie, J. Jung, V. Chauha, Y. Chen, J. Hu, H. Xiang, and G.-J. Nam. Routing-free crosstalk prediction. In 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pages 1–9. IEEE, 2020.

[8] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim. Gan-cts: A generative adversarial framework for clock tree prediction and optimization. In 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 1–8. IEEE.

[9] Y.-C. Lu, S. Nath, V. Khandelwal, and S. K. Lim. Doomed run prediction in physical design by exploiting sequential flow and graph learning. In 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pages 1–9. IEEE.

[10] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(Nov):2579–2605, 2008.

[11] D. MacMillen, R. Camposano, D. Hill, and T. W. Williams. An industrial view of electronic design automation. IEEE transactions on computer-aided design of integrated circuits and systems, 19(12):1428–1448, 2000.

[12] S. Nath and V. Khandelwal. Machine learning-enabled high-frequency low-power digital design implementation at advanced process nodes. In Proceedings of the 2021 International Symposium on Physical Design, pages 83–90, 2021.

[13] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu. Routenet: Routability prediction for mixed-size designs using convolutional neural network. In IEEE/ACM International Conference on Computer-Aided Design, 2018.