# Comparative Study and Prediction Modeling of Photonic Ring Network on Chip Architectures

Sara Karimi, Jelena Trajkovic
Electrical and Computer Engineering Department, Concordia University, Montreal, Qc, Canada
E-mail: sar_ka@encs.concordia.ca

## Abstract

Multicore systems are becoming state-of-the-art and therefore need fast and energy efficient interconnects to take full advantage of the computational capabilities. Integration of silicon photonics with traditional electrical interconnect in Network on Chip (NoC) proposes a promising solution for overcoming the scalability issues of electrical interconnect. In this paper, we implement the simulation model for two Optical NoC architectures and compare their performance. We also derive and evaluate a prediction modeling technique for the design space exploration of ONoCs. Our proposed model accurately predicts packet latency, static and dynamic energy consumption of the network. This work specifically addresses the challenge of accurately estimating performance metrics without having to incur high costs of exhaustive simulations. Our case study shows that by using only 10% of the entire design space, our proposed technique builds a prediction model that achieved average error rates as low as 5.44%, 2.67% and 3.24% for network packet latency, static and dynamic energy consumption respectively in six different benchmarks from Splash-2 benchmark suite.

## Keywords

Optical NoC, Prediction Modeling, Design Space Exploration, Simulation, Regression, Machine Learning

## 1. Introduction

By scaling down the technology node, it is expected that hundreds of cores can be accommodated on a single chip [1]. One of the most critical issues in the Chip Multiprocessors (CMP) era will be the communication among different on-chip computational resources. Although NoCs that use conventional RC wires to route data packets on shared channels are a good solution for replacing traditional dedicated buses [2], they cannot scale well with respect to performance and power when the number of cores grows to hundreds or thousands, due to capacitive and inductive effects. Optical Network on Chip (ONoC) is a promising solution for addressing this issue. ONoC demonstrates significant advantages over electrical NoC because of its high bandwidth provided by Wavelength Division Multiplexing (WDM), reduced power consumption due to bit rate transparency and low losses in the optical waveguides that make it distance insensitive [3]. In this paper, we model two electro-optical ring communication networks: ATAC [1] and ORNoC [4]. Our work has the first complete dynamic end-to-end evaluation of Optical Ring Network-on-Chip (ORNoC) [4] using real application workloads. Additionally we compare its performance with ATAC in terms of latency and energy consumption. Our results show that ORNoC improves total energy consumption on an average, by 14.97% compare to ATAC while latency remain the same. We use Graphite [5] to simulate the ONoC architectures and Graphite uses DSENT [6] for delay and energy calculation. DSENT provides models for optical components, the electrical backend circuitry and the interface between electrical and optical components.

Moreover, there are many parameters and design alternatives that can be considered for an ONoC architecture, which create a significantly large design space. Evaluation by extensive simulations require a lot of time which results in increased time-to-market and non-recurring engineering costs. Therefore designers need a method to explore a large design space without incurring high costs of exhaustive simulations. We also propose a prediction modeling technique that overcomes high simulation costs while providing low error rates. For doing this, we sample different design configurations from the entire design space and obtain the simulation results (network latency and energy consumption). While we simulate more than 26 thousand possible design alternatives, we show that only 10% of these samples are required to create our prediction model. Our derived model can be used for predicting the output metrics for the *entire* design space, including the configurations that we *do not* simulate. This work provides a rapid design space exploration opportunity to a designer who wants to capture the tradeoffs between many alternative designs in an ONoC communication architecture. Our two key contributions are:
(1) We implement the ORNoC architecture for the first time and compare it with ATAC. Ref. [4] only calculates the number of wavelengths and waveguides for ORNoC. In our work, we capture the performance and energy consumption of ORNoC using real application workloads.
(2) We present the first prediction modeling technique for the design space exploration of ONoC architectures. This model can accurately predict packet latency, and energy consumption for any design configuration without additional simulations for each of them. For comparing the ORNoC architecture with ATAC and also building our prediction model, we simulate 26640 different design configurations to obtain the network latency and energy consumption. Therefore, we provide the first comprehensive delay and energy consumption data set that is now publicly available in [7] to foster future research on ONoC architectures.

The rest of this paper is structured as follows: Section 2 reviews the previous works, Section 3 gives an overview of the ONoC architectures used. Section 4 introduces our design space exploration technique and further explains our prediction model, while the Sections 5 and 6 present the experimental results and conclusion, respectively.

## 2. Related Work

We implement the ORNoC architecture for the first time and to the best of our knowledge our work in prediction modeling is unique, till date: neither extensive design space exploration nor prediction modeling has been done in the domain of ONoCs. We compare our work to the others in three main categories: (1) ONoC architecture design (2) Simulation and modeling of NoC architectures, (3) Prediction modeling for Design Space Exploration.

### 2.1. ONoC Architecture Design

This section provides a comparison between different ONoC architectures. Ref [3] proposes an approach that uses electrical link for control flow and optical link for data transmission. This type of network suffers from high contention delay due to optical path reservation. Phastlane [8] is a hybrid electrical-optical network that uses optical switches in a mesh like topology. When contention arises, the router makes use of electrical buffers and if necessary, a high speed drop signaling network. λ-router [9] is a point-to-point, all optical, contention-free NoC. The number of wavelengths and switches increase quadratically with the number of nodes in the network for the λ-router, which limits its scalability and increases power consumption and area. Snake [10] is a wavelength-routed ONoC architecture providing point-to-point connections between cores. It uses the photonic switching elements that require waveguide crossings. This increases the optical losses significantly, which in turn increases power consumption and limits the scalability of the entire architecture. CHAMELEON [11] has a reconfiguration layer that can open point-to-point communication channels at run time, to better utilize bandwidth according to the application traffic and to reduce power consumption of the optical network. A separate electrical network manages this reconfiguration process. QuT [12] proposes an optical architecture with a wavelength assignment algorithm based on WDM to reduce the number of wavelengths and microring resonators, but requires an additional optical control network. The need for a control network in QuT and CHAMELEON results in an area and power overhead, and causes performance reduction. Ref [33] proposes a framework to synthesize hybrid photonic NoCs by utilizing Particle Swarm Optimization and Simulated Annealing. HELIX [29] proposes a framework for application specific synthesis of hybrid NoC architectures using graph based algorithm and heuristic method. Both [33] and [29] claim that they are able to explore trade-offs in the design space but use set of the design parameters orthogonal to one considered here. However, the back end of the tools uses ORION [30] for power estimation, which overestimates by 2X and 7X desired metrics compared to DSENT [5] and do not model necessary module needed as interface between electrical and optical part, therefore suffering from the lack of accuracy. Additionally, they both need a reservation process before data transmission phase to avoid any collision at destination node: need for the reservation increases both system complexity and communication delay. Corona [13], Firefly [2] and FlexiShare [14] use an optical crossbar with optical token ring arbitration. The drawback of token-based arbitration is the increase in wait time for receiving a token when the number of nodes increases. Corona uses 64 wavelengths to improve the latency; however, this requires a large number of microring resonators resulting in a large area and high power consumption. To reduce the number of waveguides and wavelengths, the authors of [4] propose the ORNoC architecture, where the same wavelengths can be reused to realize multiple communications on the same waveguide concurrently, with no arbitration. ATAC [1] architecture, also uses ring topology for communication in the optical network. In our case study, we use ATAC and ORNoC architectures. Compared to the above mentioned, due to ring topology, ATAC and ORNoC do not have waveguide crossings, and hence have smaller optical losses. Moreover, they do not need separate control network and arbitration that reduces the overall power consumption and packet latency.

### 2.2. Simulation and Modeling of NoC Architecture

ORION [30] simulator models power and area for design space exploration of an NoC. However, it does not model any optical components, and has incomplete architectural models and timing for the router. PhoenixSim [31] is a photonic NoC simulator that models optical components in an NoC. PhoenixSim lacks electrical models and depends on ORION for modeling all electrical routers and links. We use Graphite [5] for ONoC simulation. Graphite utilizes DSENT [6] for delay and energy calculation. Moreover, DSENT provides models for optical devices, the electrical backend circuitry, and the interface between electrical and optical parts. By using DSENT as a back-end, Graphite has a capability to model delay, area and energy of both optical and electrical components with 20% accuracy compared to SPICE simulation [6]. Therefore, Graphite has advantage over the other simulators, making it the best available solution for DSE. Graphite provides fast and scalable performance. It only has 41X slowdown compare to native execution [5], and by using Lax with Barrier synchronization it can imitates a cycle accurate simulation. More details are given in section 5.

### 2.3. Prediction Modeling for DSE

Traditional DSE efforts can be placed in several categories: speeding up the simulation to simulate all the design space points [15]; raising the abstraction level of the simulated model [16] and increasing the granularity of the simulation [17]; utilizing the emulation [18]; and employing machine learning methods to predict the outputs for the entire design space, based on only a few points. The latter is the scope of our proposed technique. Ref. [19] proposes a linear regression model for predicting the performance of different processor architectures, while [20] proposes regression modeling for predicting performance and power for various applications executing on any microprocessor configuration: both of which require numerically solving and evaluating linear systems to find out an efficient formulation of the linear regression function. Extracting all the nonlinear interactions between a parameter and an output is an inherently difficult task, and it also limits the applicability to only parameters with numeric values. Moreover, for finding the parameters that have a significant impact on the performance and power, these models rely on designer domain knowledge instead of a stepwise procedure, making them impractical for other domains. For predicting the processor performance in CMP systems, [21] uses Neural Network (NN) model, that requires extensive samples, which often suffers from overfitting. ArchRanker [22] formulates the DSE as a ranking problem where it trains a model to predict which of the two microprocessor architectures will perform better, however it does not precisely estimate the performance of that specific configuration. In [23], the authors propose a regression-based, application-specific performance model for design space exploration of GPUs that can predict the program run-time. Due to the differences between GPU and ONoC and their respective design spaces, this approach is unsuitable for our domain. Additionally, all of the previously proposed prediction models target processor core architecture, while we focus on architectural parameters in the communication network design. Please note that processor core, cache, main memory, and implementation technology parameters are not considered in this work, but are part of our future work.

## 3. ONoC Architectures

This section introduces two hybrid electro-optical NoC architectures called ATAC [1] and ORNoC [4]. In both of these architectures that are shown in Figure 1, processing cores are grouped into clusters with a given number of cores. Each cluster has an endpoint called a "hub" which is an interface between optical modules and electrical modules. They have two types of networks that implement a hierarchical electro-optical routing: an Electrical mesh network that connects cores with a point-to-point topology, and an Optical network that connects the hubs via waveguides using a ring topology. During the design phase, the designer chooses between two routing policies: distance-based and cluster-based. If the cluster-based routing is selected, the packets are sent over the *electrical network* if source and destination are in the same cluster, otherwise, the packet is sent over the *optical network*. If the distance-based routing strategy is chosen, there is a parameter called "distance threshold" which is specified by the designer and if the distance between source and destination is less than the distance threshold the packet will be sent over the *electrical network,* otherwise, it will be sent over the *optical network*. Note that not all the cores in a cluster connect directly to the optical hub of the cluster. In each cluster, there are one or more, so-called "access points" i.e. the cores that connect to the optical hub (and therefore to the optical network). If the core is not an access point itself, it sends the packet to the nearest access point in order to send data using the optical network. Orthogonally, each cluster has one or more optical receive networks that connect the hub to the cores in order to receive packets.

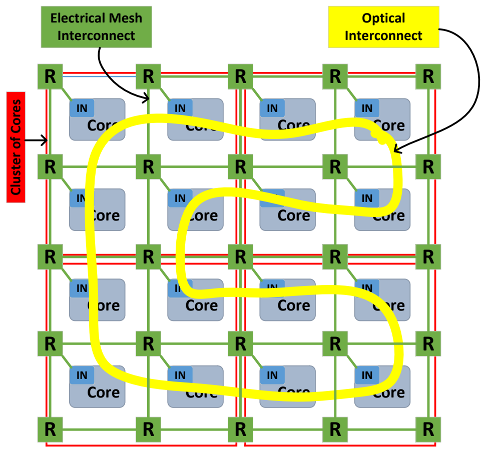

**Figure 1:** Hybrid electro-optical network architecture in ATAC and ORNoC

Although ATAC and ORNoC have the same topology, ORNoC allows the reuse of wavelengths to realize several independent communication channels on a single waveguide, while in ATAC, each connection has a specific wavelength that cannot be reused in that waveguide. In ORNoC, a wavelength-waveguide pair is statically determined based on the destination to enable the reuse of the same wavelengths in the same waveguide which minimizes the power consumption. The algorithm for assigning a wavelength-waveguide pair to each connection in network discussed in [4]. By using this algorithm to reuse wavelengths, ORNoC reduces the number of wavelengths, waveguides and therefore, the number of microring resonators needed in the network. Table I shows the number of wavelengths (#WL), the number of waveguides (#WG) and the number of micro ring resonators (#MR) present in different ONoC architectures when they are connecting a specific number of cores in network (#Core). Reduced number of WLs, WGs and MRs in ORNoC not only decreases power consumption of the network but also leads to a simpler layout and smaller area during fabrication. In section 5.1, we present the simulation results of ORNoC and compare it with that of its main competitor ATAC, using more than 26 thousand simulation samples.

**Table I:** Number of wavelengths, waveguides and micro ring resonators in different ONoC architectures

| Architecture | #Core | #WG | #WL | #MR |
|---|---|---|---|---|
| **FlexiShare** | 64 | NA | 2,464 | 550,000 |
| **Corona** | 64 | 388 | 24,832 | 45,056 |
| | 256 | 388 | 24,832 | 1,056,000 |
| **QuT** | 64 | 8 | 128 | 45,056 |
| | 128 | 16 | 256 | 172,000 |
| **λ-router** | 64 | 32 | 512 | 97,792 |
| | 128 | 64 | 1,024 | 392,192 |
| **ATAC** | 64 | 63 | 4,032 | 8,064 |
| | 128 | 254 | 16,256 | 32,512 |
| **ORNoC** | 64 | 23 | 1,418 | 2,836 |
| | 128 | 91 | 5,762 | 11,524 |

## 4. Procedure for Prediction Modeling in DSE

There are two main steps involved in deriving an accurate model for delay and energy prediction: 1) Generating the data set, and 2) Prediction modeling.

## 4.1. Generating the data set

The first step of our technique is data set generation. Different design configurations of ONoC architecture give different delay and energy consumption. We vary the values of the configuration parameters and simulate the designs to obtain three corresponding outputs: the packet latency and energy consumption of the network (i.e., static and dynamic energy). Table II identifies seven design features (design parameters): $f_1, \ldots, f_7$, and their corresponding set of values. An important step for obtaining a representative model that can predict the outputs with high accuracy is feature selection. We should select features with significant impact on outputs. Moreover including the features that have little impact on the outputs can result in overfitting that may mislead the prediction model. For selecting a relevant feature set, initially we exclude the conservative and aggressive scenario of the network parameters (like distance greater than 16 hops in the distance-based routing strategy), optical technology parameters (like ring tuning strategy, type of receiving network and type of laser) and technology node. Next, we change all the other configurable parameters in the network and then, by using the Correlation based Feature Selection (CFS) algorithm we select a subset of them that has the biggest impact on delay and energy consumption of the network. The CFS algorithm evaluates the worth of a subset of features by considering the individual predictive ability of each feature along with a degree of redundancy between them [24]. We evaluate 30 subsets of features, and then select the best subset with the highest merit (= 0.564) based on Equation 1:

$$M_S = \frac{k\overline{r_{fo}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \qquad (1)$$

Where $M_s$ is merit of a feature subset $S$ containing $k$ features, $\overline{r_{fo}}$ is the mean feature-output correlation ($f \epsilon S$), and $\overline{r_{ff}}$ is the average feature-feature inter-correlation [24]. Therefore, we select a subset of features that has high correlation with our desired outputs. Also these selected features can be generalized to the other ONoC architectures and are not only limited to ATAC and ORNoC architectures. In our data set, the feature "distance" indicates the distance threshold for the distance-based routing strategy. Therefore, "distance" does not have any meaning for the "cluster-based" routing strategy and has a missing value. For all the configurations with cluster-based routing, the value of "distance" is set to the maximum distance between any two cores in a cluster. We also convert all the numeric values to nominal values. The Cartesian Product (CP) of the sets of selected seven features $\prod_{i=1}^{7} f_i$ defines the entire design space. However we should consider the specific limitation for configuring our network which is:

$$cluster\ size \geq \#receive\ networks \ \geq \#access\ points$$

And also $f_6$ is null when routing strategy is set to cluster based. Based on the Table II, the cardinality of the CP is equal to 16384 but due to the mentioned limitations, the number of feasible design points becomes 4440. Since the latency and energy consumption of the network are dependent on the running application we use six benchmarks from Splash-2 benchmark suite [25] to build and test our prediction model. Therefore we add these benchmarks as our 8th feature.

**Table II:** Changing parameters in simulations

| | Design Parameter | Range | $\|f_i\|$ |
|---|---|---|---|
| $f_1$ | Number of Cores | 64,256 | 2 |
| $f_2$ | Cluster size | 1,2,4,8,16, 32,64,128 | 8 |
| $f_3$ | #Optical access point | 1,2,4,8,16, 32,64,128 | 8 |
| $f_4$ | #Receive Network | 1,2,4,8,16, 32,64,128 | 8 |
| $f_5$ | Routing strategy | Distance/Cluster-based | 2 |
| $f_6$ | Distance in distance-based routing | 2,4,8,16 | 4 |
| $f_7$ | Architecture | ATAC, ORNoC | 2 |
| | **Application Parameter** | **Name** | |
| $f_8$ | Splash-2 Benchmark | lu, ocean, radix, water, cholesky, barnes | 6 |

**Table III:** Fixed parameters in simulations

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Technology node | 45 nm | Cache line size | 64 Bytes |
| Temperature | 300 K | L1 Cache size | 16 KB |
| Tile width | 1 mm | L2 Cache size | 512 KB |
| Received Network | StarNet | L1 Associativity | 4 |
| Ring Tuning | Athermal | L2 Associativity | 8 |
| Flit width | 64 | Replacement Policy | LRU |
| Laser efficiency | 0.30 | Coupler loss | 2 dB |
| Waveguide loss | 0.2 dB | Ring drop loss | 1 dB |
| Ring through loss | 0.01 dB | Modulator loss | 0.01 dB |
| Photodetector Capacitance | 5 fF | Ring heating efficiency | 100K/mW |
| Optical link data rate | 2Gb/s | Link bit error rate | $10^{-15}$ |

In this way we only need to build one model to predict the outputs for all the benchmarks, instead of creating individual prediction models for each benchmark. By considering $f_8$, the total number of simulations raises to 26640. The parameters that do not change during the simulation for all the configurations are shown in Table III. In this paper we only focus on modeling the communication network. Modeling of processors and memory can be done in the same manner, but it is out of scope of this paper. Also, manufacturing and thermal variability, and hence reliability, although are very important, are out of scope of this paper, and topics of our future work.

## 4.2. Prediction Modeling

The previous step generates a data set ready to be used for formulating a model to make a prediction. We employ several models for predicting the outputs, and they can be divided into three separate categories: 1) Regression Models, 2) Tree Models, 3) Neural network Models. Regression Tree gives the most accurate predictions for the given data set and it uses them for generating experimental results (Section 5.2). The result and detailed comparison of all prediction models can be found in [32]. We describe this model in more detail below.

The regression tree algorithm works with the "Standard Deviation Reduction" technique [26]. It nominates one feature for splitting the tree and splits the data set if the standard deviation of the output is reduced by this splitting. Once no more splitting is possible, a leaf is reached representing a "decision" or a predicted value of the output. Algorithm 1 shows the pseudo-code for building a regression tree. In the first step of this algorithm, we calculate the standard deviation of the outputs for the entire data set as per Equation 2, where

**Algorithm 1** Regression Tree Algorithm

```
Input: feature[number of feature]
Input: dataset[number of feature][number of samples]
StDev_output = standard_deviation(y)
while StDev_output >threshold do
    for index = 1, ..., number of feature do
        max_StDev_reduction = 0
        split(dataset, feature[index])
        StDev_branch = standard_deviation(y)
        StDev_reduction = StDev_output − StDev_branch
        if StDev_reduction >max_StDev_branch then
            max_StDev_reduction = StDev_reduction
            decision_node = feature[index]
        end
    end
    split(dataset, decision_node)
    StDev_output = standard_deviation(y)
end
```

$y$ is the output value, $\mu$ is the mean of the output values and $n$ is the number of outputs.

$$S = \sqrt{\frac{\sum(y-\mu)^2}{n}} \qquad (2)$$

In the second step, we calculate the standard deviation of the output for each feature. $S_{split}(output, feature)$ is the value of the standard deviation for the case in which we split the data set according to the values of a single feature. It is calculated as per Equation 3, where $P(i)$ is the probability that the feature has a value $i$ and $S(i)$ is the standard deviation of the output when the feature has the value of $i$.

$$S_{split}(output, feature) = \sum_{i \in feature} P(i)S(i) \quad (3)$$

Constructing a regression tree is all about finding a feature that returns the highest Standard Deviation Reduction (SDR) for the output. To obtain the SDR we use Equation 4, where $S_{split}$ is the standard deviation of the output after splitting the data based on that particular feature and $S$ is the original standard deviation of output before any splitting.

$$SDR(output, feature) = S(output) - S_{split}(output, feature) \quad (4)$$

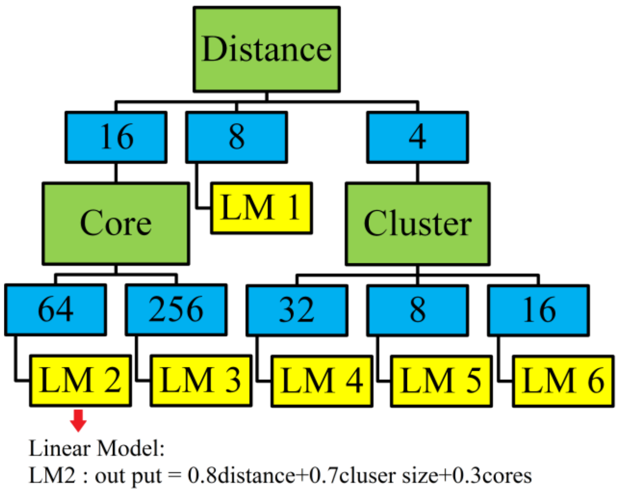

**Figure 2:** Regression Tree with linear models in leaves

In the third step, we calculate the SDR for each feature. The feature that has the largest SDR will be chosen for splitting the tree. We apply the same procedure to each node and all of its branches. This procedure is repeated on the non-leaf branches until all data is processed. The leaf nodes contain a linear regression model for the data subset that corresponds to the leaf i.e, following the path from the root to the leaf and assigning the feature value that corresponds to each split. Figure 2 shows the structure of Regression Tree for a small example.

## 5. Experimental Results

For obtaining the simulation results we use Graphite simulator. Graphite is not a cycle accurate simulator but supports several synchronization strategies that have a trade-off between timing accuracy and simulation performance. For having best timing accuracy, we use Lax with Barrier synchronization where all active threads wait on a barrier until it reaches a specified number of cycles (we use 1000 cycles). Frequently waiting on the barrier keeps the cores tightly synchronized, and imitates a cycle accurate simulation. Although cycle-accurate simulators provide extremely accurate results, they typically have 1000x to 100,000x worse simulation time [5]. We make use of the six applications listed in Table II from the Splash-2 benchmark suite [25], as a standard for comparing the performance of parallel systems. These applications use shared memory and Pthread library. Our technique can be generalized to any other parallel application. We use Weka [26] for statistical analysis, data preprocessing, and building the prediction model.

### 5.1. ORNoC vs ATAC

In this section we present the ORNoC simulation results and compare them with the ATAC simulation results. We simulate 26640 different configurations to compare network latency (in nanoseconds), static and dynamic energy consumption (in Joules) for six different benchmarks. Figures 3 and 4 show the average values of each output for *all the possible configurations* for each benchmark in ORNoC and ATAC. As Figures 3 and 4 show, ORNoC has 14.40%, 15.34%, 14.47%, 17.07%, 7.71% and 20.88% improvement in total energy consumption compared to the ATAC architecture in Radix, Ocean, Barnes, Lu, Water and Cholesky applications respectively. This improvement in energy consumption comes without any expense in network latency: ORNoC shows 0.68%, 0.41%, 0.29%, 0.26%, 0.77% and 2.73% improvement in network latency compared to the ATAC architecture for Radix, Ocean, Barnes, Lu, Water and Cholesky applications respectively. Moreover as Figure 5 demonstrates, ORNoC shows 13.31%, 15.74%, 14.32%, 16.85%, 14.21% and 17.80% improvement in Energy-Delay product compare to ATAC architecture in Radix, Ocean, Barnes, Lu, Water and Cholesky applications respectively.

### 5.2. Accuracy of Prediction Models

We evaluated three types of prediction models: Regression models, Neural Network models and Regression Tree models. The best prediction model is shown to be Regression Tree model [32]. Therefore, we only present the result of Regression Tree models here, but more results with other prediction models can be found in [32].

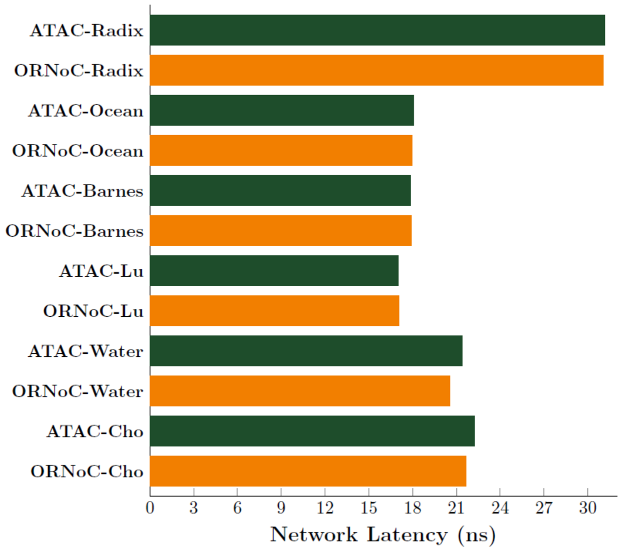

**Figure 3:** Packet Latency for ATAC and ORNoC

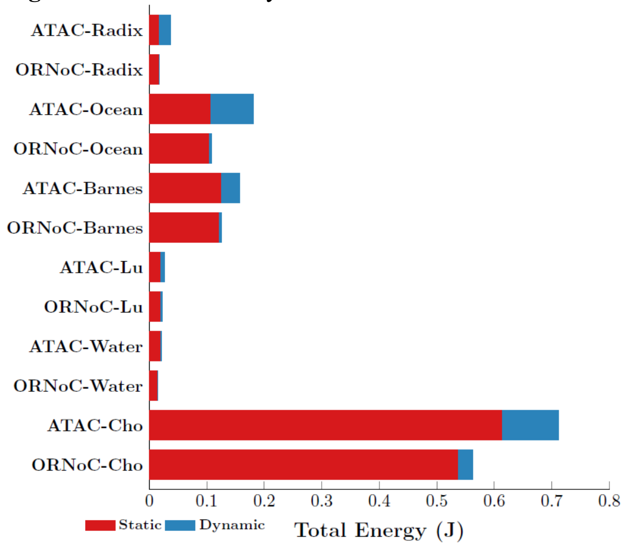

**Figure 4:** Total Energy for ATAC and ORNoC

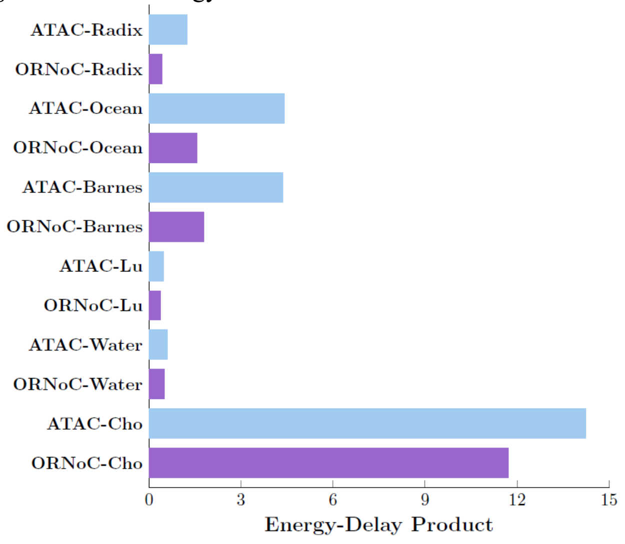

**Figure 5:** Average Energy-Delay Product for ATAC and ORNoC

For evaluating our prediction models, we use "Ten Fold Cross Validation" method [26] and we report the Root Relative Squared Error (RRSE) [32] of predicted vs simulated values. We used two different regression tree models: M5P tree [27], and REP tree [28]. The average RRSE for packet latency, static and dynamic energy of M5P tree is 7.01% while that of REP tree is 3.78%, among all the benchmarks. M5P tree has a structure as explained in section 4.2 with a linear model function at each leaf. As Table IV indicates, REP tree shows better accuracy than M5P tree. It is because REP tree uses the Reduce Error Pruning technique, which makes a better prediction. The idea of pruning is to select a subtree or a branch, and replace it with a single leaf if the replacement reduces the prediction error. Table V shows the RRSE of REP tree model for each benchmark separately.

**Table IV:** RRSE of Regression Tree model for all benchmarks

| Algorithm | Packet Latency | Static Energy | Dynamic Energy |
|---|---|---|---|
| M5P Tree | 6.48% | 8.70% | 5.86% |
| REP Tree | 5.44% | 2.67% | 3.24% |

**Table V:** RRSE of REP Tree model for each benchmark

| Benchmark | Packet Latency | Static Energy | Dynamic Energy |
|---|---|---|---|
| Barnes | 6.46% | 2.23% | 2.14% |
| Cholesky | 8.48% | 2.22% | 1.73% |
| Lu | 4.24% | 2.16% | 3.20% |
| Ocean | 7.45% | 5.67% | 6.11% |
| Radix | 3.69% | 2.73% | 1.41% |
| Water | 9.10% | 2.58% | 3.72% |

## 5.3. Prediction Accuracy for Different Sample size

During the data set generation, we gathered 26640 simulation results to cover different ONoC architectures. It is not always necessary to gather such enormous data set to predict the desired output with acceptable accuracy. We sample a small subset of configurations from the original data set and build our prediction model based on this smaller subset instead of the original large data set. We test subsets with different sizes (i.e. different sample sizes) to formulate prediction models, and compare the accuracy of the models with increase in the size of the subset. Figure 6 shows the prediction error (RRSE) as a function of the subset size used for training that predicts the "static energy". We uniformly and randomly select 10 to 90 percent of the original data set for training. As the Figure 6 shows, by increasing the number of samples for training, prediction error decreases. Moreover, by having only 10% of original data set we can have a prediction model which has the RRSE rate as low as 8.59%. However, by further increasing the size of the dataset, the value of RRSE does not significantly decrease, and only goes down to 2.67% when the training set size is 90%. We show that it is possible to formulate a prediction model with an acceptable error rate from a significantly smaller data set. Therefore, it is possible to further reduce the number of simulations, and solve the fundamental challenge of high simulation costs for the traditional design space exploration.

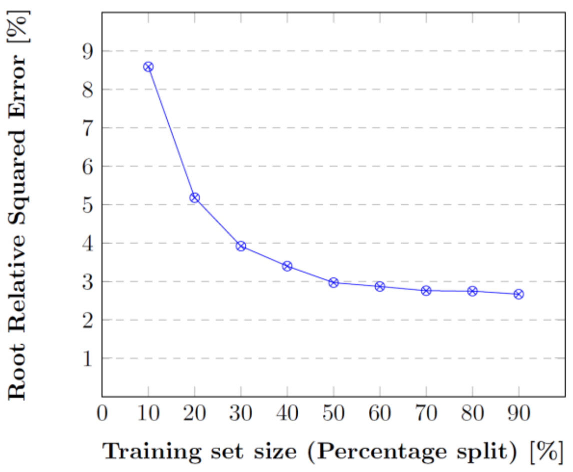

**Figure 6:** RRSE of static energy in REP Tree model as a function of training set size for all six benchmarks

We also visualize the errors for "average packet latency". Each dot in Figure 7 indicates one configuration and its actual value obtained by simulation versus its predicted value using the REP tree model. It can be seen that the dots lie along the line y = x (plotted in red) meaning that the predicted values are very close to the simulated values.

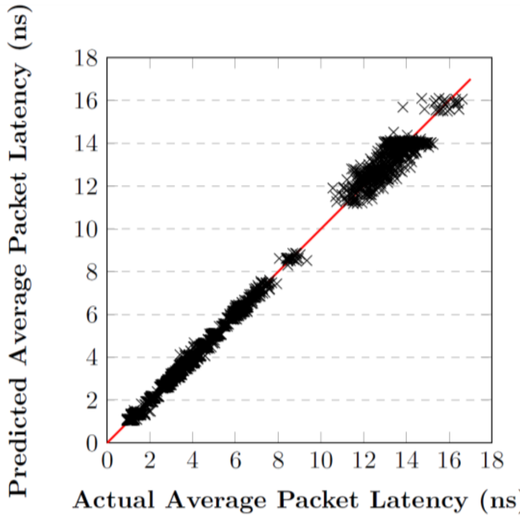

**Figure 7:** Actual (simulated) value vs predicted value with REP Tree in Barnes Application

Figure 8 shows distribution of prediction error. It represents the frequency of error values that occur in the prediction. The absolute difference between predicted value and the actual value, i.e. absolute error, is grouped into bins. The bin that belongs to the smallest absolute error is at the center and as it spread to left and right the absolute error values increase. As the Figure 8 shows the most frequent absolute error values for average packet latency are in the range [-0.0468, 0.0324]. It shows that in most of the cases the prediction value is very close to the actual value, as depicted by the central column in the Figure 8.

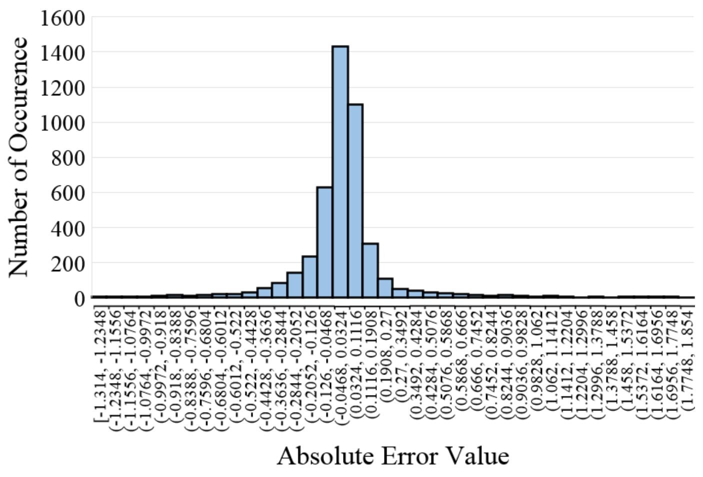

**Figure 8:** Distribution of Absolute Error values for average packet latency in Ocean Application

## 6. Conclusion and Future Work

This paper presented a comparative evaluation of ORNoC architecture using more than 26 thousand simulations, we showed that on an average, the ORNoC architecture improves total energy consumption by 14.97% compared to ATAC without any overhead in the network latency. We also presented a prediction modeling technique for design space exploration of the optical network on chip. The proposed prediction model addresses the fundamental challenge of accurately estimating the desired metrics without having to incur high simulation costs. As shown in our case study, the root relative squared error rate is as *low* as 5.44%, 2.67% and 3.24% for network packet latency, static and dynamic energy consumption, respectively. Additionally, we showed that only 10% of design space points are required for building our prediction model. We can extend our work to include other ONoC architectures, and also to include features for the memory and the core to obtain a prediction model for not only the network architecture but also for the overall system.

## Acknowledgment

The authors would like to thank Evan Livingstone-White and Amine Mhedbhi for their early work on implementation of ORNoC model.

## References

[1] G. Kurian et al., "Cross-layer energy and performance evaluation of a nanophotonic manycore processor system using real application workloads," IEEE 26th International Parallel and Distributed Processing Symposium, pp. 1117–1130, 2012.

[2] Y. Pan et al., "Firefly: Illuminating future network-on-chip with nanophotonics," ISCA, pp. 429–440, 2009.

[3] A. Shacham, et al., "Photonic networks-on-chip for future generations of chip multiprocessors," IEEE Trans, vol. 57, pp. 1246–1260, 2008.

[4] S. Le Beux et al., "Optical Ring Network-on-Chip (ORNoC): Architecture and design methodology," DATE, pp. 1–6, 2011.

[5] J. E. Miller et al., "Graphite: A distributed parallel simulator for multicores," HPCA, pp. 1–12, 2010.

[6] C. Sun et al., "DSENT - A tool connecting emerging photonics with electronics for opto-electronic networks-on-

chip modeling," 6th IEEE/ACM International Symposium on NoCS, pp. 201–210, 2012.

[7] https://github.com/OpticalNoC/DataSet

[8] M. J. Cianchetti et al., "Phastlane: A rapid transit optical routing network," ISCA '09, pp. 441–450, ACM, 2009.

[9] I. O'Connor et al., "Reduction methods for adapting optical network on chip topologies to specific routing applications," Design of Circuits and Integrated Systems, 2008.

[10] L. Ramini et al., "Contrasting wavelength-routed optical noc topologies for power-efficient 3d-stacked multicore processors using physical-layer analysis," DATE, pp. 1589–1594, 2013.

[11] S. Le Beux et al., "Chameleon: Channel efficient Optical Network-on-Chip," DATE, pp. 1–6, 2014.

[12] P. K. Hamedani et al., "QuT : A Low-Power Optical Network-on-Chip," 8th IEEE/ACM International Symposium on NoCS, pp. 80–87, 2014.

[13] D. Vantrease et al., "Corona: System implications of emerging nanophotonic technology," ISCA, pp. 153–164, 2008.

[14] Y. Pan, et al., "Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar," in HPCA, pp. 1–12, Jan 2010.

[15] E. K. Ardestani et al., "ESESC : A Fast Multicore Simulator Using Time-Based Sampling," in 19th HPCA, pp. 448–459, 2013.

[16] I. Uddin et al., "Analytical-based high-level simulation of the microthreaded many-core architectures," in 22nd Euromicro, pp. 344–351, 2014.

[17] D. Sanchez et al., "ZSim : Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems," in 40th ISCA, pp. 475–486, 2013.

[18] N. Genko et al., "A Complete Network-On-Chip Emulation Framework," in DATE, pp. 246–251, 2005.

[19] P. J. Kapil et al., "Construction and Use of Linear Regression Models for Processor Performance Analysis," HPCA, pp. 99–108, 2006.

[20] B. C. Lee et al., "Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction," 12th ASPLOS, no. 1, pp. 185–194, 2006.

[21] E. Ipek et al., "Efficient Architectural Design Space Exploration via Predictive Modeling," ACM Transactions on Architecture and Code Optimization, vol. 4, no. 1, 2008.

[22] T. Chen et al., "ArchRanker : A Ranking Approach to Design Space Exploration," 41st ISCA, pp. 85–96, 2014.

[23] W. Jia et al., "Stargazer: Automated regression-based GPU design space exploration," IEEE International Symposium on Performance Analysis of Systems and Software, pp. 2–13, 2012.

[24] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in 17th International Conference on Machine Learning, (San Francisco, CA, USA), pp. 359–366, Morgan Kaufmann Publishers Inc., 2000.

[25] S. C. Woo et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations," ISCA, pp. 24–36, 1995.

[26] I. H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 3rd ed., 2011.

[27] J. R. Quinlan, "Learning with continuous classes," pp. 343–348, World Scientific, 1992.

[28] L. Breiman and J. Friedman, Classification and Regression Trees. Wadsworth, Belmont, CA, USA: Chapman & Hall, 1984.

[29] S. Bahirat et al., "HELIX: Design and synthesis of hybrid nanophotonic application-specific network-on-chip architectures," in *Fifteenth International Symposium on Quality Electronic Design*, Santa Clara, CA, 2014, pp. 91-98.

[30] A. B. Kahng et al., "ORION 2.0: A Power-Area Simulator for Interconnection Networks," IEEE Transactions on VLSI Systems, vol. 20, no. 1, pp. 191–196, 2012.

[31] J. Chan et al., "Phoenixsim: A simulator for physical-layer analysis of chip-scale photonic interconnection networks," DATE, pp. 691–696, 2010.

[32] S. Karimi, "Prediction Modeling and Design Space Exploration in Optical Network on Chip", (Montreal, Canada), Concordia University, 2016.

[33] S. Bahirat et al., "A Particle Swarm Optimization approach for synthesizing application-specific hybrid photonic networks-on-chip," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, March 2012, pp. 78-83.