AI Technology for NoC Performance Evaluation

Biswajit Bhowmik, *Senior Member, IEEE*, Pallabi Hazarika, Prachi Kale, and Sajal Jain

***Abstract*—An on-chip network has become a powerful platform for solving complex and large-scale computation problems in the present decade. However, the performance of bus-based architectures, including an increasing number of IP cores in systems-on-chip (SoCs), does not meet the requirements of lower latencies and higher bandwidth for many applications. A network-on-chip (NoC) has become a prevalent solution to overcome the limitations. Performance analysis of NoC's is essential for its architectural design. NoC simulators traditionally investigate performance despite they are slow with varying architectural sizes. This work proposes a machine learning-based framework that evaluates NoC performance quickly. The proposed framework uses the linear regression method to predict different performance metrics by learning the trained dataset speedily and accurately. Varying architectural parameters conduct thorough experiments on a set of mesh NoCs. The experiments' highlights include the network latency, hop count, maximum switch, and channel power consumption as 30-80 cycles, 2-11, 25μW, and 240μW, respectively. Further, the proposed framework achieves accuracy up to 94% and speedup of up to 2228×.**

***Index Terms*—Networks-on-chip, machine learning, linear regression, dataset, performance evaluation.**

I. Introduction

NOWADAYS, multiprocessor SoCs (MPSoCs) cannot meet the high performance and scalability requirements to handle the complicated on-chip communications due to communication bottleneck. An NoC fabric has become a prevalent solution interconnecting hundreds to thousands of cores in many-core systems [1], [2]. The primary way of evaluating an NoC architecture is to measure its performance. The performance is traditionally evaluated via simulations [3]. Researchers use various cycle-accurate NoC simulators like BookSim, Nirgam, Noxim, etc. [4]. However, the time increases for providing the results by these simulators while the NoC size increases. Thus, there is a need to develop a faster performance evaluation scheme for NoCs.

Conventional performance measurement approaches rely on simulations at lower abstraction levels to run the architectures for accurate results. But running at lower abstraction levels takes a long time. On the other hand, running simulations at higher abstraction levels is time efficient but could not deliver accurate results [5]. So traditional approaches can be avoided using alternate technology like machine learning (ML)-based methods. Recent researches in this way are discussed in [4], [5], [6], [7], [8], [9], [10], [11], [12]. These schemes are comparative to each other. However, they are not adequate to the expected level. For example, their prediction error and speedup ranges from 9-45% and 2-1300×, respectively, including high evaluation time. It motivates to design a practical framework to improve these parameters and quickly estimate vagarious performance metrics in NoCs of larger size.

This brief proposes an ML-based performance evaluation scheme for NoCs. The scheme uses a Linear Regression algorithm trained by data collected from the BookSim simulator. Several rounds of experiments are conducted at a varying system configuration to predict multiple performance parameters for a set of mesh NoCs. The anticipated results by the proposed scheme highlight that average network latency and hop count are about 30-80 cycles, and 2-11, respectively. Other metrics like the total area, maximum switch, channel, and total power consumption are predicted as 0.08-5μm^2, 25μW, 240μW, and 300μW, respectively. The advantages of the proposed approach are that it achieves a minimum speedup of 260× and a maximum speedup of 2228× compared to the BookSim simulator. Also, the prediction error lies in the range of 6-8% resulting in up to 94% accuracy. Compared to previous works, the proposed prediction framework provides up to 44% more accuracy and renders up to 2200× more speedup than the previous ones. Note that these improvements grow with the NoC size.

The rest of this brief is organized as follows. Section II presents the proposed scheme. Section III provides the experimental results. Section IV concludes this brief.

II. Proposed Work

Artificial intelligence (AI) technology such as machine learning has surpassed human efforts and become a significant milestone for many applications, including VLSI systems and applications like performance evaluation in NoCs. The proposed performance evaluation framework using a machine learning technique is based on the linear regression algorithm. The proposed framework is expected to predict multiple performance metrics in an NoC having a large size and increase the computing flexibility of the NoC platform.

A. Linear Regression Framework for NoCs

Linear Regression (LR) is a supervised machine learning algorithm that predicts continuous/actual or numeric values. It is advantageous and easier to interpret, implement, and train efficiently. It handles overfitting well using reduction techniques, regularization, and cross validation [13] dimensionally.

Manuscript received September 15, 2021; revised October 23, 2021; accepted October 25, 2021. Date of publication November 1, 2021; date of current version November 24, 2021. This brief was recommended by Associate Editor Y. Ha and E. Bonizzoni. *(Corresponding author: Biswajit Bhowmik.)*

The authors are with the Department of Computer Science and Engineering, National Institute of Technology Karnataka, Mangalore 575025, India (e-mail: brb@nitk.edu.in).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3124297.

Digital Object Identifier 10.1109/TCSII.2021.3124297

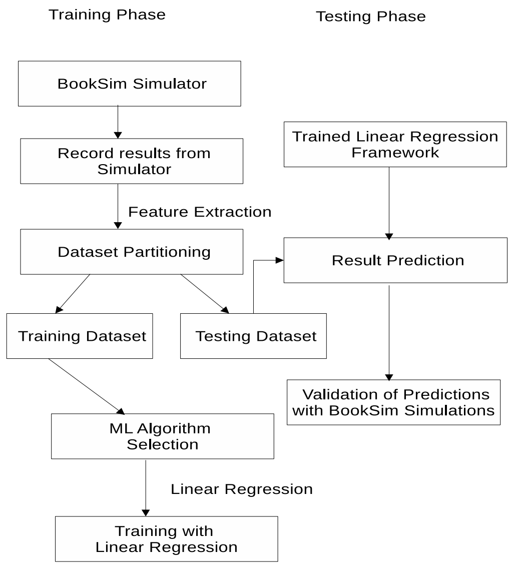


Fig. 1. Proposed LR Framework.

The LR has two parameters- weight (regression coefficient) *m* and bias *c*. Regression coefficients are estimates of the unknown dataset parameters. In other words, it is the scale factor to each input data. Bias is a factor that offsets all predictions. The prepared data is divided into “attributes” and “labels”. Attributes are the independent variables, while labels are dependent variables whose values are predicted–the algorithm models a target prediction value for the dependent variables (labels) based on the attributes. We need to check for variables that have some linear relationship between the labels and the attributes from our dataset. And accordingly, we select the attributes that are required to build the model. By training our model on the selected pair of attributes and labels, the algorithm learns the best combination of *m* and *c*. The best combination of values makes the most accurate predictions with a minor error. Training gets completed when the model returns an acceptable error value. The model finds the best-fit regression line that best fits the data points. The evaluation framework calculates this metric through the mean square error (MSE) cost function. The metric finds the error (difference) between the actual results and the predicted results.

B. Designing Prediction Framework

Figure 1 describes the steps for designing the proposed prediction framework. The framework consists of two phases: the training and testing phase. A brief description of these phases is stated here.

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(y_t^* - y_t)^2 \tag{1}$$

- *Training Phase*-The training phase is the first phase of the prediction model. In this phase, the training dataset for an NoC is collected on running a simulator for the network. The simulations evaluate multiple performance metrics, which are fed for training then for testing too. Algorithm 1 reveals the procedure of training dataset generation and collection. This work employs BookSim 2.0 simulator to generate the training dataset for the set of NoCs taken. Various performance metrics are measured via simulations, and the proposed scheme predicts the same. These parameters are average packet latency (APL), average network latency (ANL), avergare hop count (AHC), channel wire power consumption (CWPC), switch power consumption (SPC), total power consumption (TPC), and total area (TA). The collected dataset is now ready to train the model using the Linear Regression algorithm. In other words, the model learns the parameters from the trained dataset.
- *Testing Phase*-The second phase is the testing phase. The underlying regression model is trained to validate the results predicted for networks under consideration. The validation in this phase is mainly performed by comparing the simulation and predicted results generated for the networks.

Algorithm 1 Dataset_Generation

```
procedure DatasetGenerator()
Set V← setVirtualChannel(), B← setBufferSize(), R← setRoutingAlgorithm(), T← setTopologyType(), F← setTrafficPattern(), I← setInjectionRate(), S← setTopologySize(), P← setPacketSize(), and A← setSamplePeriod()

C← loadConfiguration(V,B,R,T,F,I,S,P,A)
for each(s∈ S|S={2×2, ..., 15×15}) do
 for each(t∈ T|T={uniform,tornado}) do
  for each(v∈ V|V={2,3,4,5}) do
   for each(b∈ B|B={6,8}) do
    for each(i∈ I|I={0.001,..,0.009}) do
        beginSimulation(command, C)
        result←read(APL,ANL,AHC,
                CWPC,SPC,TPC,TA)
        dataset←appendResult(result)
    end for
   end for
  end for
 end for
end for
end procedure
```

The simulation results may differ from the predicted results by the framework for an NoC. This difference in the MSE is measured using the Gradient descent. The MSE is defined in Equation (1). Here, N = number of data points, y_t is the actual value, and y_t^* is the predicted output. Gradient descent updates LR parameters by minimizing the cost function and reducing MSE by evaluating the cost function’s gradient. Pseudocode for the proposed LR framework is stated in Algorithm 2 where one finds how the MSE gets calculated.

III. Experimental Results

This section evaluates the performance of NoC architectures by the proposed learning-based framework. A set of 2D mesh NoCs with a size that ranges from 2 × 2 − 15 × 15

Algorithm 2 Linear_Regression_Framework

```
procedure LinearRegressionFramework()
loadFile(dataset)
Partition dataset into train and test data
Determine Attributes and Labels
Training and testing
    regressor.fit(train data)
    regressor.predict(test data)
Framework Evaluation
    calculateMSE(test data, prediction data)
    timingComparison(simulation time, framework time)

end procedure
```

TABLE I
BookSim Simulation Configuration Setup

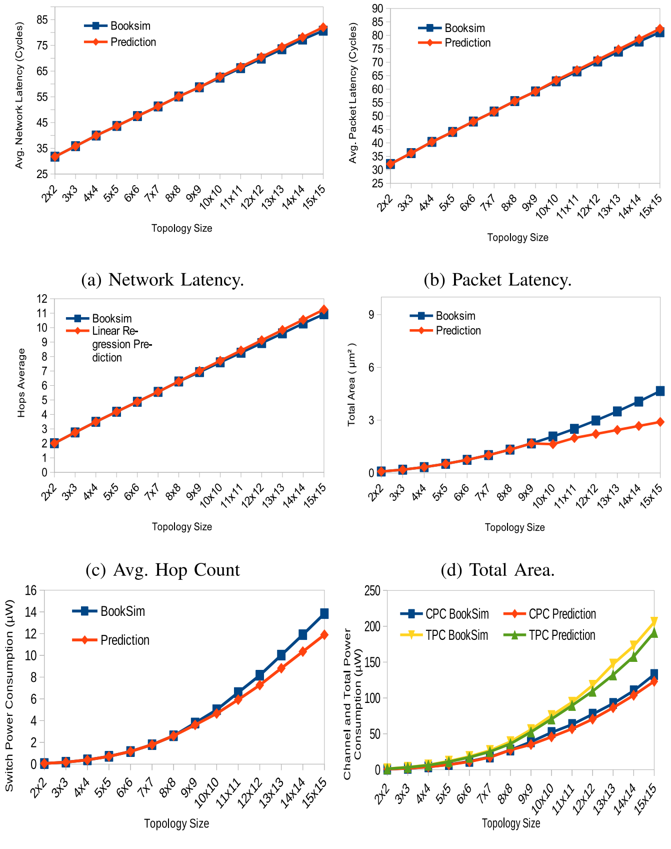
Topology Type	2D Mesh
Network size	2×2 - 15x15
Traffic Pattern	Uniform, Tornado
Routing Algorithm	XY
Virtual Channel	2,3,4,5
Buffer Size	6,8
Packet Size	20 flits
Sample Period	100000 cycles
PIR	0.001, 0.0015, ..., 0.009

are put under the evaluation. First, we briefly describe the experimental details and training dataset preparation. Then, we deeply analyze the predicted results, followed by validating the results. Next, a comparison of the time consumed by simulation and the proposed methods for evaluating the performance is discussed. Finally, the proposed scheme compares its achievements with the existing approaches.

A. Experimental Setup and Dataset Preparation

A cycle-accurate NoC simulator, namely BookSim 2.0, is modified to generate the training dataset. The BookSim configuration at a glance is provided in Table I. The dataset is generated on several architectural parameters like virtual channels (VC), buffer size, topology size, and packet injection rate (PIR). Every time a simulation is run for an extended period of 100000 cycles. The deadlock-free XY routing algorithm transmits the traffic injected using the Uniform and Tornado traffic patterns. The traffic in terms of packets is transferred. Each packet size is 20 flits. Entire experiments are accomplished in the computer system with the configuration: Core i7, 2.6 GHz, 8 GB RAM, and Ubuntu 18.04 LTE OS.

The entire dataset is partitioned into two categories: training and testing dataset. The values generated from the simulations on the 2×2 to 8×8 mesh NoC are used as training data. The values generated for the rest of the network of size 9×9 to 15×15 are exercised as the testing data. Further, the training or testing type dataset is generated at the packet injection rate (PIR) in the range of 0.001 to 0.009, whereas the simulations run at each PIR value measure the performance metrics. As stated, the estimated metrics for the meshes up to the network size 8×8 are exploited as the training data. In contrast, the measured metrics for the remaining networks are used as the testing dataset to validate the metrics predicted for them.



(a) Network Latency.

(b) Packet Latency.

(c) Avg. Hop Count

(d) Total Area.

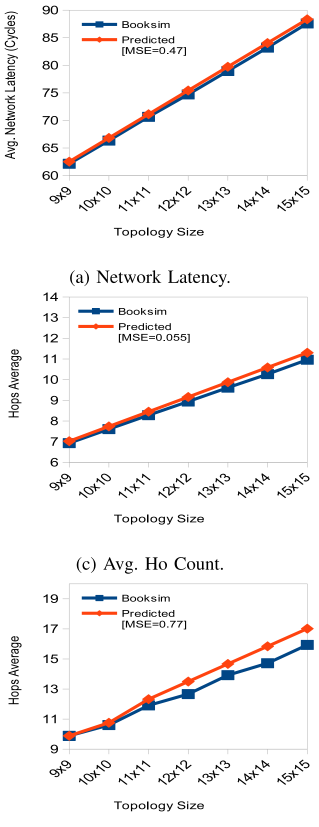
(e) Switch Power.

(f) Channel and Total Power.

Fig. 2. Learned and predicted values by the proposed scheme at Traffic = Uniform, VC = 4, PIR = 0.002, and Buffer = 8.

B. Result Analysis

The effectiveness of the proposed linear regression-based technique for NoC performance evaluation is established by predicting the performance metrics over the simulation method. Figure 2 shows the predicted results over the simulations for seven performance parameters-average network and packet latency, average hop count, total area, switch, channel, and total power consumption in several NoCs. In Figure 2, results labeled by BookSim represent simulated values for the topologies. The values marked by Prediction for $2 \times 2 - 8 \times 8$ NoCs show the level of learning by the proposed model. The values labeled by Prediction for the rest of the NoCs provide the predicted metrics. As shown in Figure 2(a) and 2(b), the proposed framework predicts the average network and packet latency as 59.62 and 60.15 cycles, respectively, for the 9×9 mesh NoC. The same parameters for the 15×15 mesh NoC are about 80-82 cycles. Prediction of other metrics- average hop count and total area on the 9×9 mesh NoC are 6.93 and 1.68 μm^2, respectively. These metrics are seen in Figures 2(c) and 2(d), respectively. The same prediction in Figures 2(e) and 2(f) shows 3.60 μW, and 35.04 μW and 52.54 μW as the switch, and channel and total power consumption in 9×9 mesh NoC. It shows that the prediction and the simulation provide



(a) Network Latency.

(b) Packet Latency.

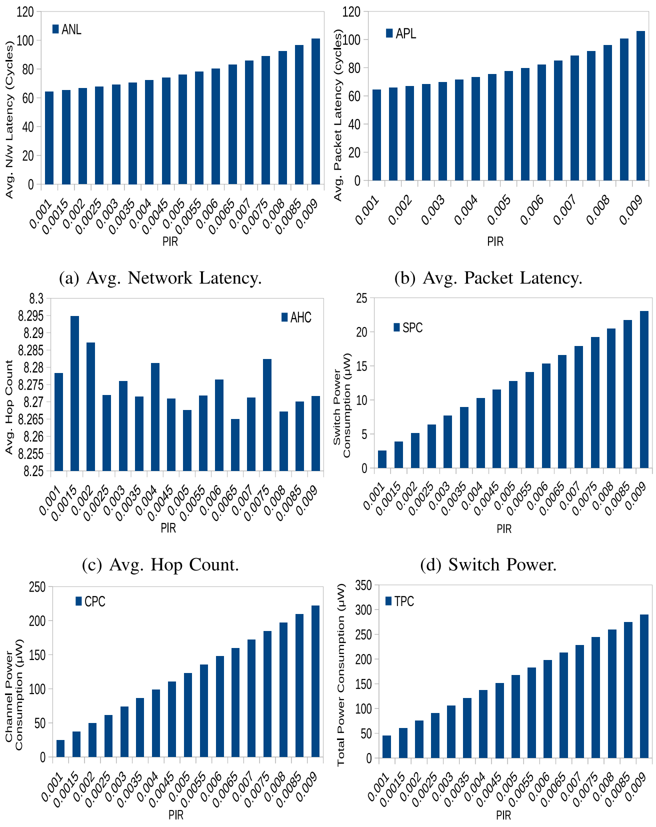
(c) Avg. Ho Count.

(d) Total Area.

(e) Avg. Hop Count.

(f) Total Area.

Fig. 3. Predicted results by the proposed scheme at (a-d) Traffic = Uniform, VC = 3, PIR = 0.0025, and Buffer = 6; (e-f) Traffic = Tornado, VC = 4, PIR = 0.0015, and Buffer = 8.



(a) Avg. Network Latency.

(b) Avg. Packet Latency.

(c) Avg. Hop Count.

(d) Switch Power.

(e) Channel Power.

(f) Total Power.

Fig. 4. Predicted performance metrics at Traffic = Uniform, VC = 5, Buffer = 8 on the 11 × 11 mesh NoC.

nearly the same result. In other words, our model gets trained properly on the training dataset and mitigates the predicted and simulated data gap. One can observe this mitigation on analysis of the testing data with the predicted results.

Experiments are accomplished in varying environments to explore the proposed LR-model more. For instance, the same set of performance parameters are evaluated at the experimental configuration of VC = 4, buffer = 8, PIR = 0.002, and Traffic = Uniform. When the setup is partially modified, say VC = 3, buffer = 6, PIR = 0.0025, one observes the predicted performance metrics over the simulated as shown in Figure 3. In this case, the average network and packet latency for varying topology sizes is about 62.52-88.34 cycles and 63.60-89.58 cycles, respectively. Further, the average hop count and total areas are 7-11 and 1.45-2.70 μm^2, respectively. As observed, the highest and lowest mean square error is 0.055 and 0.96, respectively. The experiment is extended and performed at another configuration where VC = 4, buffer = 8, PIR = 0.0015, and Traffic = Tornado. Then average hop count and total area metrics can be observed as seen in Figure 3(e) and 3(f), respectively. The highest and lowest mean square error are 0.96 and 0.77, respectively.

Figure 4 shows the prediction of different metrics on the 11 × 11 mesh NoC when our LR framework is applied on this network. Evaluations are made at the experimental configurations VC = 5, buffer = 8, PIR = 0.001-0.009, and Traffic =Uniform. As observed, the network and packet latency ranges from 64.26−101.15 cycles (Figure 4(a)) and 64.45-105.92 cycles (Figure 4(b)), respectively, whereas the range 8.26502-8.29486 (Figure 4(c)) shows the average hop count on this network. Subsequently, amounts of power consumed by different NoC components are seen in Figure 4(d) to 4(f) that show the average switch, channel, and total power consumption, respectively, evaluated using the proposed LR-model.

C. Validation of Predicted Results

The effectiveness of the proposed ML-based performance evaluation can be further verified by analyzing the predicted results using the proposed framework with the simulation results using the BookSim. The optimal value of weight m and bias c are learned by training our proposed model. The results are predicted using their optimal values. Predicted results are then compared with BookSim simulation results using the error metric MSE. As seen in Figure 2 through 4, the proposed framework approximates the evaluated performance metrics about 92-94%. In other words, the prediction error by the proposed model is 6-8%. Thus, the current NoC performance evaluation scheme achieves an acceptable effectiveness level for most performance metrics.

TABLE II
COMPARISON ON THE EVALUATION TIME

Mesh Size	Booksim Time (Sec)	Predicted Time (Sec)	Speedup (×)
9 × 9	56.42	0.2174	259.5
10 × 10	60.81	0.21987	276.57
11 × 11	84.50	0.2529	334.1
12 × 12	209.77	0.2257	929
13 × 13	282.3	0.2175	1,297.93
14 × 14	318.3	0.2306	1380.3
15 × 15	516	0.2316	2228

TABLE III
COMPARATIVE STUDY WITH PREVIOUS WORKS

Previous Works	Accuracy	Speedup	Time (sec)
Wang *et al.* [4]	88%	–	–
Silva *et al.* [5]	55–70%	–	–
Qian *et al.* [6]	88%	120×	8
Liao *et al.* [7]	62–88%	–	–
Ping *et al.* [8]	82%	–	–
Chen *et al.* [9]	85–90%	2 – 3×	–
Hou *et al.* [10]	93.32%	153×	3.7
Kumar *et al.* [11]	68%	1300×	3
Poposed LR-Model	92–94%	260–2228×	0.217–0.231

D. Comparison Study

The ML-based prediction framework is designed here to quickly evaluate NoCs over a traditional simulation-based evaluation. The framework is compared with the simulation-based method and previous works based on different AI techniques. Table II draws a comparison between the time taken by the BookSim simulator and the same taken by the designed framework for different metrics measured earlier in this section. As shown, the BookSim simulations take 56.42-516 seconds for evaluating the metrics on the mesh NoCs having the sizes 9×9 to 15×15. On the other hand, the same metrics for the same networks are predicted by our LR framework in just 0.2174-0.2529 seconds, resulting in a speedup of 260-2228× over the BookSim simulations. This speedup grows with the network size.

Different AI-enabled techniques nowadays avoid traditional simulation-based NoC performance evaluation. These are a compliment to each other. However, our proposed LR model outperforms many of them. Table III provides a relative performance of a set of previous works [4], [5], [6], [7], [8], [9], [10], [11] with our approach. The comparisons are studied regarding three factors: accuracy, speedup, and prediction time. As seen, most of these works have not considered the speedup and the prediction time in the evaluations. Further, most of them achieve an accuracy below 90%. Few works measured the speedup, including accuracy and prediction time, but their achievement is partially disappointing. For example, Although an ML-based performance evaluation scheme [10] achieves 93.32% accuracy, but provides speedup up to 153× and takes about 3.7 sec to predict the performance metrics on evaluating an 8×8 mesh. Another SVR model [11] estimates accuracy, speedup, and prediction time as 68%, 1300×, and 3 sec, respectively, on the 15×15 mesh NoC. These parameters grow up to 92-95%, 1500-2000×, and 4-8 sec when evaluated meshes with size 16×16 to 30×30. On the contrary, our proposed LR framework achieves comparatively high accuracy and speed up concerning the previous works discussed above. Further, our approach takes significantly very little time to predict the performance metrics. Thus, one can say that our scheme provides about 44% and 2200× more accuracy and speedup than the previous works.

IV. CONCLUSION AND FUTURE WORK

A Linear Regression-based framework has been proposed to predict quickly and accurately several performance parameters such as latency, hop count, power consumption, and area for many meshes NoCs. The framework is learned with the dataset supplied by the cycle-accurate BookSim simulator at varying simulation setups. As a result, the framework has achieved an accuracy level of up to 94% and speedup up to 2228×. It is more effective than a wide range of existing works. The future work aims to investigate the model's scalability and improve the accuracy by analyzing other learning algorithms for evaluating the performance metrics and extending the scheme on 3D-NoC architectures.

REFERENCES

[1] H. Zheng and A. Louri, "Agile: A learning-enabled power and performance-efficient network-on-chip design," *IEEE Trans. Emerg. Topics Comput.*, early access, Jun. 18, 2020, doi: 10.1109/TETC.2020.3003496.

[2] Z.-L. Qian, D.-C. Juan, P. Bogdan, C.-Y. Tsui, D. Marculescu, and R. Marculescu, "A support vector regression (SVR)-based latency model for network-on-chip (NoC) architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 3, pp. 471–484, Mar. 2016.

[3] C. Xu, Y. Liu, and Y. Yang, "SRNoC: An ultra-fast configurable FPGA-based NoC simulator using switch–router architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2798–2811, Oct. 2020.

[4] K.-C. Chen and T.-Y. Wang, "NN-Noxim: High-level cycle-accurate NoC-based neural networks simulator," in *Proc. 11th Int. Workshop Network Chip Archit. (NoCArc)*, 2018, pp. 1–5.

[5] J. Silva, M. Kreutz, M. Pereira, and M. Da Costa-Abreu, "An investigation of latency prediction for NoC-based communication architectures using machine learning techniques," *J. Supercomput.*, vol. 75, no. 11, pp. 7573–7591, 2019.

[6] Z.-L. Qian, D.-C. Juan, P. Bogdan, C.-Y. Tsui, D. Marculescu, and R. Marculescu, "A support vector regression (SVR)-based latency model for network-on-chip (NoC) architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 3, pp. 471–484, Mar. 2015.

[7] K.-C. J. Chen and Y.-H. Liao, "Adaptive machine learning-based temperature prediction scheme for thermal-aware NoC system," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2020, pp. 1–4.

[8] L. B. Ping, C. P. Kit, and E. K. Karuppiah, "Network latency prediction using high accuracy prediction tree," in *Proc. 7th Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2013, pp. 1–8.

[9] Y. Chen and A. Louri, "Learning-based quality management for approximate communication in network-on-chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3724–3735, Nov. 2020.

[10] J. Hou, Q. Han, and M. Radetzki, "A machine learning enabled long-term performance evaluation framework for NoCs," in *Proc. IEEE 13th Int. Symp. Embedded Multicore/Many-Core Syst.-on-Chip (MCSoC)*, 2019, pp. 164–171.

[11] A. Kumar and B. Talawar, "Machine learning based framework to predict performance evaluation of on-chip networks," in *Proc. 11th Int. Conf. Contemp. Comput. (IC3)*, 2018, pp. 1–6.

[12] H. Zheng and A. Louri, "An energy-efficient network-on-chip design using reinforcement learning," in *Proc. 56th Annu. Design Autom. Conf.*, 2019, pp. 1–6.

[13] D. Wang and J. Xu, "On sparse linear regression in the local differential privacy model," *IEEE Trans. Inf. Theory*, vol. 67, no. 2, pp. 1182–1200, Feb. 2021.