ReNoC-ML: Reliability-Aware Network-on-Chip Performance Modeling using Machine Learning

Zhuofan Lin[1,*], Yang Wei Lim[1], Yongfu Li[2], Fakhrul Zaman Rokhani[1,**]
[1]*Dept. of Comp. and Comm. Sys. Engineering, Faculty of Engineering, Universiti Putra Malaysia, Selangor, Malaysia*
[2]*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China*
Email: *209471@student.upm.edu.my, **fzr@upm.edu.my

***Abstract*—Network-on-Chip (NoC) architectures have emerged as a promising solution to address the interconnection challenges inherent to complex multi-processor integrated circuits. Nevertheless, the increase network size of NoC leads to a time-consuming process for design space exploration (DSE) using traditional cycle-accurate simulator. This study proposed a machine learning (ML) approach to accelerate the prediction of NoC performance while incorporating considerations of reliability-related parameters (ReNoC-ML). A bespoke cycle-accurate NoC simulator was employed to generate a comprehensive dataset. Following preprocessing, three ML algorithms, namely Random Forest (RF), XGBoost, and Multilayer Perceptron (MLP) were assessed for their predictive accuracy and computational efficiency. The XGBoost model exhibited superior performance, attaining an R^2 score of 0.9502, a mean absolute error (MAE) of 616.4072, and a speedup of 10^7 in comparison to the cycle-accurate simulator.**

***Index Terms*—Error Correction Code Scheme, Machine Learning, Network-on-Chip, Performance Modeling, Reliability**

I. Introduction

THE increasing complexity of multi-processor integrated circuit designs is leading to performance saturation, primarily due to the bottleneck of traditional bus-based interconnection methods. To address this challenge, NoC, drawing inspiration from the computer networking theory, is proposed. By embodying the packet-based transmission principles, NoC facilitates modular, scalable, and high-bandwidth communications [1]. However, similar to other communication systems, NoC faces reliability challenges, particularly at the high operating frequency and low operating voltage [2]. Permanent, intermittent, and transient faults result from incorrect interpretation of the destination IP in semiconductor devices. Therefore, ECC schemes are necessary, which help in detecting and correcting errors, thereby enhancing the reliability of data transmission in NoC.

In the system design level of NoC, the procedure of DSE is crucial. NoC has various parameters which affect the performance like network size, topology, injection rate, ECC scheme, BER, etc. The traditional method for performance modeling is to utilize a cycle-accurate simulator (e.g., Booksim [3], Noxim [4], PyOCN [5]) which simulates a lower abstraction to acquire an accurate result.

Another method is the queuing theory-based analytical model, which characterizes the behavior of NoC components using mathematical equations [6], [7]. An analytical model, grounded in M/G/1/K queuing theory, is proposed to provide a 5% difference to the simulation results for a 4×4 mesh NoC [6]. Another recent research presents a further evolution, which presents a generalized analytical model that conceptualizes NoC packet flow as an open, feed-forward queuing network [7]. This analytical model allows more tunable parameters as the inputs. However, the accuracy of this analytical method drops as the network size expands.

With the popularity of ML technology in recent years, an ML-based approach is regarded as an alternative way for the performance modeling to tackle the problem of runtime for large and complex NoC architectures [8]–[10]. These works generated the dataset based on Booksim 2.0 or Noxim for performance metrics (e.g., average delay, average hop, throughput, etc) of NoCs. They adopted ML algorithms like SVR, Linear Regression, or customized ANN, which performed favorably for the prediction. However, these works did not include the impact on the performance of reliability-related parameters (e.g., BER, ECC schemes).

To fill up the current research gap of lacking consideration of reliability-related parameters on NoC performance modeling, a dataset was built from a customized cycle-accurate NoC simulator which includes reliability-related parameters such as BER and ECC schemes [11]. Three different ML algorithms namely Random Forest (RF), XGBoost, and Multilayer Perceptron (MLP) are employed to model the NoC performance prediction. XGBoost shows outperform results than other MLs, which attained R^2 score of 0.9502, a mean absolute error (MAE) of 616.4072, and a speedup of 10^7 in comparison to the cycle-accurate simulator.

The rest of this article is organized as follows, Section II presents the detailed framework in detail. Section III provides the experimental results, and Section IV concludes the work and reveals the future directions of the research.

II. Methodology

The proposed methodology, as illustrated in Fig. 1, outlines the overall process for finding the optimal model for different algorithms. The dataset is necessary to be generated and preprocessed accordingly for training and testing ML models. Therefore, hyperparameter tuning is necessary, the training data are utilized to build models based on different algorithms, employing various combinations of hyperparameters. Concurrently, the performance of trained models is evaluated by the testing data in optimal model selection. By using

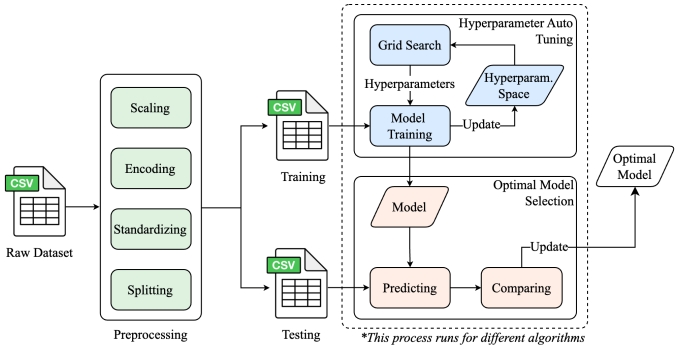

Fig. 1. Overall methodology to find the optimal model of different algorithms

the ranking algorithm, the performance metrics are compared among all the models to select the optimal models of the 3 algorithms. Then the best-performed algorithm for the NoC performance modeling can be identified by comparing these optimal models.

A. Dataset

To develop an accurate and effective ML model, a bespoke cycle-accurate simulator, including the reliability-related parameters, is employed as the golden reference to generate the dataset [11]. To efficiently evaluate the performance of the ML model in solving this problem, the combination of inputs for the simulator should be judiciously selected based on the correlation of input parameters and the hardware limitation of the host PC. Based on insights from empirical studies [8]–[10], the key parameters that significantly impact the NoC performance can be identified. These include topology, network size, injection rate, packet size, routing algorithm, etc. The parameter selection for ML modeling depends on their impact on the NoC performance outcomes, especially the average delay. In this study, reliability-related parameters such as ECC schemes and BERs are two of the main focuses. Table I shows the parameters configuration for the simulator, where four parameters (i.e. network size, injection rate, BER, and ECC scheme) are manipulated to form different NoC architectures. Network size and injection rate have the most impact on the performance of the NoC. The higher BER range from 10^{-2} to 10^{-9} is chosen for the ML modeling since smaller BER (i.e. $< 10^{-9}$) does not have a significant impact on NoC performance. All the supported ECC schemes by the simulator are selected, which include Forward Error Correction (FEC), Automatic Repeat Request (ARQ), and Hybrid ARQ (HARQ).

Data preprocessing is required for the inputs and labels to improve the result accuracy and optimize the training of ML models as the data have wildly different ranges [12]. In this study, several data preprocessing techniques are used for the ML training process. One-hot encoding is required to transform categorical data into one-dimensional numerical vectors, enabling efficient learning from string-like data like ECC schemes. Logarithmic scaling (i.e. log_{10}) is implemented

TABLE I
CONFIGURATION OF SIMULATOR

Input	Parameter	Interval
Constant Parameters		
Topology	Mesh	-
Routing Algorithm	XY	-
Traffic Pattern	Uniform	-
Injection Type	Periodic	-
Buffer Size	8	-
Number of Virtual Channels	4	-
Packet Size	8 Flits	-
Flit Size	32 Bits	-
Sweeping Parameters		
Network Size	2 - 9	+1
Injection Rate	0.01 - 0.91	+0.05
BER	$10^{-2} - 10^{-9}$	$\times 10^{-1}$
ECC Scheme	DUP, DAP, DTDP_7ED, DTDP_SEC6ED, JTEC, SQED	Sweep

for features that have an ultra-wide range of magnitude, such as BER. Standardization is also important for algorithms that are sensitive to feature scaling. This can be achieved by subtracting the mean (μ) from the features and dividing by its standard deviation (σ).

For the model train-test splitting, 80% is allocated for the training set, while the remaining 20% is reserved for testing. This division ensures a substantial amount amount of data is used for model training, while still retaining a portion for evaluating the generalization of the trained model.

B. Evaluation Metrics

For the regression problem, two evaluation metrics, coefficient of determination (R^2) and MAE, are employed to describe the accuracy of the predicted results. Another key metric is runtime speedup, which is aimed at measuring the improvement of computation time consumption.

R^2 quantifies the proportion of the variance in the dependent variable that is predictable from the independent variables. A R^2 of 1 indicates the flawless fitting of the predicted value and original data. R^2 can be calculated as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (1)$$

where y_i and $\hat{y}_i$ are the actual and predicted values respectively, and $\bar{y}$ denotes the average value of the testing data. n is the total number of samples in the testing data.

MAE measures the average magnitude of errors between predicted and actual values. With a lower MAE, the model shows a higher accuracy in the prediction. The calculation can be expressed by

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \quad (2)$$

where y_i and $\hat{y}_i$ are the actual and predicted values respectively, and $\bar{y}$ denotes to the average value of the testing data. n is the total number of samples in the testing data.

TABLE II
HYPERPARAMETER SPACE OF RF, XGBOOST AND MLP

Hyperparameter	Value
Random Forest	
n_estimators	50, 100, 200, 300
max_features	1, 2, 3, 4
max_depth	None, 3, 5, 7, 10
XGBoost	
max_depth	None, 3, 5, 7, 10
eta	0.01, 0.05, 0.1, 0.2
gamma	0, 0.1, 0.5, 1.5
Multilayer Perceptron	
number_of_neurons	[64, 32, 16], [128, 64, 32], [256,128, 64]
activation_function	Sigmoid, ReLU, Tanh
learning_rate	0.001, 0.003, 0.005, 0.007, 0.009

To measure the runtime speedup of ML models, the equations below is utilized to obtain the average speedup compared with the simulator.

$$Speedup = \frac{\sum_{i=1}^{n}(t_i/\hat{t_i})}{n} \tag{3}$$

where t_i is the actual runtime using the simulator, and $\hat{t_i}$ is the runtime of predicting using trained ML models.

C. Optimal Model Selection

Hyperparameter auto-tuning by using the Grid Search (GS) algorithm is performed to further optimize the ML model prediction accuracy before the model evaluation and selection. The hyperparameter space for different ML algorithms in the experiment is listed in the Table II. Different machine learning models consist of different hyperparameters to tune when optimizing the model. The trained models with different hyperparameters and their results of R^2 and MAE on the testing dataset are recorded for the GS algorithm implementation. In the optimal model selection, a ranking algorithm based on the comparison of performance metrics of trained models is used to find the optimal model from the three ML algorithms.

III. EXPERIMENTAL RESULTS

Throughout the optimal model selection based on the pre-processed dataset using the proposed ReNoC-ML framework, XGBoost shows an outstanding performance in both accuracies (i.e., the highest R^2 and lowest MAE) and runtime as summarized in Table III. To verify the accuracy of the optimal XGBoost model, two characteristic plots of NoC ECC schemes are generated. First, the plot of average delay versus injection rate shows how the performance varies when the data volume per unit cycle is increasing. In the experiment, the range of BER from 10^{-2} to 10^{-7} was used for generating the plots as shown in Fig. 2, different ECC schemes have more varying inflection points for average delay at larger BER. The fitting of the prediction is overall satisfying. However, it can be observed that there are inaccurate prediction that usually happens in the inflection point for large BER like 10^{-2} and 10^{-3}.

TABLE III
OPTIMAL MODEL RESULTS FOR EACH ALGORITHM

Algorithm	R^2	MAE	Speedup
RF	0.9271	830.8500	$\times 10^5$
XGBoost	**0.9502**	**616.4072**	$\times \mathbf{10^7}$
MLP	0.9375	795.8215	$\times 10^6$

*The result was evaluated by the testing dataset.

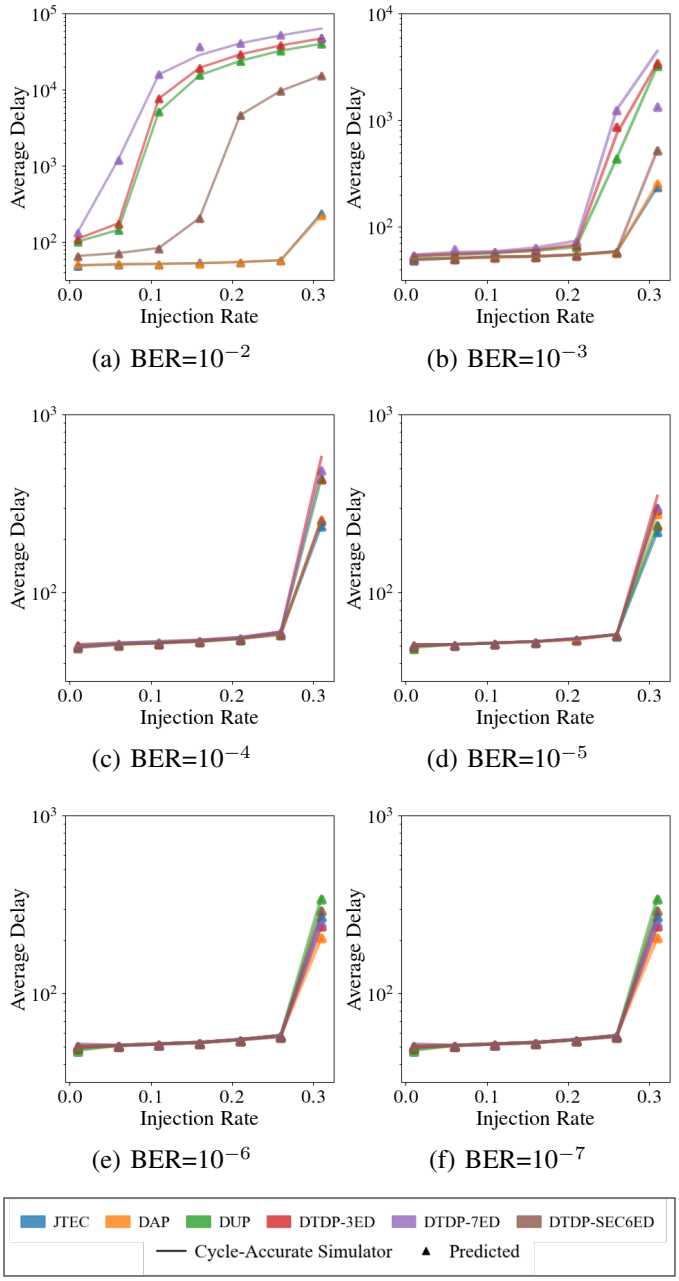

Fig. 2. Average delay vs injection rate under different ECC schemes

The plot for smaller BERs shows better fitting results from the observation. The reason may be due to the imbalanced dataset for different BER values. This can be observed from

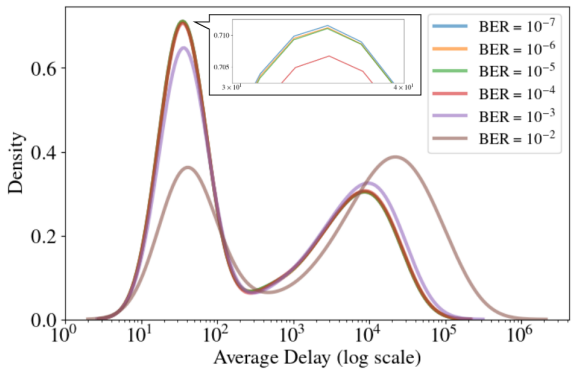

Fig. 3. KDE plot for average delay under different BERs

the Kernal Density Estimation (KDE) in Fig. 3, which shows the distribution of average delay for different BER values in the training dataset. The data with smaller BER (i.e., 10^{-4}-10^{-7}) have a similar distribution for the average delay, which can be considered as the majority in the dataset, while the distribution of data at larger BER (i.e., 10^{-2} and 10^{-3}) are the minority. Even though preprocessing techniques have been implemented to optimize the distribution, the impacts still existed. It is also the reason why XGBoost has the best performance out of the three algorithms. In RF and MLP, the impact was enlarged since they did not have a strategy to deal with the issue. XGBoost performs better since it integrates the following characteristics that can help mitigate the impact of imbalanced data: tree-based structure, built-in regularization, and scale in-variance. The same reason could be used to explain the reason for the inaccuracy in inflection points. It shows that the distribution is bimodal, which means it has two distinct peaks at both low values and very high values. Therefore, the density of the inflection point is quite low, which causes another imbalanced situation. This imbalance of the dataset usually makes the algorithm hard to learn accurately in this domain and causes the erroneous prediction [13].

Another characteristic is presented by average delay versus BER shown in Fig. 4 which contains the plots with different injection rates from 0.01 to 0.26. It can be observed that the accuracy is acceptable as there are only a few inaccurate predictions with small differences. On the other hand, the prediction points show an expected trend of average delay affected by increasing BER for different ECC schemes.

IV. Conclusion

This paper demonstrates a rapid and accurate NoC performance prediction involving reliability-related parameters (i.e. BER and ECC scheme) through the ML approach. Three different ML approaches, RF, XGBoost, and MLP, were implemented with certain hyperparameter tuning strategies to improve the prediction accuracy. XGBoost shows up to 10^7 times speedup as compared to the cycle-accurate simulator and the highest R^2 score of 0.95. The proposed ML-based NoC

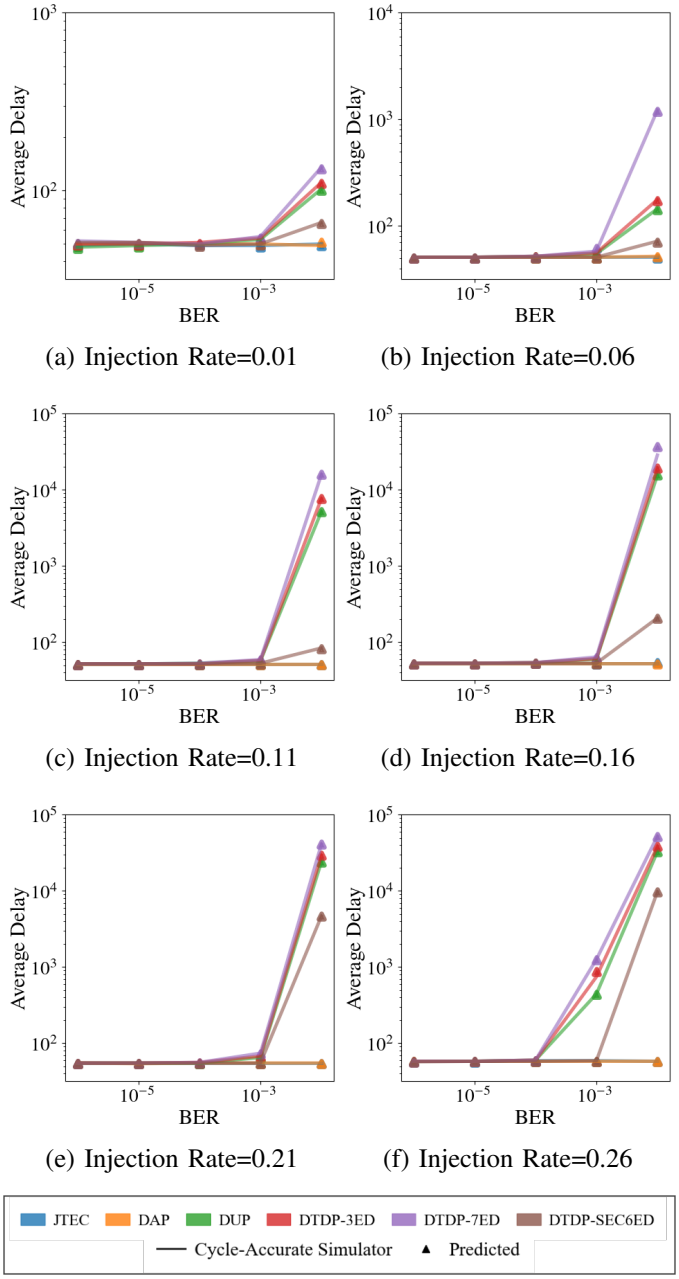

Fig. 4. Average delay vs BER under different ECC schemes

performance predictor ReNoC-ML can be used for NoC design parameter optimization via the DSE flow. This work can be further extended by addressing the imbalanced dataset and expanding the prediction domain with different parameters.

References

[1] N. E. Jerger, T. Krishna, and L.-S. Peh, *On-Chip Networks: Second Edition*, 2nd ed. Morgan & Claypool Publishers, 2017.

[2] W. N. Flayyih, K. Samsudin, S. J. Hashim, F. Z. Rokhani, and Y. I. Ismail, "Crosstalk-Aware Multiple Error Detection Scheme Based on Two-Dimensional Parities for Energy Efficient Network on Chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 7, pp. 2034–2047, 2014.

[3] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate Network-on-Chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.

[4] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Improving the energy efficiency of wireless Network on Chip architectures through online selective buffers and receivers shutdown," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE Press, 2016, pp. 668–673.

[5] C. Tan, Y. Ou, S. Jiang, P. Pan, C. Torng, S. Agwa, and C. Batten, "PyOCN: A Unified Framework for Modeling, Testing, and Evaluating On-Chip Networks," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, 2019, pp. 437–445.

[6] J. Wang, Y.-b. Li, and C. Wu, "An analytical model for Network-on-Chip with finite input buffer," *Frontiers of Computer Science in China*, vol. 5, pp. 126–134, 2011.

[7] A. V. Bhaskar and T. Venkatesh, "Performance analysis of network-on-chip in many-core processors," *Journal of Parallel and Distributed Computing*, vol. 147, pp. 196–208, 2021.

[8] A. Kumar and B. Talawar, "Machine Learning Based Framework to Predict Performance Evaluation of On-Chip Networks," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*, 2018, pp. 1–6.

[9] B. Bhowmik, P. Hazarika, P. Kale, and S. Jain, "AI Technology for NoC Performance Evaluation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 12, pp. 3483–3487, 2021.

[10] Y. R. Muhsen, N. A. Husin, M. B. Zolkepli, N. Manshor, A. A. J. Al-Hchaimi, and H. M. Ridha, "Enhancing NoC-Based MPSoC Performance: A Predictive Approach With ANN and Guaranteed Convergence Arithmetic Optimization Algorithm," *IEEE Access*, vol. 11, pp. 90 143–90 157, 2023.

[11] W. N. Flayyih, "Crosstalk Aware Error Control Coding Techniques for Reliable and Energy Efficient Network on Chip," Ph.D. dissertation, Universiti Putra Malaysia, 2014.

[12] F. Chollet, *Deep Learning with Python*. Manning Publications, 2021.

[13] Y. Yang, K. Zha, Y. Chen, H. Wang, and D. Katabi, "Delving into deep imbalanced regression," in *International conference on machine learning*. PMLR, 2021, pp. 11 842–11 851.