# COMRANCE: A Rapid Method for Network-on-Chip Design Space Exploration

Mingzhe Zhang*†, Yangguang Shi*†, Fa Zhang‡ and Zhiyong Liu*
*Beijing Key Laboratory of Mobile Computing and Pervasive Device, ICT, CAS, Beijing, China
†University of Chinese Academy of Sciences, Beijing, China
‡Key Lab of Intelligent Information Processing, ICT, CAS, Beijing, China
{zhangmingzhe, shiyangguang, zhangfa, zyliu}@ict.ac.cn

***Abstract*—As the communication sub-system that connecting various on-chip components, Network-on-Chip (NoC) has a great influence on the performance of multi-/many-core processors. Because of NoC model contains a large number of parameters, the design space exploration (DSE) for NoC is a critical problem for the architects. Similar to the core design, existing DSE process mainly depends on iteratively time-consuming simulations. To lower the time budget, many previous studies focus on reducing the simulations. However, most of the proposed works based on regression or machine learning techniques, whose accuracy will be significantly affected by the scale of training set. It still needs a lot of simulations to build the training set.**

**Previous work of network analysis has shown that, the upper limit of network latency can be estimated at the network diameter and congestion. Based on this perspective, we propose a novel approach to DSE of NoC, called COMRANCE, which provides rapid searching for the configuration with the best performance. We experimental show that, the method can provide exploration for a variety of parameters for detail designing while allowing the architect to balance the exploration granularity and time budget. The method also supports the comparison of area and power consumption among various configurations, and it is convenient to incorporate with expert knowledge in the method.**

**We implement the framework of COMRANCE and evaluate it with different configurations and benchmarks. The experiment shows that, on average, COMRANCE achieves the reliability at $90.91\%$ with benchmark driven and $92.32\%$ with synthetic traffic driven. Besides this, we analyze the factors which affects the reliability of COMRANCE, such as injection rate and NoC scale. We also find that the major factor that affects the reliability of COMRANCE varies as the injection rate changes. Finally, we show that the DSE time cost for COMRANCE linearly increased as the scale of NoC grows.**

## I. Introduction

As the chip scale moving forward to multi-core or even many-core, the design of Network-on-Chip(NoC) gains increasingly attentions. Since NoC contains large numbers of parameters, determining the best performing configuration from the large design space is a critical problem. Currently, the design space exploration (DSE) for NoC mainly depends on iterative simulations, which cost large time budget.

To speed up the DSE, many prediction-based techniques have been proposed. Although these works focus on whole chip designing, they can be probably shifted to NoC. However, the prediction approach requires large training set, which is constituted by a series of simulated samples. Since the scale of training set affects the accuracy of prediction, it is still very time-consuming to achieve a well-trained prediction model.

Fortunately, previous work on network analysis provides a new perspective. Leighton *et al.* proved that the upper limit of latency in any packet routing network can be estimated from the congestion of the routing path and the diameter of network [25]. According to the conclusions, we argue that, when taking maximum latency as the measurement, the NoC configurations can be compared by calculating their diameters and congestions.

Based on this observation, we propose a framework, called COMRANCE (Congestion Matrix based Rapid NoC Evaluation), which provides fast DSE for NoC without simulation. We show that our framework can provide the relativity of NoC configurations with high accuracy while saving a large amount of time. Moreover, COMRANCE can provide rapid design space exploration with constraints on circuit area and power consumption.

We compare the results from COMRANCE and simulation with the configurations random sample from a vast design space. The experiment result shows that COMRANCE achieves the reliability at $90.91\%$ and $92.32\%$ on average respectively when compared with benchmark-driven and synthetic-traffic-driven simulation. We analyze the affection factor to the reliability of COMRANCE and find that the reliability increases (from $84.31\%$ to $93.75\%$ on average) as the injection rate grows (from $0.10$ to $0.55$) while the reliability decreases (from $93.65\%$ to $91.33\%$ on average) as the NoC scales grows (from $16-core$ to $256-core$). Besides this, we find the *dilation* and *congestion* are the major factor to the reliability of COMRANCE respectively with different injection-rate levels. Furthermore, we show that although the *congestion matrix* grows rapidly as the scale of NoC becoming larger, the DSE time cost of COMRANCE does not increase dramatically, due to *constrain check module* filters much of the configurations.

In summary, the main contributions of our work are as follows. First, we propose a novel technique to compare the NoC configurations and introduce a framework called COMRANCE. Second, we show that the NoC components with micro-architecture can be transformed to sub-network, so that much more parameters can be incorporated into the framework. This also allows the architects balance the

978-1-5090-5117-5/16/$31.00 

time cost and exploration granularity. Third, we implement the COMRANCE framework and use several configurations, which are randomly sampling from the vast design space, with different benchmark applications to show the accuracy of our framework.

The rest of the paper will be organized as follows. Section II presents the background of our work. Section III describes the framework of COMRANCE in detail. Section IV introduces the methodology of our work and Section V presents the experiment results. We briefly review the related work in Section VI and draw some conclusions and future works in Section VII.

## II. Background

As a common type of networks, packet routing network plays an important role in various systems, which scales from multicore SoCs to Internet. In general, packet routing network moves a series of packets from one node to another, focusing on finishing the transmission with best effort. Since there are various choices of routing, *store-and-forward routing* is the most popular method, which stores the arrived packet in the router until the switching components get ready for forwarding it.

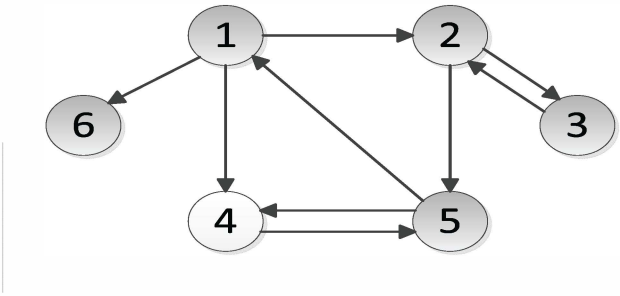



Fig. 1. A sample network with $dilation = 4$ and $congestion = 13$

A typical packet routing network with a store-and-forward method is as shown in Figure 1. The nodes labeled from 1 to 6 represent the routers with switching components and buffer, and the edges between the nodes represent to wires. At each step, a packet can traverse at most one edge. The buffers are organized as FIFO queues. When a packet is about to traverse an edge, the *receive queue* should be checked firstly. Once there is enough space, the packet will be moved from sender to receiver. One packet will leave the network when it arrives its destination. Note that, here we adopt a typical assumption that a packet will never be forwarded along the same link more than once, which implies that the routing paths are acyclic.

To analyze the network, two key concepts have been identified as follows. The first one is called *dilation d*, which refers to the length of the longest acyclic path traversed by any packet of the network. The second one is called *congestion* $c$, which indicates the largest number of packets that have to traverse any edge in the network. It is easy to see that the time needed to deliver the packets to their destinations will be at least $\min\{c, d\}$.

Furthermore, given any network $G$ with congestion $c$ and dilation $d$, we can simply accomplish the transmission in at most $c \cdot d$ steps, since any routing path will contain at most $d$ edges and any packet will wait in the buffer of an edge for at most $c$ steps.

Interestingly, Leighton *et al.* [25] proved that, given any network $G$ with congestion $c$ and dilation $d$, there always exists a schedule that can deliver the packets in $O(c+d)$ steps. Here *schedule* refers to the strategy of determining the transmission priority of the packets. Their result becomes constructive in [26]. This result enables us to directly estimating the performance of a configuration by estimating the worst-case congestion and dilation.

## III. Congestion Matrix Based Exploration Model

### A. Matrix Based NoC Comparison Method

In this section, we introduce COMRANCE from a system perspective. A typical NoC with mesh topology is as shown in Figure 2(a), in which we assume that all links are bi-directional and each direction transmits one packet per cycle. The routing algorithm is *static x-y*, which routes the packet in x- and y-dimension orderly. Let $\vec{L}$ denote all links in a NoC, and each item $l \in L$ denotes a specific link in NoC. Let $\vec{R}$ denote all possible communication requirements, and each item $r \in R$ denotes a specific communication requirement in the network. We then define $C = \vec{L}^T \times \vec{R}$ as the *congestion matrix* of the given NoC (as shown in Figure 2(b)). The $(l, r) - th$ element in $C$ will be assigned according to following rules:

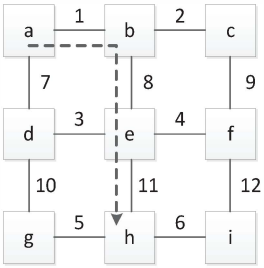
$$C(l,r) = \begin{cases} 1, & \textit{if link } l \textit{ is used for communication } r \\ 0, & \textit{if link } l \textit{ is not used for communication } r. \end{cases}$$

Considering the circuit area and power consumption limitation, most of the proposed NoCs uses oblivious routing algorithms [28], which makes each packet has a deterministic path in the network. For such routing strategy, every element in the corresponding *congestion matrix* can be determined directly. For instance, in Figure 2(a), when *node a* transmits a packet to *node h*, the packet passes *link 1, 8 and 11* orderly. Accordingly, the column corresponds to this communication is filled as shown in Figure 2(b) (in dashed box).

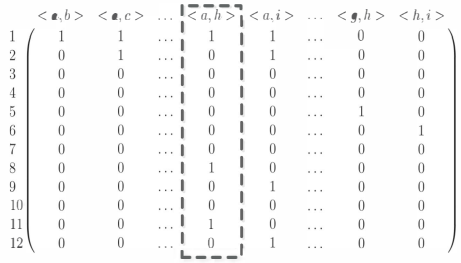
As mentioned in Section II, the upper bound on the packet transmission delay can be estimated with the network dilation and congestion. When using static routing algorithms, the dilation only depends on the network topology, therefore, we will only focus on the upper bound of congestion. In particular, we will analyze the congestion incurred by the worst-case traffic, since related literature [5], [1] indicates that a worst-case traffic model can properly reflect the dynamic behaviors of packet switching networks.

Let $\vec{x}$ denote the communication requests vector, and $\overrightarrow{load} = C \cdot \vec{x}$ denotes the load vector in the network. Then, we could get the congestion according to the definition of Lp-norm:

$$congestion = \left\| \overrightarrow{load} \right\|_\infty = \left\| C \cdot \vec{x} \right\|_\infty . \quad (1)$$




(a) Sample NoC with mesh topology




(b) Congestion matrix

Fig. 2. Sample of translating NoC to congestion matrix

In the worst-case, the upper bound on *congestion* can be calculated as

$$\begin{aligned} c_{max} &= \frac{\|C \cdot \vec{x}\|_\infty}{\|\vec{x}\|_\infty} = \|C\|_\infty \\ &= \max\left\{\sum_{i=1}^{r}|c_{1i}|, \sum_{i=1}^{r}|c_{2i}| \cdots, \sum_{i=1}^{r}|c_{li}|\right\} \end{aligned} \tag{2}$$

where the $\|C\|_\infty$ denotes the *induced* infinite norm of matrix $C$ and $\sum_{i=1}^{r}|c_{li}|$ denotes the cumulative result of $line\ l$.

So far, we could compare two given NoC configurations from $(dilation + congestion)$.

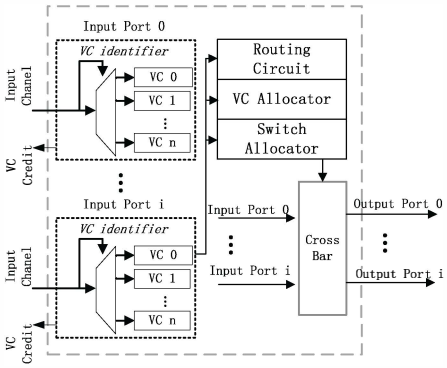
### B. Router Micro Architecture Parameters

Based on the method proposed in Section III-A, the performance of topology and routing algorithm can be estimated and compared. However, there are much more parameters about router micro-architecture need to be considered. In this section, we will show how to extend the method for these parameters.

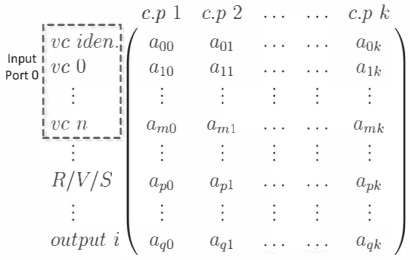
A typical NoC router owns the structure as shown in Figure 3(a). In each port, there are n-way FIFO buffers, which called *Virtual Channels (VCs)*. Before a packet is sent, the *VC-identifier* in the receiver port will check the usage of local VCs, and accepts the packet only when an available VC is found. In each cycle, the locally stored packets have a chance to be selected by the *routing circuit*. The *routing circuit* will calculate the destination of next hop according to a certain algorithm, while the VC-allocator will check for routing conditions. Once the routing conditions are met, switch allocator will configure the crossbar and the packet will be sent to the next router.

Based on the previous descriptions, we use a network to describe the structure of router, each component will be represented as a node. Then we construct vector $\overrightarrow{L_{sub}}$ with the nodes in the network and build a new congestion matrix $C_{sub}$ similar with matrix $C$ (as shown in Figure 3(b)). When a component is used by a communication request, the corresponding item $C_{sub}\,(l, r)$ will be assigned to 1. Here we assume that the VC is allocated with Round-in-Robin policy. Furthermore, since the port owns p-size buffer in each VC, in a router which allows VC-reuse for different packets, the accumulative result of a line which corresponds to a VC should minus $p$. Because the congestion only happens when the buffer is full.

Finally, we insert $\overrightarrow{L_{sub}}$ and $C_{sub}$ to $\vec{L}$ and $C$ respectively to get $\overrightarrow{L^*}$ and $C^*$.




(a) Typical i-port NoC router architecture




(b) Congestion matrix of NoC router

Fig. 3. Sample of building congestion matrix for NoC router

### C. Circuit Area and Power Consumption

In this section, we will describe the method for area budget and power consumption estimation. As proposed in Section III-B, the expanded vector $\overrightarrow{L^*}$ contains all considered components. Let vector $\overrightarrow{Area}$ denote the set of each component's circuit area, and the total area should be

$$TotalArea = \overrightarrow{Area} \cdot \overrightarrow{L^*}. \tag{3}$$

The power consumption of NoC includes *static power consumption* and *dynamical power consumption*. For *static power*, it is related to the circuit design of each component and manufacturing technology, which can be treated as a series of constants and be estimated as

$$StaticPower = \overrightarrow{E_s} \cdot \overrightarrow{L^*} \tag{4}$$

where vector $\overrightarrow{E_s}$ denote the set of each component's static power consumption.

Furthermore, because the *dynamical power consumption* of NoC relates to the traffic pattern, let $e_d \in \overrightarrow{E_d}$ denote the power

that the component consumed for achieving its function, and the *dynamical power consumption* should be

$$\begin{aligned} DynamicalPower &= \overrightarrow{E_d} \cdot \overrightarrow{load} = \overrightarrow{E_d} \cdot (C \cdot \vec{x}) \\ &= \left(\overrightarrow{E_d} \cdot C\right) \cdot \vec{x} = \overrightarrow{C^*} \cdot \vec{x} \end{aligned} \tag{5}$$

In worst-case, we could assign that $\|\vec{x}\| = 1$, and based on the definition of 1-norm, the maximum dynamical power consumption will be

$$\begin{aligned} DynamicalPower_{max} &= \left\|\overrightarrow{C^*} \cdot \vec{x}\right\|_{max} = \left\|\overrightarrow{C^*}\right\|_{max} \\ &= \max\left\{\sum_{i=1}^{l} |c_{i1}^*|, \sum_{i=1}^{l} |c_{i2}^*| \cdots, \sum_{i=1}^{l} |c_{ir}^*|\right\} \end{aligned} \tag{6}$$

where the $\sum_{i=1}^{l} |c_{ir}^*|$ represents the accumulative result of $column\ r$. Overall, the power consumption of NoC can be estimated as

$$\begin{aligned} TotalPower = &(StaticPower + DynamicalPower) \\ &\times frequency. \end{aligned} \tag{7}$$

Note that, the item values in vector $\overrightarrow{Area}$, $\overrightarrow{E_s}$ and $\overrightarrow{E_d}$ are calculated by specific model, such as ORION [20] and DSENT [29], which can be updated as the manufacture technology moves forward.

### D. Exploration Algorithm

Based on proposed method, we implement a framework, called COMRANCE. Besides searching for the possible configuration with the best performance, COMRANCE also supports the exploration with constraints, such as circuit area and power consumption. COMRANCE uses *local search* technique to find target configurations. This allows COMRANCE to partially update the matrix between adjacent steps, which reduce the time significantly. The searching process is shown as Algorithm 1, in which $D(s)$ indicates the result of *dilation+congestion* and $H(l, s)$ indicates the level that the configuration $s$ satisfies the constraint $l$. In each step, an adjacent configuration will be selected for evaluation. Once there are more than one choices, all configurations will be evaluated orderly. To accelerate COMRANCE, when a new configuration is selected from the design space, it will be checked according to the given constraints at first, and then configuration meet all constraints will be built corresponding matrix.

**Algorithm 1** Matrix Based DSE with Local Search Technique

**Input:**
$\mathbb{S}$: design space;
$\mathbb{L}$: constraint set with descend order by importance;
**Output:**
Exploration result: the best performed configuration which satisfies the constraints;

```
Randomly choose s ∈ 𝕊, s_next = φ;
while s ≠ s_next do
    s ← s_next
    if s satisfies every constraint in 𝕃 then
        for each s_i ∈ εs do
            s_next ← min{D(s), D(s_i)};
        end for
    else
        for each s_i ∈ ε_s do
            s_next ← max{G(l_h, s), G(l_h, s_i)};
        end for
    end if
end while
return s;
```

## IV. Methodology

### A. Simulator and Benchmark

To evaluate COMARANCE, we build a simulation platform for fast data collection based on BookSim 2.0 [16], which is a widely used open source simulator for cycle-accurate NoC simulation, supporting a wide variety of reconfigurable parameters. We take average latency and power consumption as the main measurements of performance.

We use 8 synthetic traffic patterns (including *uniform, bitcomp, bitrev, shuffle, transpose, tornado, neighbor* and *randperm* ) and 9 benchmarks to drive the NoC simulator. The detail of benchmark is shown in Table I. When using benchmark driven, we choose the standard trace files from Netrace [13], which are generated from a 64-core CMP system running PARSEC 2.1 benchmark suite [4]. Since each trace file contains messages last for billions of cycles and the injection rate varies dramatically, we pick up a set of traces (10M cycles in each set) from each trace file which provides the maximum injection rate.

TABLE I
PARALLEL PROGRAMS FROM PARSEC 2.1

| *Program* | *Input Size* | *Packet Number* |
|---|---|---|
| blackscholes | simlarge | 132438 |
| bodytrack | simlarge | 453678 |
| canneal | simmedium | 543914 |
| dedup | simmedium | 551801 |
| ferrest | simmedium | 301611 |
| fluidanimate | simlarge | 687724 |
| swaptions | simlarge | 304779 |
| vips | simmedium | 437259 |
| x264 | simmedium | 97922 |

### B. Design Space

Our experiment based on a 64-node NoC with $65nm$ manufacture technology and $1GHz$ working frequency. The NoC design space includes 7 common parameters of NoC, each parameter can be one of several selections (as shown in Table II). The descriptions of some parameters are as shown in Table III. Totally, the whole design space includes $90,116$ configurations.

### C. Experiment

To evaluate the reliability of COMRANCE, we compare the exploration result from COMRANCE against simulating results. In each experiment, unless otherwise specified, we randomly choose 1000 configurations from the design space mentioned in Section IV-B, and each selection will be simulated with benchmark and synthetic traffic driven. Accordingly,

TABLE II
NETWORK-ON-CHIP DESIGN SPACE

| Parameters | Values | Number |
|---|---|---|
| **Topology** | | |
| fly | $(k, n)$ | $(2,6), (4,3), (8,2)$ | 3 |
| | Routing Algorithm | $dest\_tag$ | 1 |
| mesh | $(k, n)$ | $(8,2)$ | 1 |
| | Routing Algorithm | $dim\_order,\ dim\_order\_ni,\ dim\_order\_pni$ | 3 |
| torus | $(k, n)$ | $(64,1),\ (8,2)$ | 2 |
| | Routing Algorithm | $dim\_order\_bal,\ dim\_order\_ni,\ dim\_order$ | 3 |
| cmesh | $(c, x, y, xr, yr)$ | $(4,4,4,2,2),\ (16,2,2,4,4)$ | 2 |
| | Routing Algorithm | $dor,\ dor\_no\_express$ | 2 |
| flattened butterfly[22] | $(c, k, n, x, y, xr, yr)$ | $(1,4,2,4,4,1,1),\ (4,4,1,2,2,2,2)$ | 2 |
| | Routing Algorithm | $ran\_min,\ ugal,\ ugal\_pni$ | 3 |
| **Flow Control** | | | |
| number of VC | | 8∼128: 8+ | 16 |
| VC buffer size | | 1∼32: 1+ | 32 |
| VC allocation algorithm | | Round-in-Robin | 1 |
| **Router Micro-architecture** | | | |
| input speedup | | 1∼4: 1+ | 4 |
| output speedup | | 1∼4: 1+ | 4 |
| **Total** | | | 180,232 |

TABLE III
DESCRIPTIONS OF PARAMETERS IN TABLE II

| *Parameter* | *Description* |
|---|---|
| $k$ | The number of routers per dimension |
| $n$ | The number of network dimensions |
| $c$ | The number of nodes which are sharing a single router |
| $x$ | The number of routers in X dimension |
| $y$ | The number of routers in Y dimension |
| $xr$ | The number of nodes in the X direction per router |
| $yr$ | The number of nodes in the Y direction per router |

all configurations will be evaluated with COMRANCE. We repeat the experiment for 10 times to lower sampling bias.

Then we set a model as follows for the description of reliability. We firstly divide the configurations into several samples, each sample contains two configurations, the total number of samples will be $C_{1000}^2 = 499,500$. For a certain sample $P(a, b)$, we use $\chi(x)$ representing the comparison result of $a$ and $b$, where $1$ indicates $a$ providing better performance than $b$ and $-1$ for otherwise. So the reliability can be quantified as:

$$Reliability = \frac{1}{n}\sum_{i=1}^{n} H(P_i) \tag{8}$$

where $H(x)$ is defined as:

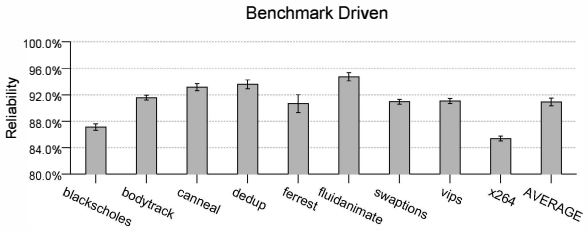
$$H(P) = \begin{cases} 0, & \frac{\chi_s(P)}{\chi_m(P)} < 0 \\ 1, & \frac{\chi_s(P)}{\chi_m(P)} > 0 \end{cases} \tag{9}$$

The function $\chi_s(P)$ and $\chi_m(P)$ indicate the comparisons based on simulation result and COMRANCE respectively.

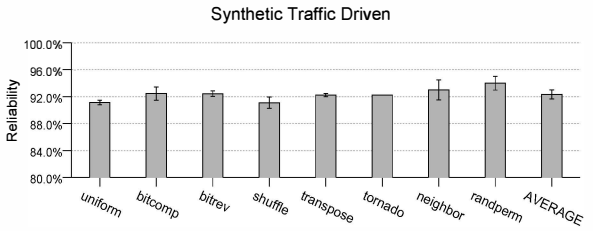
## V. EXPERIMENT RESULTS

### A. Reliability

The reliability is the key character for a DSE method. To prove the reliability of COMRANCE, we compare COMRANCE to simulator using the model described in Section IV-C. The comparison results are as shown in Figure 4. From Figure 4(a), we find that, compared with the simulation results with benchmark driven, the reliability of COMRANCE is 90.91% on average and varies from 85.38% in *x264* to 94.73% in *fluidanimate*. From Figure 4(b), we find that, compared with the simulation results with synthetic traffic driven, the reliability of COMRANCE is 92.32% on average and varies from 91.09% in *shuffle* to 93.99% in *randperm*. The results show that the reliability of COMRANCE is related to both injection rate and traffic pattern, due to these two factors affects the possibility of traffic congestion in NoC.




(a) DSE Reliability for Benchmark Driven




(b) DSE Reliability for Synthetic Traffic Driven (Injection Rate=0.45)

Fig. 4. Reliability of COMRANCE

### B. Sensitivity to Injection Rate

As we described in Section III, COMRANCE uses the *dilation* of network and *congestion* in worst-case to predict the comparison result of NoC configurations. For a certain configuration, higher injection rate usually means closer to the worst-case, which leads to better prediction reliability in COMRANCE. Otherwise, when NoC working with low injection rate, most of the circuit resources for heavy workload

may rarely be used, which may lead to high static power consumption and longer transmission latency. This makes the reliability of COMRANCE decreases. As shown in Figure 5, for a certain configuration, when injection rate raises from 0.10 to 0.55, the reliability various from 84.31% to 93.75% on average. The raising speed is affected by different traffic pattern. Figure 4 also shows that, COMRANCE gets higher reliability with synthetic traffic than with benchmarks, because the current parallel applications can not provide very high injection rate. This character makes COMRANCE suitable in designing NoC for the unknown heavy workload.

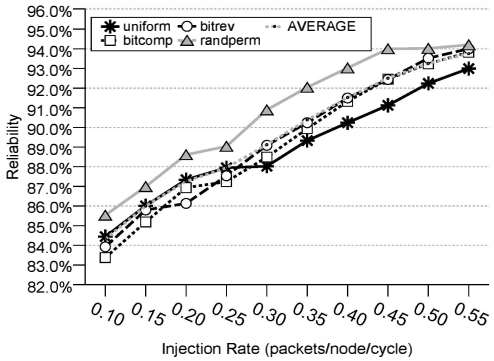



Fig. 5. Reliability comparison when injection rate changes

### C. Sensitivity to NoC Scale

In this section, we analyze the relationship between reliability of COMRANCE and scale of NoC. As shown in Figure 6, with the same injection rate, COMRANCE achieves higher reliability in 16-core NoC than in 64-core and 256-core NoCs. This is because in small NoC, the average transmission distance becomes short, and more packets can reach the same node simultaneously, which makes it much closer to the worst-case communication. Otherwise, when the scale of NoC grows, the packets have less chance to influent each other. For instance, the reliability of COMRANCE various from 93.65% to 91.33% on average when the scale of NoC grows from $16-core$ to $256-core$. An interesting phenomenon is that, with the *neighbor* traffic pattern, COMRANCE gets similar reliability in 16-core, 64-core, and 256-core. This phenomenon indicates that the traffic pattern will affect the level of congestion.

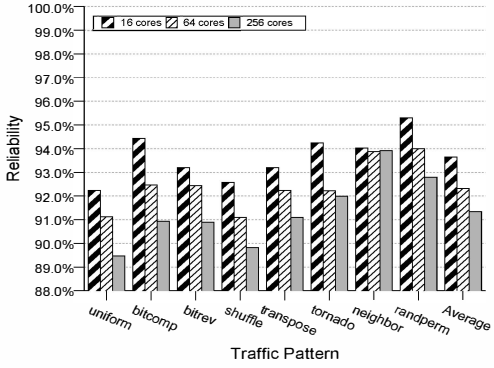



Fig. 6. Reliability comparison when NoC scales changes (Topology=Mesh, Injection Rate=0.45 packets/cycle/node)

### D. Dilation, Congestion and Injection Rate

In this section, we try to predict the comparison results of configurations with *dilation* and *congestion* respectively. As shown in Figure 7, for a 64-core NoC configuration, when the injection rate increases from 0.10 to 0.55, the reliability based on *dilation* decreases from 90.20% to 63.19% while the reliability based on *congestion* increases from 52.23% to 76.91%. The similar trend appears in the experiment based on 256-core NoC. We consider that when injection rate is low, the congestions seldom happen, the transmission latency mainly relies on the distance that a packet passes, while the ability to solve congestion gradually becomes the major factor when injection rate raises.

The other interesting phenomenon shown in Figure 7 is, with the same injection rate, the reliability based on *dilation* performs better in 256-core NoC than in 64-core, while the reliability based on *congestion* performs better in 64-core than in 256-core. We consider that when the injection rate stays at a low level, larger NoC further reduces the possibility of congestion appearance, which strengthens the affection of *dilation*. Similarly, when injection rate raises, smaller NoC has a higher possibility of congestion, which makes the reliability based on *congestion* performs better in smaller NoC.

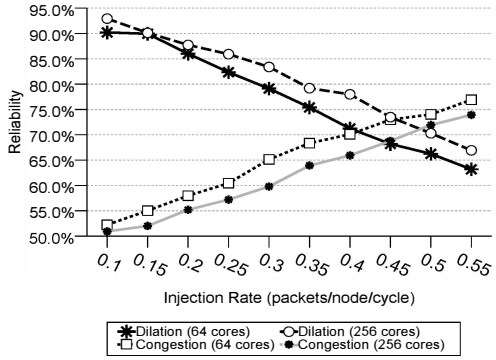



Fig. 7. Reliability comparison when using *dilation* and *congestion* respectively (Topology=Mesh, Injection Rate=0.45)

## VI. Related Work

### A. Prediction Based DSE Method

Considering the time cost of simulation, directly evaluation without simulation becomes very attractive, several excellent works focused on this area have been presented. In order to build a predictive model, regression-based techniques are the most common choices [17], [23], [24]. However, regression models depend on large training sets and may probably have a limitation in predicting the non-linear response behaviors [17]. Besides, the artificial neural network (ANN) techniques are also common choices[18], [14], [15] due to its good support for non-linear response behaviors and requiring smaller training sets. Furthermore, to support unknown applications, ANN combination technique[9] and expanded training set[21] are also presented to improve this approach.

Besides the above works, other machine learning techniques are also introduced to DSE. For instance, Support Vector

Machine (SVM) technique for cross-platform compiler performance prediction[10], Kernel Canonical Correlation Analysis (KCCA) technique for training set reduction[30], and semi-supervised learning algorithm for training acceleration[12]. Especially, Chen *et al.* formulated the DSE process as a ranking problem and provided a learning-to-rank technique based model called ArchRanker [6]. This model predicts the comparison result of two configurations, and significantly reduces the scale of the training set while provides high accuracy ANN-based models.

The above works can provide high accuracy prediction, however, to achieve high prediction accuracy, preparing for the training set which contains hundreds or even thousands of simulation results still can be very time-consuming. Furthermore, the models need re-training when new parameters are involved. Our work using a compare-and-eliminate model, which can be calculated directly and rapidly. Based on this change, the DSE process can be finished in considerable time even facing architectures with dramatic changes.

### B. Analytical Model Based for DES

The other approach to reducing the DSE time budget is to use analytical models. Analytical models formulate target systems as mathematic models and evaluate the configuration without simulation. Palermo *et al.* [27] presented an analytical framework for embedded SoC design space exploration. Based on the framework, they also compared a series of heuristic algorithms to find the approximate Pareto-optimal configure sets. Ascia *et al.* employed a multi-objective evolutionary algorithm to reduce the number of experimental parameter sets in application specific SoC parameter optimization, while accelerating the simulation based on Fuzzy System technique [2]. Genbrugge *et al.* [11] proposed a statistical simulation technique which focused on modeling memory address behaviors more architecture-independently and the time-varying executions of programs. Bakhouya *et al.* proposed a network calculus based analytical model for NoC performance evaluation [3]. Chou *et al.* introduced a design methodology for on-chip interconnection in heterogeneous embedded SoCs, which considering the user experience and energy saving at the same time [7]. Cohen *et al.* [8] presented a statistical NoC analysis method, called traffic-load distribution plots (T-Plots), to illustrate the usage of capacity over-provision for a different level of Quality-of-Service guarantee. Jung *et al.* proposed a theoretical framework based on queuing network models to describe multi-core processors at the thread level, which can evaluate general performance properties over large design space rapidly [19].

Analytical modeling depends less on simulation and provides evaluation result rapidly. However, it requires human interventions and can be too complex to integrate new parameters. In contrast, our work proposes a simple method to generate an analytical model and add new parameters.

## VII. Conclusion and Future work

In this paper, we present a novel framework for DSE of NoC, called COMRANCE. Differ from the existed prediction approach, our method searches the best-performed configuration by comparing the upper limit of network latency, while avoiding the time-consuming preparation for trained model. The experiment results show that the reliability of COMRANCE reaches $90.91\%$ and $92.32\%$ respectively compared with benchmark driven and synthetic traffic driven simulation. Then we analyze the influences of injection rate and NoC scale to the reliability of COMRANCE. Additionally, we analyze the principle relationship among *dilation*, *congestion* and reliability of COMRANCE, and show that the major factor of the reliability gradually changes from *dilation* to *congestion* as the injection rate raises.

Our future work will focus on the following three parts. First, we will try to improve the accuracy of the matrix-based comparison method. Second, since COMRANCE is currently used for exploring the best-performed configuration, we plan to expand its scope to DSE for other parts of the chip. Third, new searching algorithms, especially parallel algorithms, will be integrated into the framework to accelerate the exploration.

## Acknowledgment

This work is supported partially by the National Natural Science Foundation of China (NSFC) Major International Collaboration Project 61520106005.

## References

[1] M. Andrews, A. Fernández, A. Goel, and L. Zhang. Source routing and scheduling in packet networks. *Journal of the ACM (JACM)*, 52(4):582–601, 2005.

[2] G. Ascia, V. Catania, A. G. Di Nuovo, M. Palesi, and D. Patti. Efficient design space exploration for application specific systems-on-a-chip. *Journal of Systems Architecture*, 53(10):733–750, 2007.

[3] M. Bakhouya, S. Suboh, J. Gaber, and T. El-Ghazawi. Analytical modeling and evaluation of on-chip interconnects using network calculus. In *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, pages 74–79. IEEE, 2009.

[4] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, October 2008.

[5] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *Journal of the ACM (JACM)*, 48(1):13–38, 2001.

[6] T. Chen, Q. Guo, K. Tang, O. Temam, Z. Xu, Z.-H. Zhou, and Y. Chen. Archranker: A ranking approach to design space exploration. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 85–96, June 2014.

[7] C.-L. Chou and R. Marculescu. Designing heterogeneous embedded network-on-chip platforms with users in mind. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(9):1301–1314, 2010.

[8] I. Cohen, O. Rottenstreich, and I. Keslassy. Statistical approach to networks-on-chip. *Computers, IEEE Transactions on*, 59(6):748–761, 2010.

[9] C. Dubach, T. Jones, and M. O'Boyle. Microarchitectural design space exploration using an architecture-centric approach. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 262–271. IEEE Computer Society, 2007.

[10] C. Dubach, T. M. Jones, and M. F. O'Boyle. Exploring and predicting the architecture/optimising compiler co-design space. In *Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*, pages 31–40. ACM, 2008.

[11] D. Genbrugge and L. Eeckhout. Chip multiprocessor design space exploration through statistical simulation. *Computers, IEEE Transactions on*, 58(12):1668–1681, 2009.

[12] Q. Guo, T. Chen, Y. Chen, Z.-H. Zhou, W. Hu, and Z. Xu. Effective and efficient microprocessor design space exploration using unlabeled design configurations. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1671, 2011.

[13] J. Hestness, B. Grot, and S. W. Keckler. Netrace: dependency-driven trace-based network-on-chip simulation. In *Proceedings of the Third International Workshop on Network on Chip Architectures*, pages 31–36. ACM, 2010.

[14] E. İpek, S. A. McKee, R. Caruana, B. R. de Supinski, and M. Schulz. Efficiently exploring architectural design spaces via predictive modeling. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, pages 195–206, New York, NY, USA, 2006. ACM.

[15] E. İpek, S. A. McKee, K. Singh, R. Caruana, B. R. d. Supinski, and M. Schulz. Efficient architectural design space exploration via predictive modeling. *ACM Transactions on Architecture and Code Optimization (TACO)*, 4(4):1, 2008.

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

[17] P. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pages 99–108. IEEE, 2006.

[18] P. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. A predictive performance model for superscalar processors. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 161–170. IEEE Computer Society, 2006.

[19] H. Jung, M. Ju, and H. Che. A theoretical framework for design space exploration of manycore processors. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, pages 117–125. IEEE, 2011.

[20] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: a fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the conference on Design, Automation and Test in Europe*, pages 423–428. European Design and Automation Association, 2009.

[21] S. Khan, P. Xekalakis, J. Cavazos, and M. Cintra. Using predictivemodeling for cross-program design space exploration in multicore systems. In *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*, pages 327–338. IEEE Computer Society, 2007.

[22] J. Kim, W. J. Dally, and D. Abts. Flattened butterfly: A cost-efficient topology for high-radix networks. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, pages 126–137, New York, NY, USA, 2007. ACM.

[23] B. C. Lee and D. M. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *ACM SIGPLAN Notices*, volume 41, pages 185–194. ACM, 2006.

[24] B. C. Lee, J. Collins, H. Wang, and D. Brooks. Cpr: Composable performance regression for scalable multiprocessor models. In *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*, pages 270–281. IEEE, 2008.

[25] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling ino (congestion+ dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[26] T. Leighton, B. Maggs, and A. W. Richa. Fast algorithms for finding o (congestion+ dilation) packet routing schedules. *Combinatorica*, 19(3):375–401, 1999.

[27] G. Palermo, C. Silvano, and V. Zaccaria. Multi-objective design space exploration of embedded systems. *Journal of Embedded Computing*, 1(3):305–316, 2005.

[28] E. Salminen, A. Kulmala, and T. D. Hamalainen. Survey of network-on-chip proposals. *white paper, OCP-IP*, pages 1–13, 2008.

[29] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic. Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 201–210. IEEE, 2012.

[30] C. Zhang, A. Ravindran, K. Datta, A. Mukherjee, and B. Joshi. A machine learning approach to modeling power and performance of chip multiprocessors. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 45–50. IEEE, 2011.