Proceedings of 2018 Eleventh International Conference on Contemporary Computing (IC3), 2-4 August, 2018, Noida, India

Machine Learning Based Framework to Predict Performance Evaluation of On-Chip Networks

Anil Kumar
SPARK Lab, Dept of Computer Science and Engineering
National Institute of Technology Karnataka
Mangalore, India.
anilkumar.cs14f05@nitk.edu.in

Basavaraj Talawar
SPARK Lab, Dept of Computer Science and Engineering
National Institute of Technology Karnataka
Mangalore, India.
basavaraj@nitk.edu.in

***Abstract*—Chip Multiprocessors(CMPs) and Multiprocessor System-on-Chips(MPSoCs) are meeting the ever increasing demand for high performance in processing large scale data and applications. There is a corresponding increase in the volume and frequency of traffic in the Network-on-Chip(NoC) architectures like CMPs and SoCs. NoC performance parameters like network latency, flit latency and hop count are critical measures which directly influence the overall performance of the architecture and execution time of the application. Unfortunately, cycle-accurate software simulators become slow for interactive use with an increase in architectural size of NoC. In order to provide the chip designer with an efficient framework for accurate measurements of NoC performance parameters, we propose a Machine Learning(ML) framework. Which is designed using different ML regression algorithms like Support Vector Regression(SVR) with different kernels and Artificial Neural Networks(ANN) with different activation functions. The proposed learning framework can be used to analyze the performance parameters of Mesh and Torus based NoC architectures. Results obtained are compared against the widely used cycle-accurate Booksim simulator. Experiments were conducted by variables like topology size from 2×2 to 30×30 with different virtual channels, traffic patterns and injection rates. The framework showed an approximate prediction error of 5% to 8% and overall minimum speedup of 1500× to 2000×.**

Keywords: Network-on-Chip, Machine Learning, Support Vector Regression, Booksim, Artificial Neural Network, Latency, Hop Count.

I. INTRODUCTION

NoC architectures are high performance and scalable which are capable of handling complicated On-Chip communications [1], [2]. NoC researchers use cycle-accurate performance and power simulators viz. Booksim2.0[3], Noxim[4], SICOSYS[5], Nirgam[6], Nocsim[7] to explore the micro-architectural design space of NoC's well before the actual hardware implementation. Among these, Booksim2.0 is prominently used NoC performance analysis tool. Booksim offers network parameters such as routing algorithm, flow control, topology, and router micro-architecture including buffer management, allocation schemes and so on. System operation speed and overall performance of NoC architecture depends on network latency, flit latency and hop count [8].

Large NoC architectures are time insensitive to complete simulation. Hence, there is a need for a faster method to explore NoC architectures. Figure 1 shows the simulation time of Booksim simulator varying from 6 seconds to 10 days for Mesh topology with Uniform traffic pattern and architectural size ranging from 2×2 to 54×54.

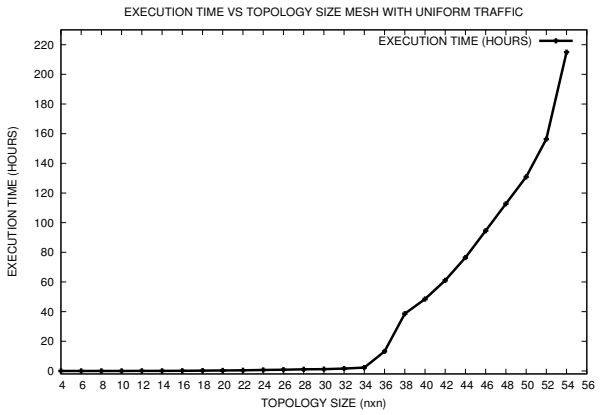


Fig. 1: Booksim simulation time for k-ary 2-dimensional Mesh networks(k=2 to 54)

Simulation time of NoC increases with the increase in architectural size and resources linearly [3]. Experiments were conducted to verify the correlation between performance parameters and size of NoC architecture. It was observed that performance parameters also increases linearly with the increase in size of NoC architecture. The major contribution of our work is proposing a ML framework which can be used to predict performance parameters by training the datasets obtained from Booksim simulator. The framework is designed using widely used ML regression algorithms like SVR and ANN. This ML framework can be used to configure different NoC architectures. Our framework shows a speedup of 1000× to 1500× over the Booksim simulator.

The rest of the paper is organized as follows: Section II describes the related work, Section III specifies Experimental Procedure, Section IV presents the results and Section V concludes this paper.

II. RELATED WORK

We briefly introduce various NoC simulators and also the work done on NoC using ML.

978-1-5386-6835-1/18/$31.00 ©2018 IEEE

A. Network-on-Chip Simulators

To explore the micro-architectural design space of NoCs, researchers have relied on simulators to evaluate the power and performance. Booksim2.0 [3] is a cycle-accurate simulator providing large set of configurable network parameters such as topology, routing algorithm, flow control and router micro-architecture. Noxim[9] NoC simulator is implemented in SystemC. In this several parameters such as network size, buffer size, packet size distribution, routing algorithm etc., can be customized using command line interface. Nirgam[6] is another simulator in SystemC. It helps us in understanding NoC designs by allowing us to experiment on various routing algorithms and topologies. SICOSYS [5] is a general-purpose interconnection network simulator. It helps in modeling message routers using traffic pattern, applied load and message length as input parameters. Orion [10] is a set of architectural power model for On-Chip interconnection routers. It can be used to estimate Network-on-Chip area and power consumption accurately in early design phases.

B. Machine Learning methods used in different aspects of NoC

Qian et al. [11], [12] used SVR model for evaluating Network-on-Chip latency performance. This work differs from state-of-the-art NoC analytical models. Classical queuing theory was used to compute the average channel waiting time and Booksim generated training data was used to analyse NoC latency.

Das et al. [13] proposed a robust design optimization methodology to improve the energy efficiency of 3D NoC architectures. Online ML algorithm was employed combining the benefits of small-world networks and machine learning techniques [14]. The optimized 3D small-world NoC on an average achieved 35% Energy-Delay-Product(EDP) reduction over conventional 3D Mesh.

In NoC performance modeling and analysis, most of the learning models focus on improving the area/power model accuracy. In [15], [16], the multivariate adaptive regression splines (MARS) technique was used to develop a NoC router power and area regression model. Compared to the conventional ORION2.0 [10], the learning-based model improves the prediction accuracy over a variety of NoC implementations. Later in [17], the model accuracy was further enhanced by explicit modeling of control and data paths in regression analysis.

By considering all the work done so far, the work has done on different aspects on NoC using ML. Most of the work was concentrated on Mesh topology with smaller architectural sizes. No much work has been done on other performance parameters and topologies of higher architectural sizes.

III. DESIGN STRATEGY

Figure 2 shows the complete design of the ML framework where each task has been divided into stages.

TABLE I: Booksim Configuration Details

Booksim Network Configuration Details	
Topology Type	Mesh and Torus
Network size	2×2, 3×3, 4×4, ..., 30×30
Traffic Pattern	Uniform, Tornado, Shuffle, Transpose, Bitrev, Bitcomp
Number of Virtual Channels	2, 4, 6, 8, 10
Buffer Size	10
Packet Size	20 flits
Sample Period	100000 cycles
Injection Rate	0.02, 0.025, 0.03, 0.035...0.1
Routing Algorithm	Dimension Order Routing

TABLE II: System Configuration Details

System Configuration for Mesh and Torus	
Processor	Intel Xeon CPU E5-2650 V2
Frequency	2.6Ghz
Memory	64GB

A. Environmental setup of Booksim simulator

All traffic patterns available in Booksim2.0 simulator such as Uniform, Tornado, Bitrev etc. have been considered. Topologies i,e. Mesh and Torus with wormhole flow control along with deterministic XY routing algorithm are being used. Various topology sizes, injection rates and virtual channels were employed as specified in Table I. In all the experiments conducted with Booksim simulator, *sample_period* of 100000 cycles has been employed.

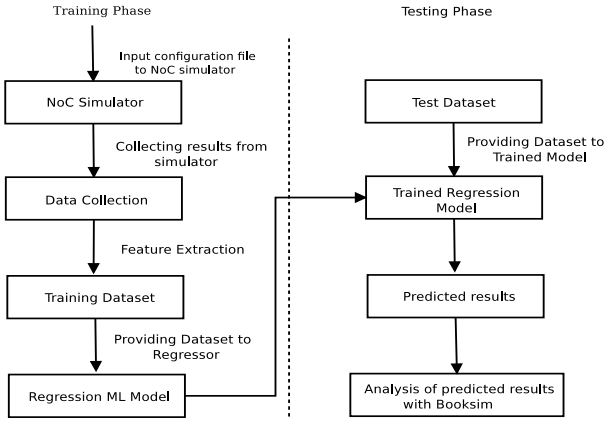


Fig. 2: Outline of Machine Learning framework.

B. Data Collection using Booksim simulator

The system specifications used for data collection is shown in Table II. Booksim simulator was run by varying topology sizes from 2×2 to 30×30 and the output was recorded. Among the results obtained we considered topology size, virtual channel, traffic pattern and injection rates as input features. Average network latency, flit latency and Average hop count were considered to be output features.

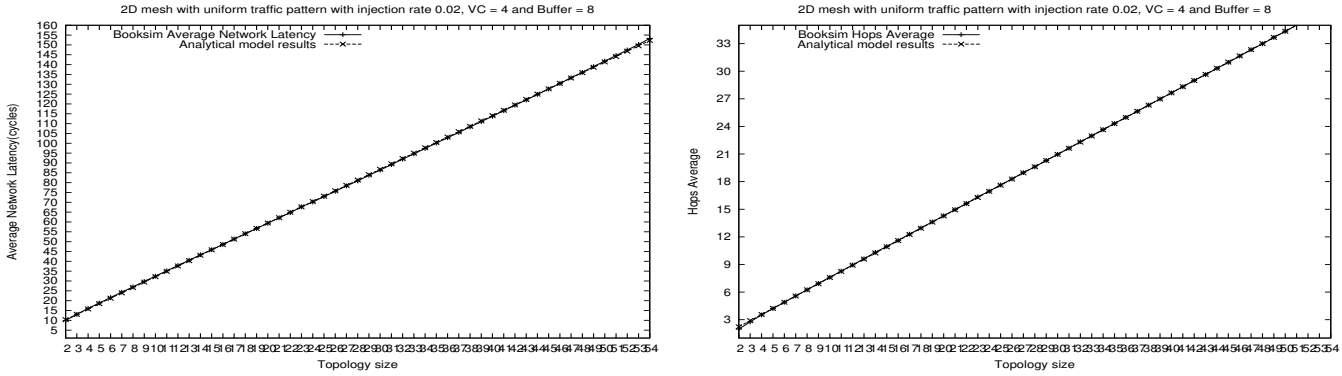
C. Formulation of Dataset

The data is collected separately for each network size, traffic pattern etc., as stated above using Booksim simulator. This

data is divided into training and testing datasets which we further use in ML framework. 50% dataset(2×2 to 15×15) was considered for training and remaining 50% dataset(16×16 to 30×30) was considered for testing purpose.

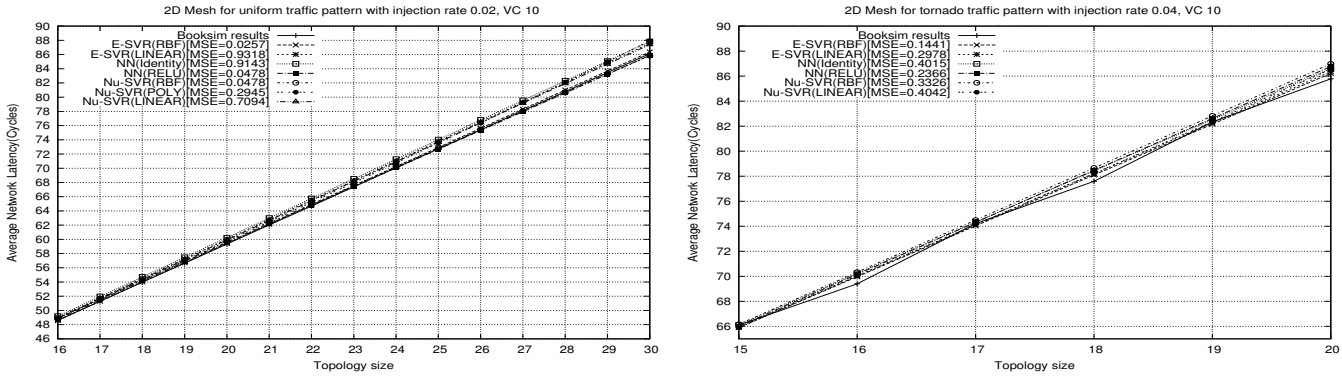
D. Design of Machine Learning Framework

ML framework was designed using widely used regression algorithms like SVR and ANN [18]. SVR has two implementations namely ϵ-SVR and υ-SVR with different kernels. The basic idea of regression is to determine a function that



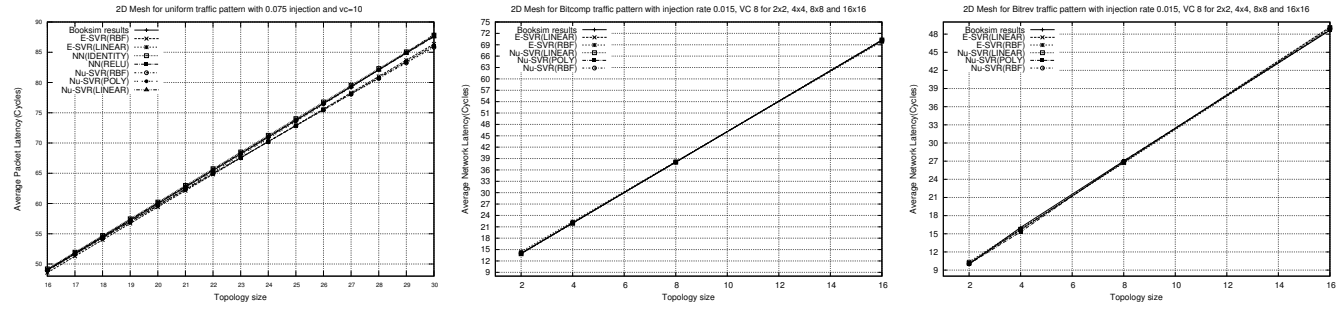
(a) (b)

Fig. 3: Comparison of Performance parameters between Booksim simulator vs Analytical model for Mesh topology, where (a) Average Network Latency, (b) Average Hop count with injection rate 0.02, VC=4



(a) (b)

Fig. 4: Average Network latency comparison of Mesh topology against various ML algorithms for different traffic patterns where (a) Uniform traffic pattern, (b) Tornado traffic pattern with injection rate 0.02,0.04 and VC=10



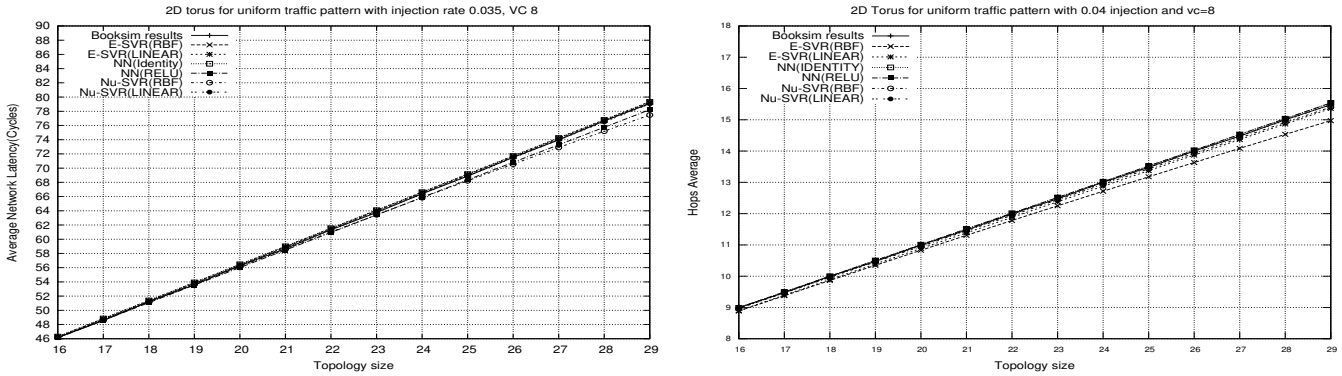
(a) (b) (c)

Fig. 5: Average Packet and Network latency comparison of Mesh topology against various ML algorithms for different traffic patterns where (a) Uniform traffic pattern(packet latency), (b) Bitcomp traffic pattern (c) Bitrev traffic pattern with injection rate 0.075,0.015 and VC=10,8

TABLE III: Mean Square Error of Mesh topology for Uniform traffic pattern and topology size ranging from 16×16 to 30×30 with different injection rates and VC=10

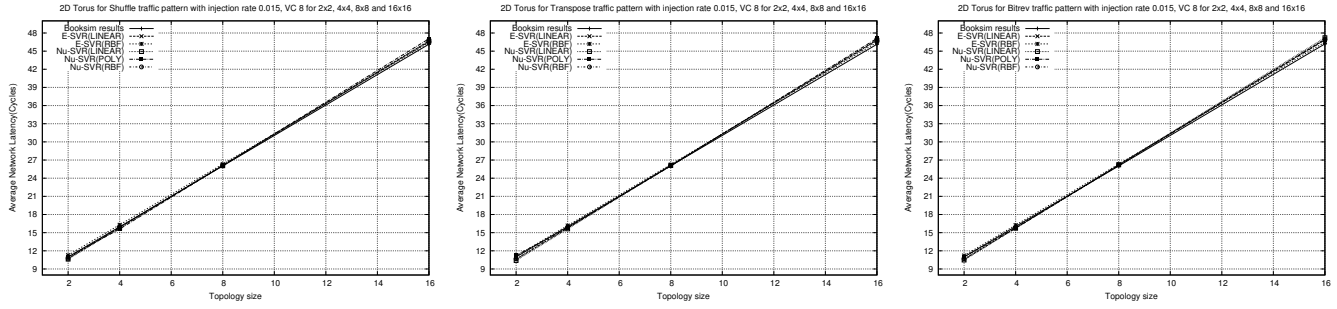
Mean Square Error of Different Machine Learning methods							
injection rates	ϵ***-SVR(linear)***	ϵ***-SVR(rbf)***	υ***-SVR(linear)***	υ***-SVR(poly)***	υ***-SVR(rbf)***	***NN(identity)***	***NN(relu)***
0.02	0.9319	0.0257	0.7093	0.2946	0.0478	0.9143	0.025
0.025	0.8534	0.0131	0.6518	0.2695	0.0497	0.8521	0.0306
0.03	0.7659	0.0047	0.5861	0.244	0.0528	0.7797	0.0375
0.035	0.6829	0.0024	0.5237	0.2234	0.0599	0.7105	0.0467
0.04	0.5984	0.0062	0.459	0.1917	0.0819	0.6375	0.0655
0.045	0.4913	0.0232	0.3741	0.1514	0.1183	0.5392	0.096
0.05	0.3941	0.0526	0.2973	0.1213	0.1645	0.4485	0.1338
0.055	0.297	0.1015	0.2202	0.0896	0.2325	0.3545	0.1876
0.06	0.2054	0.1738	0.1478	0.0614	0.3296	0.2622	0.2652
0.065	0.1197	0.283	0.0809	0.0448	0.4632	0.1708	0.372
0.07	0.0538	0.4327	0.0323	0.042	0.6437	0.0941	0.5182
0.075	0.0137	0.652	0.0086	0.0662	0.899	0.0356	0.7278
0.08	0.02	0.9564	0.0296	0.1325	1.2482	0.0155	1.0195

approximates the target values accurately using a set of input values. SVR is used to find the mapping function between input and output values [19]. The results have been tested with both the methods. In ϵ-SVR, ϵ stands for insensitive loss function which is employed to solve the problem of quadratic optimization. The value of ϵ need to be set prior to the training of SVR model. However it is also hard to forecast the value of ϵ in most problems. To overcome this limitation, υ-SVR [20] is used where, υ indicates a lower bound on the number of support vectors and an upper bound on the fraction of training samples.The kernels used in framework are linear, polynomial and radial basis function which works for both linear and non-linear data. There are other parameters of SVR like *C* which controls the trade-off between the margin and the size of the slack variables and *gamma* which is kernel coefficient for 'rbf', 'poly' and sigmoid. Higher the value of *gamma*, will try to fit



(a) (b)

Fig. 6: Average Network latency and Average hop count comparison of Torus topology against various ML algorithms for Uniform traffic pattern where (a) Packet Latency, (b)Average Hop count with injection rate 0.035,0.04 and VC=8



(a) (b) (c)

Fig. 7: Average Network latency comparison of Torus topology against various ML algorithms for different traffic patterns where (a) Shuffle traffic pattern,(b) Transpose traffic pattern (c) Bitrev traffic pattern with injection rate 0.015 and VC 8

TABLE IV: Mean Square Error of Torus topology for to Uniform traffic pattern and topology size ranging from 15×15 to 29×29 with different injection rates and VC=10

Mean Square Error of Different Machine Learning methods						
injection rates	ϵ-SVR(rbf)	ϵ-SVR(linear)	υ-SVR(rbf)	υ-SVR(linear)	NN(identity)	NN(relu)
0.02	0.0309	0.0194	0.2884	0.0659	0.1121	0.0251
0.025	0.0442	0.0127	0.3136	0.054	0.0987	0.0245
0.03	0.0622	0.0067	0.3527	0.0403	0.0816	0.0286
0.035	0.0873	0.0041	0.4001	0.0285	0.0655	0.0355
0.04	0.1282	0.006	0.4696	0.016	0.0452	0.0473
0.045	0.1868	0.0157	0.567	0.006	0.0238	0.0695
0.05	0.2812	0.0456	0.7139	0.0085	0.0093	0.1145
0.055	0.4149	0.1021	0.9071	0.0312	0.0106	0.1851
0.06	0.6247	0.2112	1.1928	0.0947	0.0442	0.3094

as per the training dataset.

Neural Networks(NN) consists of large number of neurons interconnected in complex ways and often organized into layers. For our framework, we have used Feedforward neural network. In this network, the information moves only in forward direction from the input nodes through the hidden nodes and activation function goes to the output node. We have used two activation functions i,e. Identity and ReLU(Rectified Linear Unit). We have tested datasets with different kernels for SVR and activation functions for NN to check whether datasets works with all above mentioned methods.

The proposed ML framework consists of two phases, namely training phase and testing phase as shown in Figure 2. The framework is fed with the training dataset and the framework is trained accordingly. Later, testing dataset is used in testing phase and outputs are recorded to compare with Booksim simulator in order to calculate error-rate.

IV. RESULTS AND DISCUSSION

A. Experimental results

1) Verification of Analytical Model: The correlation study of performance parameters with NoC architectural size was concluded and the results showed performance parameters increase linearly with the increase in architectural size of NoC. Figure 3 justifies Avg Network latency and Average Hop count obtained from Booksim increases linearly. Similar results can be observed for flit latency. Network latency follows the linear curve $4.8625 + 2.7303 * (architecture\_size)$, hop count follows the linear curve $0.8832 + 0.66929 * (architecture\_size)$ and flit latency follows $5.3495 + 2.7040 * (architecture\_size)$. These results were obtained from mesh topology with injection rate 0.02, virtual channel 4 and buffer size 8.

2) Regression framework for Mesh-based topology: Experiments were conducted for different performance parameters by varying network architecture sizes with all traffic patterns, virtual channels and injection rates for Mesh topology. Figure 4(a),(b) shows comparison of Booksim Avg Network latency against the different ML algorithms for Uniform and Tornado traffic pattern with virtual channel 10, injection rate 0.02, 0.04 and buffer size of 10 respectively. ML framework gave accuracy around 90%-93% with average Mean Square Error(MSE) of 8%-10%. Table III shows the MSE values of various injection rates for Uniform traffic pattern for different NoC architecture sizes. Figure 5(a) shows the comparison Avg Packet latency with different ML algorithms for Uniform traffic pattern with injection rate 0.075 and virtual channel 10. ML framework gave accuracy around 90%-95% with average MSE of 6%-8%. Figure 5(b),(c) shows results for Bitrev and Bitcomp traffic patterns respectively, the experiments were conducted for new virtual channels which was not present in the training dataset. The error rates for these traffic patterns are less than 3%.

3) Regression framework for Torus-based topology: Figure 6(a),(b) shows comparison of Booksim Average Network latency and Average hop count against the different ML algorithms for Uniform traffic pattern with virtual channel 8, injection rate 0.035, 0.04 and buffer size of 10 respectively. ML framework gave accuracy around 90%-95% with average MSE of 8%-10%. Table IV shows the MSE values of various injection rates for Uniform traffic pattern for different NoC architecture sizes. Figure 7 (a),(b),(c) shows results for Shuffle, Transpose and Bitrev traffic patterns respectively, the experiments were conducted for new virtual channels which was not present in the training dataset. The error rates for these traffic patterns are less than 3%.

B. Validation

Testing dataset formulated earlier is used to check the effectiveness of the framework. Validation is performed for different combinations of (ϵ, C and γ) in ϵ-SVR. The values for the insensitive loss function ϵ= 0.1-0.5, γ = 0.001-1 and cost C = 20-25 respectively. Similarly for υ-SVR (υ, γ and C) values are set to 0.5-1, 0.001-1 and 25-50 respectively. 15-100 hidden layers were considered for NN with adaptive learning. Proposed framework was further valuated with other real-time network sizes ranging from 16×16 to 30×30. The results obtained were compared with the actual Booksim results. MSE was calculated for algorithms used. ϵ-SVR and υ-SVR showed MSE of 8% to 10% and NN showed MSE of 5% to 8% for Mesh and Torus topologies.

C. Runtime comparison

We have conducted all the experiments using the system configuration as shown in Table II for Mesh and Torus topologies. Table V shows the timing comparison of Booksim simulator vs other ML algorithms. Total time taken to

TABLE V: Timing comparison of Booksim simulator with ML algorithms(Seconds).

Topology size	$\epsilon - SVR$			$\upsilon - SVR$			NN	
	Booksim	**RBF**	**LINEAR**	**RBF**	**POLY**	**LINEAR**	**Identity**	**ReLU**
16x16	322.124	0.028102	3.10412	0.11677	6.0043	0.11774	0.1834	0.01917
17x17	306.217	0.0632	3.097	0.11223	5.9765	0.1207	0.1437	0.020918
18x18	399.006	0.02864	3.0973	0.1126	5.977	0.121355	0.1421	0.023
19x19	476.108	0.02496	3.128	0.1128	6.0461	0.119945	0.142064	0.0315
20x20	610.485	0.125	3.109	0.1164	5.942	0.120592	0.1427	0.0632
21x21	702.152	0.136	3.324	0.1167	6.348	0.1196	0.26	0.0462
22x22	745.826	0.113	3.653	0.124	5.971	0.1296	0.247	0.0523
23x23	1004.22	0.086	3.782	0.1279	6.317	0.1247	0.293	0.0621
24x24	1342.05	0.0654	3.652	0.1249	6.097	0.11783	0.214	0.125
25x25	877.69	0.0756	3.731	0.1164	5.912	0.1247	0.349	0.62
26x26	2148.47	0.0861	3.964	0.1196	5.932	0.1369	0.56	1.3
27x27	1765.4	1.65	4.124	0.1296	6.176	0.194	1.24	0.946
28x28	2356.31	1.78	4.864	0.1247	6.198	0.3546	0.98	1.36
29x29	2965.52	0.981	3.756	0.1379	6.298	0.48	1.86	1.8
30x30	2556.09	0.994	3.986	0.1396	6.34	0.426	1.35	1.02

complete the execution of 16×16 to 30×30 architecture size is 15624 seconds for Booksim simulator, 8-10 seconds for ϵ-SVR with different kernels, 6-8 seconds for υ-SVR with different kernels and 6-8 seconds for NN. The results show that proposed ML-framework is 1500× to 2000× faster in execution as compared to cycle accurate Booksim simulator.

V. CONCLUSION AND FUTURE WORK

In the proposed ML framework, we evaluated performance parameters of Network-on-Chip architectures. Proposed framework was designed using SVR and ANN algorithms. Different algorithms were employed to test the accuracy of each algorithm. Individual datasets were created for each traffic pattern for network size ranging from 2×2 to 30×30 and these datasets were split into training and testing datasets. The designed ML framework predicted performance parameters with an accuracy of 90%-95% in Mesh and Torus architectures. Proposed framework also showed minimum speedup of 1500× against the Booksim simulator in terms of execution time.

Our future work is to concentrate on power and area of NoC architectures and remaining topologies with Real-time traffic. In addition to this, we aim to extend the work on 3D-NoC architectures.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: a new soc paradigm," *computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*. IEEE, 2001, pp. 684–689.

[3] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 86–96.

[4] V. Catania *et al.*, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *26th IEEE International Conference on Application-specific Systems, Architectures and Processors, ASAP 2015, Toronto, ON, Canada, July 27-29, 2015*, 2015, pp. 162–163.

[5] V. Puente *et al.*, "Sicosys: an integrated framework for studying interconnection network performance in multiprocessor systems," in *Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on*, 2002, pp. 15–22.

[6] L. Jain *et al.*, "Nirgam: a simulator for noc interconnect routing and application modeling," in *Design, Automation and Test in Europe Conference*, 2007, pp. 16–20.

[7] D. Whelihan and H. Schmit, "The nocsim simulator," 2003.

[8] A. E. Kiasari *et al.*, "An analytical latency model for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 113–123, 2013.

[9] Davide Patti. Noxim.

[10] A. B. Kahng *et al.*, "Orion 2.0: A power-area simulator for interconnection networks." Institute of Electrical and Electronics Engineers, 2011.

[11] Z. Qian *et al.*, "Svr-noc: A performance analysis tool for network-on-chips using learning-based support vector regression model," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 354–357.

[12] Z.-L. Qian *et al.*, "A support vector regression (svr)-based latency model for network-on-chip (noc) architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 3, pp. 471–484, 2016.

[13] S. Das *et al.*, "Optimizing 3d noc design for energy efficiency: A machine learning approach," in *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 705–712.

[14] J. Boyan and A. W. Moore, "Learning evaluation functions to improve optimization by local search," *Journal of Machine Learning Research*, vol. 1, no. Nov, pp. 77–112, 2000.

[15] A. B. Kahng *et al.*, "Improved on-chip router analytical power and area modeling," in *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*. IEEE Press, 2010, pp. 241–246.

[16] K. Jeong *et al.*, "Accurate machine-learning-based on-chip router modeling," *IEEE Embedded Systems Letters*, vol. 2, no. 3, pp. 62–66, 2010.

[17] A. B. Kahng *et al.*, "Explicit modeling of control and data for improved noc router estimation," pp. 392–397, 2012.

[18] J. Friedman *et al.*, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[19] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004.

[20] B. Schölkopf *et al.*, "New support vector algorithms," *Neural computation*, vol. 12, no. 5, pp. 1207–1245, 2000.