IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876
 Journal Volume 11, Iss 09, 2022

EXPLORING THE DESIGN SPACE: HIGH-SPEED INVESTIGATION WITH GRAPH NEURAL PROCEDURES

[1]Natrasimhulu V,[2]Hariprathap Reddy B,[3]Heena Kousar M,[4]Raghavendra S
[1,2,3]Assistant Professor,[4]Associate Professor
Department of ECE

Tadipatri Engineering College,Tadipatri,AP

ABSTRACT: Adders are a crucial component of microprocessors' data channel logic, therefore their design has been at the forefront of VLSI research for quite some time. While EDA flow helps designers get closer to an optimal adder architecture, it isn't always enough. The design space is huge, which is why this is the case. A machine learning-based strategy was offered in earlier studies as a means to investigate the design space. Weak feature representations and an inefficient two-stage learning loop cause prefix adder structures to underperform. A multi-branch framework that combines a variational graph autoencoder and a neural process (NP) is first demonstrated; this is the graph neural process.

Index Terms—*Design space exploration, graph neural process, high speed adder, neural process*
.

1. INTRODUCTION

The design industry has moved away from hand-crafted designs and toward computer-aided designs (CAD) throughout the course of the previous half century, all while increasing the stringency of design standards for VERY large-scale integration (VLSI). This transition away from hand-crafted designs has been accompanied by an increase in the number of design standards. The complexity of the design process, on the other hand, has significantly increased as a consequence of the lightning-fast and shockingly rapid scaling down of the various nodes that comprise semiconductor technology. The ever-increasing size of the design space for microprocessors, along with the time-consuming nature of synthesis cycles, led to the creation of a novel method that is known as efficient design space exploration, or DSE for short. This method is intended to reduce the amount of time spent exploring the design space.

Even if other goals are supplied as limits, Ultimate DSE will only optimize for one goal at a time so that it can achieve that goal. VLSI design, on the other hand, requires the examination of trade-offs covering a wide variety of Quality-of-Result (QoR) variables, including performance, power, and area, amongst others. In this scenario, dynamic programming with multiple objectives is used to produce optimal solutions that strike a balance that satisfies the requirements of a variety of various goals. It is difficult to tell which of these Pareto optimal locations is superior to the others in terms of all of the goals that need to be achieved in the absence of additional knowledge about those goals. It is challenging, but not impossible, to identify a group of Pareto-optimal locations that are not only cost effective but also conveniently located. There are times when the objective functions are unknown, and the only way to find out what they are is to perform an evaluation point-by-point, which requires a large amount of time and effort. In these cases, the only way to find out what the goal functions are is to conduct an evaluation. In the present EDA design cycle, each call has the potential to return one implementation, but even if it does, there is no guarantee that the implementation will be a member of a Pareto set. Moreover, even if it does return one implementation, there is no guarantee that it will be a member of a Pareto set. The procedure of establishing the set that is optimal according to the Pareto principle takes a considerable amount of time because it requires the performance of multiple evaluations. In conclusion, multi-objective dynamic programming seeks to find a set of locations that are Pareto-optimal while reducing the amount of

ISSN PRINT 2319 1775 Online 2320 7876
Research Paper Journal Volume 11, Iss 09, 2022

time and effort that is spent analyzing the multiple goals.

This article focuses mostly on the DSE approaches for adder design as its topic of discussion. When very large-scale integration is carried out, the design of carry-propagation units, which is at the core of the adder design, poses one of the primary challenges that needs to be addressed. Even though the unit can be constructed by leveraging enormous parallel prefix structures, there is still a need for actual synthesis and physical design operations. The fact that such implementations are feasible does not change the fact that this is the case. Recent research has suggested regular adder architectures that can construct a single prefix adder network by applying a specific set of restrictions; however, at this time, these designs are only able to handle a very small subset of the potential topologies. An active learning-based optimization model was used to examine Pareto-optimal adder designs by integrating two pruning approaches into the prior, and Ma et al. produced a state-of-the-art solution for generating the prefix graph structures. This was accomplished by combining the two pruning approaches into the prior. The addition of two different methods of pruning into the earlier iteration makes it possible to achieve both of these goals.

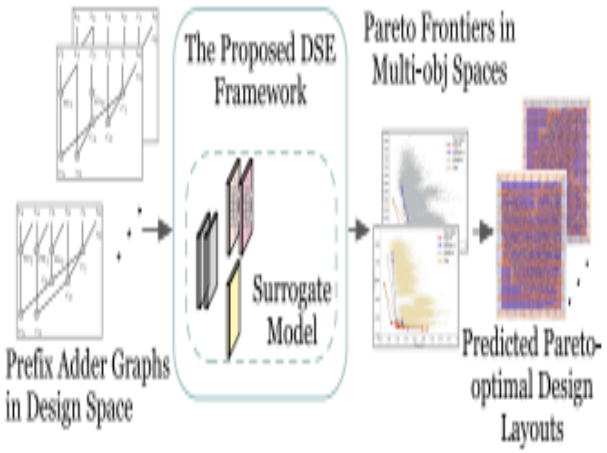


Fig.1. High-speed adder DSE

2. BACKGROUND WORK

Graph Structured Data

G = (a, V, E) defines a graph, where an is a global attribute1, V is the set of nodes where V = vii=1:Nv with vi being the node's attribute, and E is the set of edges where E = ek, vk, ukk=1:Ne where ek is the attribute on the edge, vk is the attribute on the nodes that are connected by the edge, and uk is the set of nodes. The methods shown here are not limited to undirected graphs; in fact, they can be used with many different types of graphs.

Conditional Neural Processes

Conditional Neural Processes (CNPs) are a subset of the more well-known Gaussian Process. A CNP takes in an IRdx at the xi input and outputs another at the yi output. By placing restrictions on any given set of context points XC:= (xi)iC and their associated outcomes YC, we are able to design a family of conditional distributions that may be realized. The conditional distribution can therefore be used to characterize an unlimited number of objectives XT:= (xi)iT and their associated outcomes YT. The model does not care what sequence the context clues and the targets are given in. Because of this invariance, random samples of edges can be used in learning and imputation. We shall use C T as an example, although the model is defined for any combination of these two parameters.

To do this, we use the commutative operation 2, which translates elements in some IRd to a single element in the same space, to add up the data about the context points. This is known as the rC context vector in the academic world. The detected context points have their data condensed into rC. The CNP is now formally learning this conditional distribution.

$$P(Y_T|X_T, X_C, Y_C) \iff P(Y_T|X_T, r_C)$$

In order to put this into action, we must first provide the context points to a DNN h, which will then build an embedding ri of each context point,

ISSN PRINT 2319 1775 Online 2320 7876
 Journal Volume 11, Iss 09, 2022

of a length we specify. At each context point, multiply the representation vector by the constant to get rC. Based on the assumption of rC, we may calculate the distribution zi of the desired target outputs yi by decoding the target points XT.

$$r_i = h_\theta(\vec{x_i}) \qquad \forall \vec{x_i} \in X_C$$
$$r_C = r_1 \oplus r_2 \oplus r_3 \oplus \cdots \oplus r_n$$
$$z_i = g_\phi(\vec{y_i}|r_C) \qquad \forall \vec{y_i} \in X_T$$

Edge Imputation

In many settings, like traffic forecasting or social networks, its existence is acknowledged but its value is uncertain. In conventional edge imputation, a value estimate for the edge is created as a point estimate. Mean filling, regression, and classification are all viable options for this [11]. Traditional approaches, especially mean filling, may obscure vital aspects of the edge values, such as variance. To see how this works in practice, consider the following statement: "Bias in variances and covariances can be greatly reduced by using a conditional distribution and replacing missing values with draws from this distribution." This, together with the neural structure of the conditional estimation, lends credence to the idea that Graph Neural Processes are useful in imputation because they preserve essential features of edge values.

Bayesian Deep Learning

A Bayesian neural network is trained by providing it with a series of inputs (X = x1, • • •, xn) and expecting it to produce a sequence of outputs (Y = f(x)). The correlation between x and y is commonly explained using a fixed neural network since it seems reasonable. There is a lot written about this, and as a result, two distinct schools of Bayesian Deep Learning have developed. There are many more options available than just these two. To begin, a neural network's hidden layer weights W are estimated using a probability distribution. It can be thought of as simulating a random variable with a known prior distribution over the weights. Here we keep track of how much of an unknown the neuronal shift is from the outset. The output of the neural network can be regarded of as a random variable because the weight values W are not fixed. The neural network and loss function are used to learn a generative model. Integrating with regard to the posterior distribution of W yields predictions from such networks.

$$p(y|x, X, Y) = \int p(y|x, W)p(W|X, Y)dW.$$

There are many proposed solutions in the academic literature, despite the fact that this integral is notoriously difficult to compute in practice. The output of a Bayesian neural network encodes a distribution across all possible outcomes for a given set of inputs. This immensely useful aspect of Bayesian deep learning can be captured with the use of GNPs. The output of a Graph Neural Process can be represented as a random variable, whose conditional distribution can be taught. Unlike the first type of Bayesian neural networks, the weights W in this research did not come from a random distribution.

3. ADDER FEATURE EXTRACTION AND REGRESSION

Characteristics employed by conventional machine learning methods are, for the most part, created by hand with the assistance of domain experts. Previous research on machine learning-based adder DSE, which does things like employ hand-engineered features and splits the feature extractor and the resulting learning model, follows this line of reasoning. Separating them increases the likelihood that the whole system will have to settle for less-than-ideal results.

To avoid needing domain expertise or laborious feature extraction, deep learning algorithms aim to employ a general-purpose learning technique to automatically discover high-level features from data. The end-to-end learning system has also achieved recent progress in a number of EDA-

IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876
 Journal Volume 11, Iss 09, 2022

related domains. Using the automatic feature extractor modified for prefix adder networks and the regressor, we construct a full-stack, deep learning-based model called GNP.

This section reveals the anticipated GNP's elaborate tree structure. The suggested multibranch flow is illustrated in Figure 2; it is supported by a spine (the encoder of the graph autoencoder) and operates concurrently on two branches (the decoder of the graph autoencoder and the NP, which stands in for the typical Gaussian process). An input prefix adder's latent representation is obtained by using a graph autoencoder (GAE) on the underlying data and the input stream. The regression values and associated uncertainty for stream II are generated by an NP that employs an encoder-decoder design.

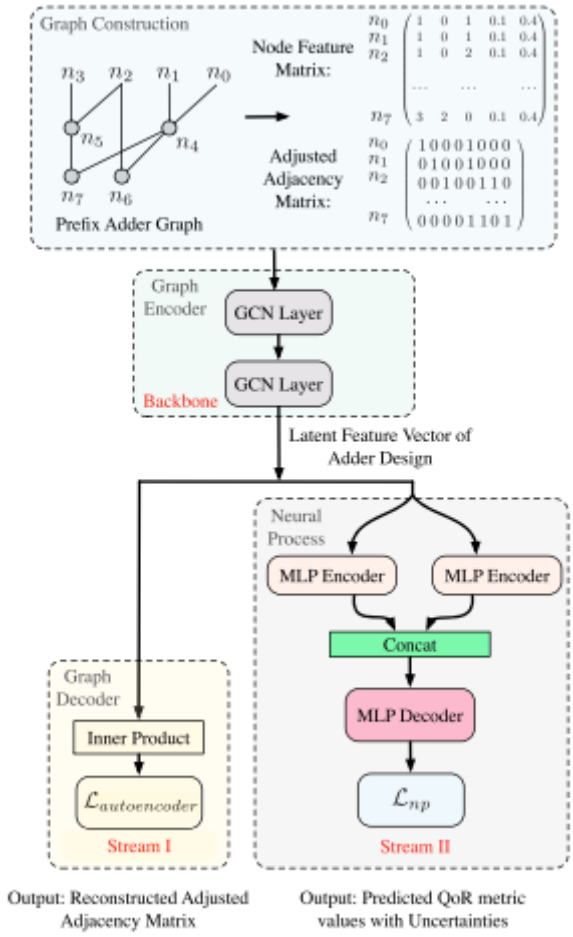


Fig. 2. Diagram of the proposed graph neural process

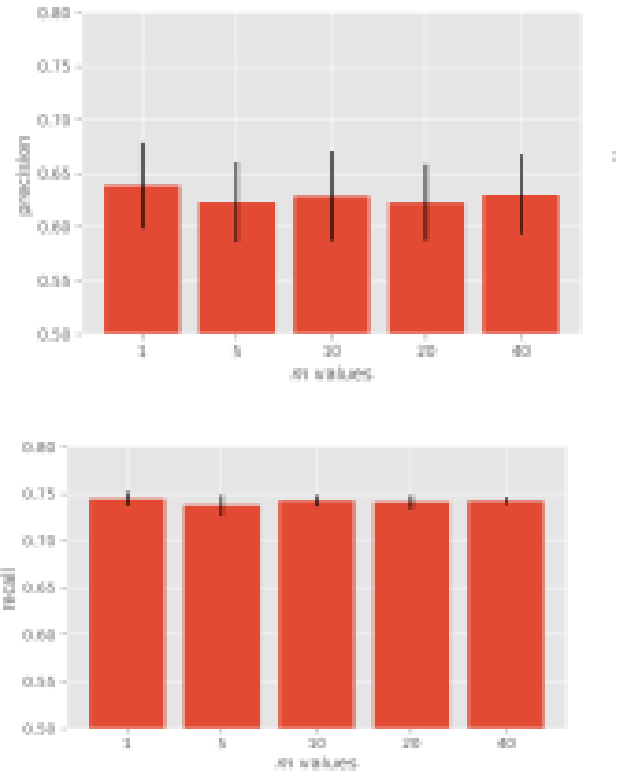
Algorithm 1 Graph Neural Processes

Algorithm 1 Graph Neural Processes. All experiments use $n_{\text{epochs}} = 10$ and default Adam optimizer parameters

Require: p_0, lower bound percentage of edges to sample as context points. p_1, corresponding upper bound. m, size of slice (neighborhood) of local structural eigenfeatures.
Require: θ_0, initial encoder parameters. ϕ_0 initial decoder parameters.

Let X input graphs
for $t = 0, \cdots, n_{\text{epochs}}$ **do**
  for x_i in X **do**
    Sample $p \leftarrow \text{unif}(p_0, p_1)$
    Assign $n_{\text{context points}} \leftarrow p \cdot |\text{Edges}(x_i)|$
    Sparsely Sample $x_i^{cp} \leftarrow x_i|_{n_{\text{context points}}}$
    Compute degree and adj matrix D, A for graph x_i^{cp}
    Compute $L \leftarrow I_{n_{\text{context points}}} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
    Define F^{cp} as empty feature matrix for x_i^{cp}
    Define F as empty feature matrix for full graph x_i
    for edge k in x_i **do**
      Extract eigenfeatures $\Lambda|_k$ from L, see eq (7)
      Concatenate $[\Lambda|_k; v_k; u_k; d(v_k); d(u_k)]$ where v_k, u_k are the attribute values at the node, and $d(v_k), d(u_k)$ the degree at the node.
      if edge $k \in x_i^{cp}$ **then**
        Append features for context point to F^{cp}
      end if
      Append features for all edges to F
    end for
    Encode and aggregate $r_C \leftarrow h_\theta(F^{cp})$
    Decode $\tilde{x}_i \leftarrow g_\phi(F|r_C)$
    Calculate Loss $l \leftarrow \mathcal{L}(\tilde{x}_i, x_i)$
    Step Optimizer
  end for
end for

4. RESULTS



IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876
Research Paper Journal Volume 11, Iss 09, 2022

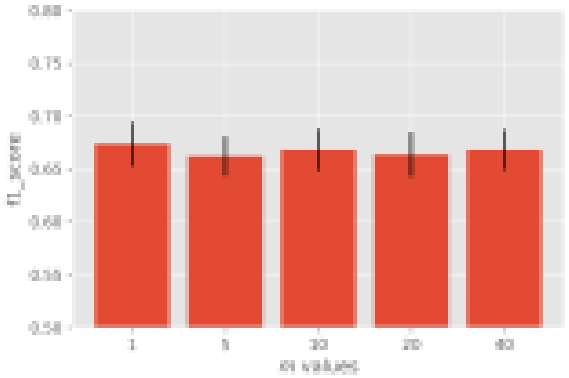


Figure 3: The biggest eigen value encodes enough GNP information in m experiments.

That is to say, for a broad span of m values, the findings are inconclusive. Let's start with the fact that the GNP has the highest F1-score on 14 of the 16 datasets and the highest recall on 14 of the 16 datasets (recall is identical with classification accuracy in this context). By learning an abstract representation of the data and a conditional distribution across edge values, the Graph Neural Process achieves superior performance than both naïve and strong baselines in edge imputation. This may be done with datasets containing anywhere from a few hundred to more than nine thousand graphs using the GNP. We also remark that the GNP can ultimately win out over class distinctions.

Dataset	$\lvert X\rvert$	$\lvert N\rvert$	$\lvert E\rvert$	$\lvert \cup \{e_k\}\rvert$
AIDS	2000	15.69	16.20	3
BZR_MD	306	21.30	225.06	5
COX2_MD	303	26.28	335.12	5
DHFR_MD	393	23.87	283.01	5
ER_MD	446	21.33	234.85	5
Mutagenicity	4337	30.32	30.77	3
MUTAG	188	17.93	19.79	4
PTC_FM	349	14.11	14.48	4
PTC_FR	351	14.56	15.00	4
PTC_MM	336	13.97	14.32	4
Tox21_AHR	8169	18.09	18.50	4
Tox21_ARE	7167	16.28	16.52	4
Tox21_ARLBD	8753	18.06	18.47	4
Tox21_aromatase	7226	17.50	17.79	4
Tox21_ATAD5	9091	17.89	18.30	4
Tox21_ER	7697	17.58	17.94	4

Table 1: Features of the explored data sets

AIDS

The AIDS Antiviral Screen dataset compiles the findings obtained from tests conducted on thousands of different compounds to assess whether or not these compounds exhibit anti-HIV activity. Through the use of the screen's results, the data relevant to these substances is presented in the form of a chemical graph. On a dataset of equivalent size, the GNP displays performance that is 7% better than that of the RF. This advantage can be attributed to the fact that the GNP has more features.

bzr,cox2,dhfr,er.

An investigation into the pharmacophore kernel was carried out with the assistance of the chemical compound databases BZR, COX2, DHFR, and ER. The 3D coordinates of the compounds are included in each and every one of these datasets. On these datasets, different algorithms produce variable outcomes; these are the datasets that, on average, have orders of magnitude more edges than the other datasets. These are the datasets that have been subjected to the various methods. These datasets have a considerable class imbalance; if you guess the edge label that occurs most frequently, you will have an accuracy of approximately 90%. For example, the bzr dataset contains 61,594 entries in class 1, yet there are only 7,273 records in total throughout the next four classes combined. In spite of this, the GNP has the best F1 and recall on two out of the four of these, whereas random forest has the highest precision on three out of the four of these. across addition to the fact that there is an overwhelming amount of data to deal with, it is probable that the difficulty can be attributable to the fact that the classes are not dispersed evenly across the world.

Mutagenicity, MUTAG.

There are 188 unique chemical compounds that make up the MUTAG dataset. Each of these chemicals has been classified into one of two groups depending on the mutagenic effect it has

ISSN PRINT 2319 1775 Online 2320 7876
Research Paper , Journal Volume 11, Iss 09, 2022

on a bacterial population. Despite the fact that the mutagenicity dataset includes a collection of chemicals and information regarding their interactions with in vitro, the dataset has been criticized for its lack of transparency. Although both GNP and RF are much more accurate than naive baselines, the results of this particular test show that GNP performs somewhat better than random forest by a few percentage points.

PTC_*.

The several datasets for the Predictive Toxicology Challenge each contain several hundred organic compounds that have been categorized according to the degree to which they induce cancer in male and female mice and rats. The goal of this challenge is to develop a method that can accurately predict the likelihood that a substance will cause cancer in animals. On the PTC family of graphs, the performance of GNP is superior to that of random forests by a margin of 10–15% in terms of precision and by 3–10% in terms of F1-score; nonetheless, both methods perform better than using naive baselines.

Tox21_*.

The human nuclear receptor signaling and stress pathway was probed with a total of 10,000 unique chemical compounds, and the data presented here is the result of those investigations. The principal goals of the project were to improve human health in general and conduct research on the relationships between structure and activity. On the Tox family of graphs, the GNP displays much greater performance in comparison to every other model by approximately 20% in precision, approximately 12% in F1, and approximately 10% in recall.

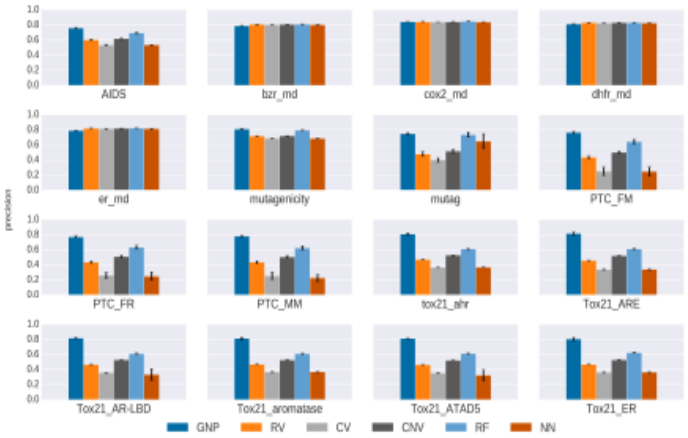


Figure 4: Our method is 0.2 better. GNPs' edge imputation works across metrics and datasets.

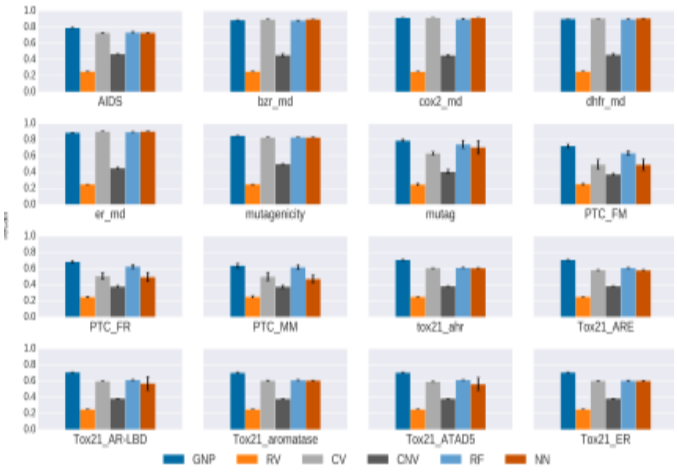


Figure 5: Experimental recall graph compared with baselines

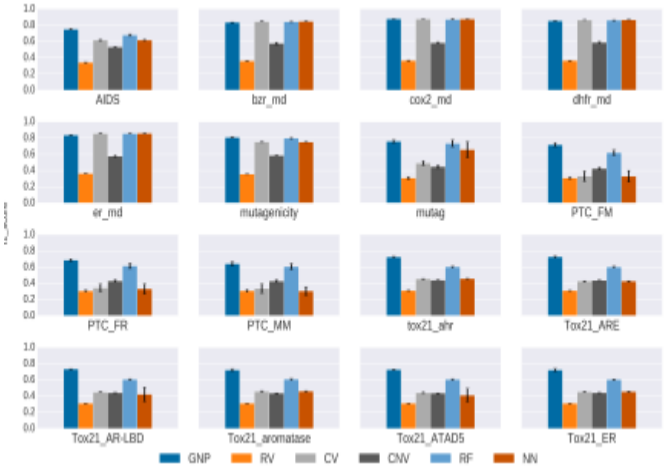


Figure 6: Experimental F1-score graph compared with baselines

5. CONCLUSION

As a novel end-to-end learning model, the graph neural process provides both a GAE and an NP for prefix adder architectures. Automatic feature extraction from prefix adder structures is made possible by GNP. Furthermore, we based our DSE methodology for low-power, high-throughput prefix adders on a sequential optimization model informed by the GNP. Automatic feature extraction and exploration of the adder space at a high grade. The experiments verified the new approach's superiority. We theorized that as VLSI designs evolved, we would be able to modify our theory to deal with a wider variety of DSE issues (including, but not limited to, the multiplier DSE problem and adder DSE worries across bit width and technology nodes).

REFERENCES

[1] Q. Guo, T. Chen, Y. Chen, Z.-H. Zhou, W. Hu, and Z. Xu, "Effective and efficient microprocessor design space exploration using unlabeled design configurations," in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), 2011, pp. 1671–1677.

[2] D. Li, S. Yao, Y.-H. Liu, S. Wang, and X.-H. Sun, "Efficient design space exploration via statistical sampling and adaboost learning," in Proc. ACM/IEEE Design Autom. Conf. (DAC), 2016, pp. 1–6.

[3] S. Roy, Y. Ma, J. Miao, and B. Yu, "A learning bridge from architectural synthesis to physical design for exploring power efficient highperformance adders," in Proc. IEEE Int. Symp. Low Power Electron. Design (ISLPED), 2017, pp. 1–6.

[4] C. Lo and P. Chow, "Multi-fidelity optimization for high-level synthesis directives," in Proc. Int. Conf. Field Programmable Logic Appl. (FPL), 2018, pp. 272–279.

[5] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in Proc. Int. Conf. Machine Learning (ICML), 2018, pp. 3312–3320.

[6] Murali Krishna G., Karthick G., Umapathi N. (2021) Design of Dynamic Comparator for Low-Power and High-Speed Applications. In: Kumar A., Mozar S. (eds) ICCCE 2020. Lecture Notes in Electrical Engineering, vol 698. Springer, Singapore.

[7] M. Gori, G. Monfardini, and F Scarselli. A new model for learning in graph domains. IJCNN, 2:729–734, 2005.

[8] F. Scarselli, S. L. Yong, M. Gori, M. abd Hagenbuchner, A. C. Tsoi, and M. Maggini. Graph neural networks for ranking web pages. IEEE/WIC/ACM International Conference on Web Intelligence, pages 666–672, 2005.

[9] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. ICLR, 2016. [10] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications.

[11] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. https://arxiv.org/abs/1806.01261, 2018.

[12] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 34(4):18–42, 2017.

[13] Emre Aksan and Otmar Hilliges. 2019. STCN: Stochastic Temporal Convolutional Networks. In International Conference on Learning Representations, ICLR.

[14] Gabriel Appleby, Linfeng Liu, and Li-Ping Liu. 2020. Kriging convolutional networks. In The Thirty-Fourth Conference on Artificial Intelligence AAAI, Vol. 34. 3187–3194.

[15] Saikrishna, D., Umapathi, N., & Mothe, S. (2022). Delays in the Generation of Test Patterns and in the Selection of Critical Paths. Specialusis Ugdymas, 2(43), 2986-2997.

[16] Di Chai, Leye Wang, and Qiang Yang. 2018. Bike flow prediction with multi-graph convolutional networks. In International Conference on Advances in Geographic Information Systems, SIGSPATIAL. 397–400.

[17] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. A neural attention model for urban air quality inference: Learning the weights of monitoring stations. In AAAI. 2151–2158.

[18]. Srinivas, L., & Umapathi, N. (2022, May). New realization of low area and high-performance Wallace tree multipliers using booth recoding unit. In AIP Conference Proceedings (Vol. 2393, No. 1, p. 020221). AIP Publishing LLC.

[19]. Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: A pareto driven machine learning approach," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 38, no. 12, p. 2298–2311, Dec. 2019.

[20]. Prasad, R., UmapathI, N., & Karthick, G. (2022). Error-Tolerant Computing Using Booth Squarer Design and Analysis. Specialusis Ugdymas, 2(43), 2970-2985.

[21]. Christopher M Bishop and Nasser M Nasrabadi. 2006. Pattern recognition and machine learning. Springer.