A Generic Traffic Model for On-Chip Interconnection Networks

Jun Ho Bahn
Qualcomm CDMA Technologies
Qualcomm Inc., 5775 Morehouse Drive
San Diego, CA 92121-1714
Email : jbahn@qualcomm.com

Nader Bagherzadeh
EECS, University of California, Irvine
325 Engineering Tower
Irvine, CA 92697-2625
Email : nader@uci.edu

***Abstract*— On-chip interconnection networks or Network-on-Chips (NoCs) are becoming the *de-facto* scaling communication techniques in Multi-Processor System-on-Chip (MPSoC) or Chip Multiprocessor (CMP) environment. However, the current traffic models for on-chip interconnection networks are insufficient to capture the traffic characteristics as well as evaluate the network performance. As the technology scaling enables the increase of available on-chip resources and innumerable new network architectures are proposed, there is a need to make NoCs more application-specific. Therefore, a traffic model to characterize such an application-specific network is necessary. In this paper, we propose a generic traffic model for on-chip interconnection networks. Our traffic model is based on three empirically-derived statistical characteristics using temporal and spatial distributions. With captured parameters, our model can generate accurate traffic patterns recursively to show similar statistical characteristics of the observed on-chip networks. Therefore, using the proposed traffic model defined by captured statistics, any kind of on-chip interconnection traffic patterns can be reproduced.**

I. Introduction

As the number of integrated IP cores in the current System-on-Chips (SoCs) keeps increasing to meet the design requirements for computation-intensive applications and highly integrated low power solutions, communication requirements among cores can not be sufficiently satisfied using either traditional or multi-layer bus architectures because of their poor scalability and bandwidth limitation on a single bus. While new interconnection techniques have been explored to overcome such a limitation, the notion of utilizing Network-on-Chip (NoC) technologies for the future generation of high performance and low power chips for myriad of applications, in particular for wireless communication and multimedia processing, has been of great importance [1]. By applying network-like communication which inserts routers in-between each communication object, the interconnection network improves scalability and freedom from the limitation of complex wiring. Replacement of SoC busses by NoCs will follow the same path as data communication systems where from the economics point of view NoC can potentially reduce SoC manufacturing cost, time to market, time to volume, and design risk and at the same time improve performance. According to [2], the NoC approach has a clear advantage over traditional busses and most notably as far as the system throughput is concerned. Though hierarchies of crossbar or multi-layer busses have characteristics somewhere in between traditional busses and NoC, they still fall far short of the NoC with respect to performance and complexity. Recently many researchers have proposed various routing algorithms as well as different router architectures appropriate for on-chip interconnection network environments.

In order to evaluate the performance of either these routing algorithms or their routers, including implementations, many researchers have used conventional traffic patterns [3], [4] or some limited number of real traffic traces. Even though these static traffic patterns exhibit similar patterns of some particular applications, there is a fundamental limit in covering complete traffic patterns of real applications. For this reason, some researchers have used real traffic patterns extracted from real applications to evaluate the performance of their proposed routing algorithm or router based on more practical benchmarks [5], [6].

In traditional networks such as Internet, Ethernet, and wireless LANs transporting TCP/IP, HTTP, and FTP traffic among others, network traffics have been traced and analyzed to understand the traffic behavior of these networks and characterize them. Therefore, various extensive traffic models for diverse networks have been developed [7], [8], [9], [10], [11], [12], [13], [14]. These models provide not only meaningful insight into understanding the traffic behavior of these networks, but have also been effectively used in evaluating current and newly designed networks. Because on-chip interconnection network is a new class of networks where the overall communications occur in a single chip, a similar approach to understanding the behavior of NoC traffic and evaluating networks in NoC environment is needed.

In this paper, we propose a generic traffic model for NoC environments. The proposed model is based on the spatial/temporal profile of traffic using three statistical parameters. These three statistical parameters construct node burstiness, node injection rate, and the distribution of source-to-destination pairs. Different from the other previous proposals where statistical parameters were extracted from overall nodes and formulated in a single statistic model of each component, each statistical parameter is extracted from each node and the associated statistic model is constructed per node. Therefore, the degree of accuracy of the proposed traffic model

TABLE I

CONVENTIONAL STATIC TRAFFIC PATTERNS [3]

Name	*Pattern*
Random	$\lambda_{sd} = 1/N$
Permutation	
Bit permutations	
Bit complement	$d_i = \sim s_i$
Bit reverse	$d_i = s_{b-i-1}$
Bit rotation	$d_i = s_{i+1} \bmod b$
Shuffle	$d_i = s_{i-1} \bmod b$
Transpose	$d_i = s_{i+b/2} \bmod b$
Digit permutations	
Tornado	$d_x = s_x + (\lceil k/2 \rceil) \bmod k$
Neighbor	$d_x = s_x + 1 \bmod k$

in emulating real traffic situation is much higher than the previous work.

The organization of this paper is as follows. Section 2 provides previous related works in the field of traffic modeling and motivation of this paper. Next, Section 3 explains an overview of our traffic model with three different statistical components for NoC. In Section 4, the details of each component are presented and the overall procedure of generating traffic pattern with the given parameters is described. Section 5 validates the accuracy of the proposed traffic model by comparing it with real traffic traces. Finally, Section 6 concludes this paper.

II. RELATED WORK

Conventional traffic patterns consider the spatial distribution of messages in interconnection networks. Therefore, the distribution between source nodes and destination nodes is defined depending on the type of conventional traffic patterns. Table I lists some common static traffic patterns used to evaluate interconnection networks. In Table I, In conventional traffic patterns, random traffic is described by a traffic matrix with all fraction of traffic sent from node λ_{sd} as $1/N$. Permutation traffic, in which all traffic from each source is directed to one destination, can be more compactly represented by a permutation function. Bit permutations are those in which each bit d_i of the b-bit destination address is a function of one bit of the source address, s_i. In digit permutations, each (radix-k) digit of the destination address d_x is a function of a digit s_y of the source address. Historically, several of these patterns are based on communication patterns that arise in particular applications. For instance, matrix transpose or corner-turn operations induce the transpose pattern, whereas fast Fourier transform (FFT) or sorting applications might cause the shuffle permutation, and fluid dynamics simulations often exhibit neighbor patterns [3].

While these models enable a network to be stressed with a regular, predictable pattern and provide NoC researchers with helpful insights, they do not cover real application traffics to explore a realistic NoC design-space. Until now, few researches have been able to present results in the field of realistic NoC traffic models. Varatkar and Marculescu [15] have reported the evidence of self-similarity in NoC burst traffic between on-chip modules in typical MPEG-2 video applications and captured traffic characteristics between pairwise nodes. Also using a generic tile-based communication architecture, they proposed a technique for synthetically generating traces having statistical properties similar to those obtained from real video clips. Soteriou et al. [16] proposed an empirically-derived model of NoC traffic based on traffic traces obtained from full system simulations. Their model comprehensively espouses the spatio-temporal characteristics of traffic with three dimensionless statistical components in a three-tuple model. Also they illustrate two potential uses of their traffic model: how it allows us to characterize and gain insights on NoC traffic patterns, and how it can be used to generate synthetic traffic traces that can drive NoC design-space exploration. Tedesco et al. [17] presented application driven traffic modeling for NoCs. In their work, applications are characterized according to their delivery requirements (e.g. real-time streaming and block transfer) and QoS service levels (e.g. CBR and VBR). Also they identify three methods to model traffic: constant injection rate is the most commonly used, but least accurate. Probabilistic methods are normally used in simulation for applications with variable rates. Finally trace-based traffic models are more suitable for emulation.

III. OVERVIEW

We propose a generic traffic model for NoC based on traffic traces obtained from full system simulation or real system devices. This model combines the spatio-temporal characteristics of traffic with three independent components, (H_s, λ_s, $\delta_{(s,d)}$) where s and d represent the indices of source node and destination node, respectively. With three independent components, the given traffic can be analyzed and characterized in a statistical manner. Different from the approach used in [16], each statistical component is derived per node. To define the burstiness of each node, the Hurst exponent H_s for source node s, is adopted. As a component of the characteristics of self-similarity, H_s determines the temporal burstiness of traffic at each node, that is, the peak size of injection packets and their injection patterns of arrival time. To define one of spatial properties in NoC traffic traces, the distribution of average injection rate at every node, denoted by λ_s is captured. Finally $\delta_{(s,d)}$ representing the distribution of traffic ratio from s node to d node in the given injection rate λ_s is extracted.

For each component of our (H_s, λ_s, $\delta_{(s,d)}$) traffic model, we analyze and extract the proposed statistical distribution against 8 traffic traces used in [18]. Those are SPLASH-2 [19] traces gathered by running the benchmarks on Bochs [20], a multiprocessor simulator with an embedded Linux 2.4 kernel. Each benchmark was run in Bochs with 49 (= 7×7) concurrent threads, and the memory trace is captured. This memory trace is then applied to a memory system simulator that models the classic MSI (Modified, Shared, Invalid) directory-based cache coherence protocol, with the home directory nodes statically assigned based on the least significant bits of the tag, distributed across all processors in the entire chip.

IV. Traffic Modeling

In this Section, we explain the details of our (H_s, λ_s, $\delta_{(s,d)}$) traffic model. Based on the extracted parameters, the procedure for generating a synthetic traffic trace will be provided as well.

A. Temporal Burstiness: H_s

In classic networks, self-similarity is one of the key features to characterize burstiness as well as long-range dependence (LRD) of traffic in the temporal sense. To measure such a burstiness and LRD, the Hurst parameter H is used where $H \in (1/2, 1)$ indicates the presence of LRD. As many communication traffics are proven to be statistically self-similar, some researchers already showed that the traffic in NoC also has a self-similar characteristic [15], [16]. Thus, we parameterize such a degree of burstiness or LRD using H. Furthermore, in order to be accurate, this parameter indicating the burstiness is analyzed on every injection node.

Because the definitions of self-similarity are well described in the literature, in this Section, a brief description of self-similarity will be introduced. For more details, the reader is recommended to read several references [7], [8], [9], [21].

Considering a cumulative process $Y(t)$ with stationary increments, let X_t be its corresponding incremental process:

$$X_t = Y(t) - Y(t-1) \tag{1}$$

The process $X_s^{(m)}$ is defined as an aggregated process of X_t if

$$X_s^{(m)} = [X_{sm-m+1} + X_{sm-m+2} + \ldots + X_{sm}]/m \tag{2}$$

Process X_t is *self-similar* if X_t is indistinguishable from $X_s^{(m)}$. Because this is a very restrictive definition, usually *second-order self-similarity* is considered for traffic analysis, i.e. *auto-covariance* of the original and aggregated processes should be same:

$$\gamma^{(m)}(k) = \gamma(k) \tag{3}$$

$$\lim_{m\to\infty} \gamma^{(m)}(k) = \gamma(k) \tag{4}$$

where $\gamma(k) = E[(X_t - \mu)(X_{t+k} - \mu)]$ and $\gamma^{(m)}(k) = E[(X_s^{(m)} - \mu)(X_{s+k}^{(m)} - \mu)]$. The process X_t is *exactly* second-order self-similar or *asymptotically* second-order self-similar if Eq. (3) or Eq. (4) is satisfied, respectively.

In order to measure the degree of self-similarity, the Hurst parameter H is used where a process is self-similar with parameter $H(0 < H < 1)$ if:

$$Y(t) = k^H Y(kt), \forall k > 0, t \geq 0 \tag{5}$$

which means that the original and normalized aggregated processes should have the same distribution. In other words, the self-similarity can be understood as the ability of an aggregated process to preserve the burstiness of the original process, i.e. the property of *slowly decaying variance*:

$$var(X^{(m)}) \sim m^{2H-2} \tag{6}$$

In this paper, Eq. (6) is computed to measure the Hurst parameter H. Table II shows the measured H value per node for eight different traces.

B. Injection Rate: λ_s

As one of the spatial parameters in our traffic model, traffic injection rate determines the distribution of injection load per node. In [16], this spatial injection distribution is parameterized by the standard deviation σ of the injection distribution with an actual coordinate assignment. In that approach, it assumes that the actual results possess Gaussian-type distributions. Even though that approach can help the injection distribution be quantified using single σ value, the mapping to Gaussian-like distribution is not alway accurate in real NoC traffic situation. Also it requires large amount of computation to find out the exact coordinate assignment. Hence, in this paper, the original distribution of injection rate on every node is kept as it is. This enables more accurate synthetic traffic generation than σ-based Gaussian-like distribution. Figure 1 shows injection rate distributions for traffic traces in a 7×7 mesh.

C. Spatial Distribution: $\delta_{(s,d)}$

Another spatial distribution $\delta_{(s,d)}$ represents the traffic ratio from source node s to destination node d based on the injection rate λ_s. In [16], spatial hop distribution p is adopted. In order to formulate the hop count distribution model, they applied the mechanism so that the mapping should not choose a receiver whose distance is d hops from the sender unless it cannot choose any other node whose distance to the sender is less than d. Also, in that model, there is no concern about the geometry of destination nodes. In other words, all nodes with same d-hop distance from the source node are considered to have the same statistical characteristics. Thus, this assumption is somehow far from the actual NoC traffic regardless of the optimal communication mapping. However, our model considers the difference of location of destination nodes within same distance of hops when the traffic ratio between source and destination node is analyzed. Moreover, the matrix of traffic ratio from each source node is constructed in order to characterize the spatial distribution of source/destination pairs. Figure 2 illustrates the distribution of traffic ratio for each node in the `barnes` application.

D. Synthetic Traffic Generation

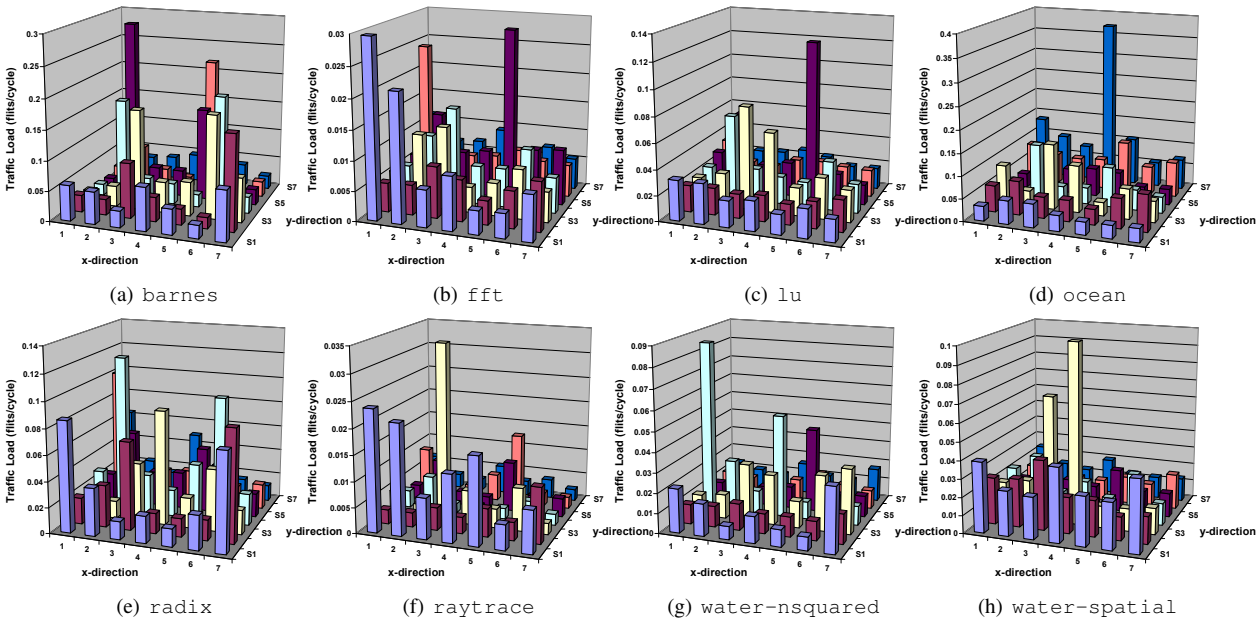
To describe how our (H_s, λ_s, $\delta_{(s,d)}$) traffic model can generate synthetic network traffic, we implemented ***tgNePA***, a tool that automatically generates NoC traffic of the given network topology from the configured (H_s, λ_s, $\delta_{(s,d)}$) traffic model. Figure 3 shows the traffic generation flow in ***tgNePA***.

tgSelfSimilar*: Traffic generation based on (H_s, λ_s).** To generate self-similar NoC traces, ***tgNePA uses the method described in [22]. In this method, the synthetic self-similar traffic is obtained by aggregating multiple sub-streams, each consisting of alternating Pareto-distributed on/off periods. Pareto distribution is defined by a heavy-tailed distribution with the probability-density function $f(x) = ab^\alpha/x^{\alpha+1}, x \geq b$ where α is a shape parameter, and b is a location parameter. Pareto distribution with $1 < \alpha < 2$ has a finite mean and

TABLE II

MEASURED HURST PARAMETER FOR TRAFFIC TRACES IN 7×7 MESH

barnes							fft						
0.95322	0.930023	0.976069	0.976115	0.931223	0.958229	0.967662	0.975095	0.941273	0.948791	0.964113	0.951594	0.956773	0.961565
0.953165	0.958007	0.906468	0.927599	0.932508	0.952065	0.938917	0.95084	0.951059	0.966805	0.966816	0.960732	0.952544	0.963932
0.961447	0.950251	0.960211	0.976686	0.923968	0.9375	0.864216	0.95335	0.978645	0.977989	0.962314	0.958788	0.968619	0.963367
0.958195	0.890648	0.918889	0.93756	0.958898	0.927886	0.904769	0.961431	0.973687	0.992332	0.959887	0.96177	0.968292	0.964245
0.956649	0.985442	0.939089	0.920304	0.904582	0.938582	0.885551	0.964755	0.972209	0.972016	0.97097	0.990788	0.968756	0.974325
0.961257	0.88265	0.938839	0.95921	0.96127	0.917485	0.928582	0.993044	0.972676	0.971063	0.975506	0.976879	0.973728	0.970238
0.869166	0.965638	0.895383	0.906786	0.909061	0.883905	0.88992	0.975961	0.980018	0.977654	0.973719	0.968751	0.964043	0.960366
lu							ocean						
0.924199	0.953505	0.956322	0.953416	0.955267	0.961355	0.949458	0.880499	0.808062	0.794881	0.845817	0.799616	0.816938	0.832904
0.958744	0.95534	0.953154	0.954404	0.954908	0.952774	0.956247	0.7324	0.784654	0.830542	0.826276	0.77571	0.753008	0.795864
0.954391	0.967936	0.985967	0.980213	0.953937	0.95738	0.956906	0.901085	0.865756	0.723628	0.749781	0.781642	0.752834	0.742008
0.959172	0.982929	0.961082	0.954406	0.95408	0.967851	0.956651	0.802515	0.757628	0.798828	0.75681	0.760965	0.790886	0.750362
0.963854	0.9776	0.95748	0.965053	0.992639	0.962783	0.954991	0.82159	0.810766	0.726558	0.748242	0.777665	0.78399	0.78296
0.967356	0.950398	0.952623	0.964804	0.954611	0.961498	0.962472	0.818765	0.770188	0.809174	0.785652	0.803659	0.762584	0.782661
0.956253	0.960129	0.957397	0.95746	0.957518	0.95611	0.954621	0.837008	0.761783	0.87089	0.763205	0.77494	0.78582	0.778135
radix							raytrace						
0.902839	0.944403	0.96101	0.972136	0.964553	0.947608	0.985433	0.942255	0.961226	0.940355	0.962855	0.95834	0.932357	0.963871
0.957888	0.942072	0.971374	0.973585	0.966459	0.964246	0.977311	0.945377	0.948391	0.943875	0.936158	0.964887	0.943996	0.961554
0.964332	0.957286	0.980439	0.98841	0.983089	0.985034	0.974669	0.945114	0.957771	0.974867	0.950478	0.960309	0.968862	0.945402
0.969146	0.983704	0.971503	0.977166	0.96415	0.981327	0.957322	0.92635	0.967857	0.956754	0.949919	0.951917	0.974327	0.968231
0.973206	0.982417	0.97069	0.960063	0.975664	0.963702	0.969817	0.929483	0.961821	0.964149	0.936166	0.974707	0.941642	0.946531
0.93137	0.966252	0.969756	0.975648	0.958885	0.963826	0.96038	0.949944	0.93797	0.93593	0.943694	0.958023	0.934166	0.947076
0.946714	0.966286	0.961873	0.944223	0.947943	0.965468	0.962096	0.963812	0.963292	0.934751	0.959196	0.958551	0.960313	0.949267
water-nsquared							water-spatial						
0.938858	0.954619	0.961653	0.961435	0.964605	0.959879	0.983422	0.954124	0.941421	0.96231	0.980993	0.970395	0.968765	0.979926
0.958101	0.944468	0.963525	0.957574	0.95899	0.957685	0.966726	0.967107	0.964892	0.976365	0.954031	0.954496	0.95555	0.951143
0.954599	0.952399	0.978571	0.973547	0.959643	0.974414	0.978607	0.956379	0.958845	0.985977	0.992441	0.946157	0.954781	0.949645
0.995104	0.974171	0.963846	0.987655	0.952222	0.966112	0.952725	0.950117	0.957826	0.949018	0.943914	0.948824	0.960112	0.94836
0.951494	0.971262	0.954555	0.958582	0.982684	0.958186	0.965041	0.94065	0.939393	0.957809	0.954648	0.962962	0.949257	0.950307
0.958905	0.95965	0.959323	0.961995	0.965036	0.963629	0.962181	0.95947	0.95342	0.956035	0.95468	0.955787	0.955492	0.955214
0.961457	0.955258	0.957095	0.965474	0.952312	0.959847	0.965554	0.969339	0.965527	0.963964	0.966033	0.961421	0.960948	0.962182



(a) barnes (b) fft (c) lu (d) ocean

(e) radix (f) raytrace (g) water-nsquared (h) water-spatial

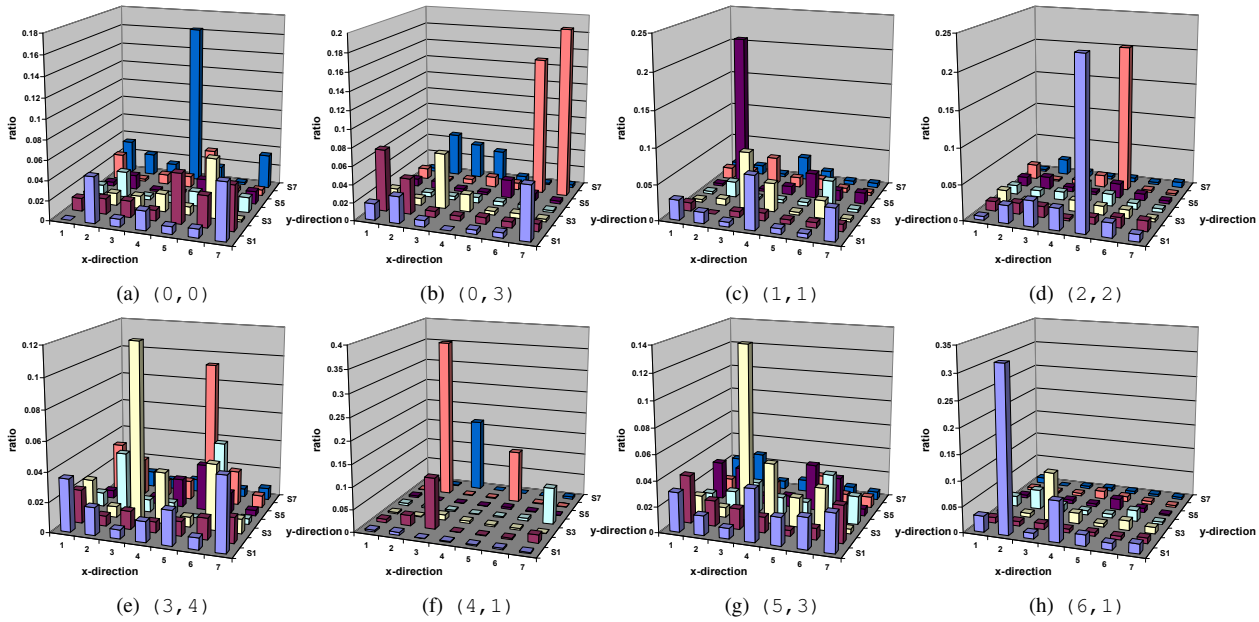
Fig. 1. Injection rate distributions for traffic traces in 7×7

an infinite variance. To generate Pareto-distributed values, the following formula is used: $X_{Pareto} = b/[U^{1/\alpha}]$ where U is a uniform random variable ($0 \leq U \leq 1$). The Hurst parameter H of self-similar trace generated by this method can be derived by $H = (3-\alpha)/2$ [22], [23].

Additionally, while generating Pareto-distributed values, the injection rate for each sub-stream can be controlled. Therefore, by applying λ_s to each generation of self-similar stream for the corresponding node s, the (H_s, λ_s) configured self-similar traffic can be obtained.

Depending on the method of self-similar traffic generation, its accuracy may be varied. To minimize the error between the expected (H_s, λ_s) and the measured value from the generated traffic, a recursion is applied as shown in the first phase *tgSelfSimilar* of Figure 3. Along with generating self-similar traffic with the expected (H_s, λ_s) configuration, (H'_s, λ'_s)-tuple components of the generated traffic are measured. If the error of the expected (H_s, λ_s) and the measured (H'_s, λ'_s) is acceptable, then the generated self-similar traffic is delivered to the next step *splitPE*. Otherwise, the generation of self-similar traffic with the similar configuration is repeated.

***splitPE*: Traffic generation based on** $\delta_{(s,d)}$. The second phase generates the destination node upon the generated self-similar traffic of each node. Because the ratio of traffic from each source node s to each destination node d is already provided by the distribution of $\delta_{(s,d)}$, the generation of destination



(a) (0,0) (b) (0,3) (c) (1,1) (d) (2,2)

(e) (3,4) (f) (4,1) (g) (5,3) (h) (6,1)

Fig. 2. Distributions of traffic ratio on selected nodes for barnes traffic trace in 7×7

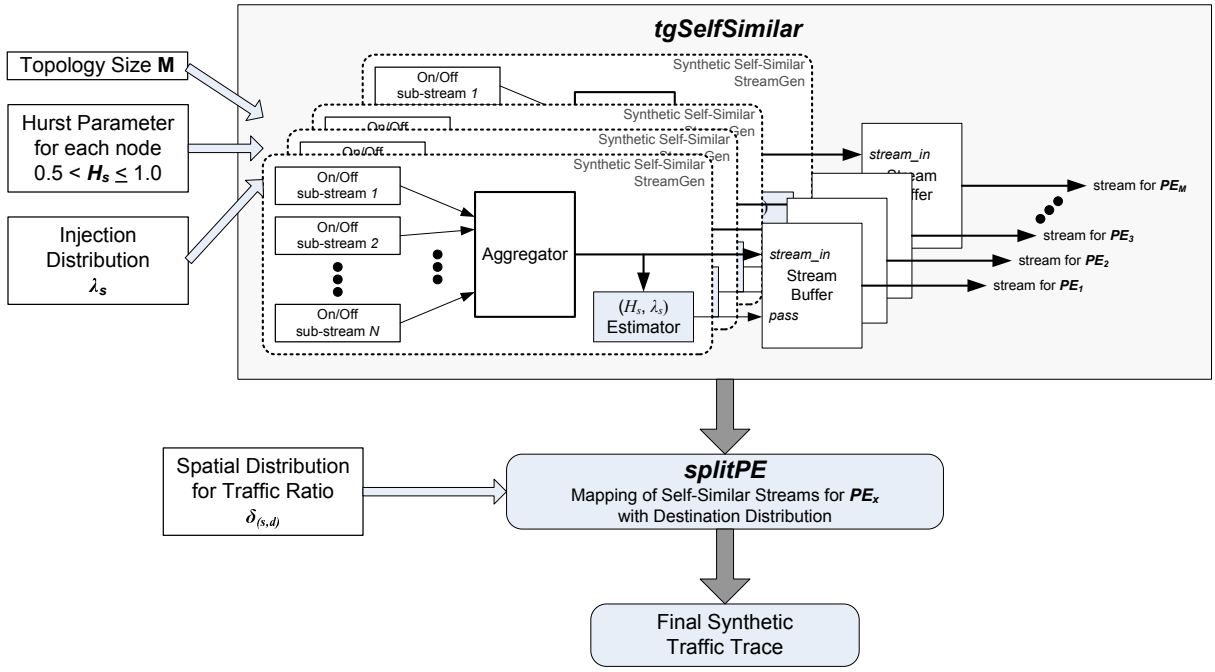


Fig. 3. *tgNePA* traffic generation flow diagram

node for each instance of traffic from the corresponding source node s can be accurately controlled randomly. Different from the Trident's approach [16], the ratio of traffic for each pair of source and destination is separately assigned. Therefore, the distribution of source/destination pairs can be more accurately emulated.

V. Validation

Each synthetic traffic is generated using the analyzed (H_s, λ_s, $\delta_{(s,d)}$) for each application mentioned in the previous Section. In order to control the recursion of *tgSelfSimilar*, we set the marginal error bound of H_s and λ_s to 5%. In recursive generation of self-similar traffic for each node, the Hurst parameter H_s can be easily matched with the given marginal percentage. However, in matching the lower injection rate λ_s, it requires excessive computation time. For that reason, to reduce such a large computation time in matching the injection rate, a proportional margin value is applied as an alternative approach. That is, in relatively higher injection rate, the tighter margin value is applied. Reversely, in relatively lower injection

rate, the lighter margin value is applied. For instance, by using logarithmic scale of injection rate, the marginal value can be scaled by multiplying $|\log_{10}(\lambda_s)|^{|\log_{10}(\lambda_s)|}$. Because the higher injection rate is dominant, the effect of larger error at source nodes with lower injection rates can be minimized. Table III and Table IV show the measured Hurst parameter and injection rates of synthetic traffic according to the analyzed traffic model of each application. For H_s parameter in synthetic traffic, the accuracy is in the range of 2.7% to 4.3% average error bound. However, the accuracy of λ_s is varied depending on the level of injection rates of applications because the propotional margin value to the level of injection rates is applied in matching the injection rate during the first phase of traffic generation. For instance, in `barnes` application, the average of injection rates of original traffic is 0.065 and the ratio of average error in injection rates is 6.8%. On the other hand, in `fft` application, the average of injection rates of original traffic is 0.0089 and the ratio of average error is 26%. In this case, the level of injection rates is relatively low, i.e. the scale factor to apply a propotional margin is 27 (=3^3) during the recursion. Therefore, the resultant synthetic traffic has relatively large error from the original injection rates.

For source/destination distribution $\delta_{(s,d)}$ of synthetic traffic, its accuracy is almost 100% as shown in Figure 4.

Finally, throughout the cycle accurate NoC simulation [24], [25] using original traffic traces as well as synthetic traffic traces, the accuracy of overall network performance is observed. As shown in Table V, the synthetic trafic patterns for applications except for `fft` and `raytrace` have maximum 17% error from the perspective of the offered load. For two exceptional applications with high error ratio in the offered load, their offered load is relatively low. Therefore, even a small difference results in a large percentage of error ratio.

VI. Conclusion and Future Works

We proposed a generic traffic model for on-chip interconnection networks. To keep the temporal and spatial distribution of traffic traces, every statistical information is measured per node. In order to characterize the burstiness of injection nodes, the Hurst parameter H_s is selected. For specifying the temporal statistics, the distribution of injection rates λ_s and ratio of source/destination pairs $\delta_{(s,d)}$ on the given source node are used. With the proposed traffic model, we also introduced a recursive traffic generation method to minimize the error of statistical components, and allow synthetic traffic traces with similar temporal and spatial statistics to be generated. Throughout detailed comparison of each component and performance simulation, our proposed traffic model can reconstruct traffic patterns with a similar tendancy of real NoC traffic and provide insights into NoC traffic.

As the future works, an advanced methodology needs to be developed to validate the accuracy of synthetic traffic patterns. In this paper, only the statistical measurement such as comparing average parameters, which does not evaluate the accuracy in time, is used. To be scalable, the proposed traffic model should be tested in different size or type of networks.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *DAC '01: Proceedings of the 38th Conference on Design Automation*. New York, NY, USA: ACM, 2001, pp. 684–689.

[2] ARTERIS, "A comparison of network-on-chip and busses," http://www.arteris.com/noc_whilepaper.pdf.

[3] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2004.

[4] K. Lahiri, S. Dey, and A. Raghunathan, "Evaluation of the traffic-performance characteristics of system-on-chip communication architectures," in *VLSID '01: Proceedings of the 14th International Conference on VLSI Design*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 29–35.

[5] J. Hu and R. Marculescu, "Dyad: smart routing for networks-on-chip," in *DAC '04: Proceedings of the 41st annual conference on Design automation*. New York, NY, USA: ACM, 2004, pp. 260–263.

[6] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "A methodology for design of application specific deadlock-free routing algorithms for noc systems," in *CODES+ISSS '06: Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis*. New York, NY, USA: ACM, 2006, pp. 142–147.

[7] P. Doukhan, G. Oppenheim, and M. S. Taqqu, *Theory and Applications of Long-Range Dependence*. Birkhäuser Boston, December 2002.

[8] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*. New York, NY, USA: John Wiley & Sons, Inc., September 2000.

[9] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Network*, vol. 2, no. 1, pp. 1–15, February 1994.

[10] V. Paxson and S. Floyd, "Wide-area traffic: The failure of poisson modeling," in *SIGCOMM '94: Proceedings of the Conference on Communications Architectures, Protocols and Applications*. New York, NY, USA: ACM, 1994, pp. 257–268.

[11] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan, "Characterizing user behavior and network performance in a public wireless lan," *SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 195–205, June 2002.

[12] I. Y. Bucher and D. A. Calahan, "Models of access delays in multiprocessor memories," *IEEE Trans. Parallel Distributed Systems*, vol. 3, no. 3, pp. 270–280, May 1992.

[13] F. Darema-Rogers, G. F. Pfister, and K. So, "Memory access patterns of parallel scientific programs," in *SIGMETRICS '87: Proceedings of the 1987 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. New York, NY, USA: ACM, 1987, pp. 46–58.

[14] S. W. Turner, "Performance analysis of multiprocessor interconnection networks using a burst-traffic model," Ph.D. dissertation, Champaign, IL, USA, 1995.

[15] G. V. Varatkar and R. Marculescu, "On-chip traffic modeling and synthesis for mpeg-2 video applications," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 1, pp. 108–119, January 2004.

[16] V. Soteriou, H. Wang, and L.-S. Peh, "A statistical traffic model for on-chip interconnection networks," in *MASCOTS '06 : Proceeding of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 104–116.

[17] L. Tedesco, A. Mello, L. Giacomet, N. Calazans, and F. Moraes, "Application driven traffic modeling for nocs," in *SBCCI '06: Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems Design*. New York, NY, USA: ACM, 2006, pp. 62–67.

[18] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," in *ISCA '07: Proceedings of the 34th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2007, pp. 150–161.

[19] "Splash-2," http://www-flash.stanford.edu/apps/SPLASH/.

[20] K. P. Lawton, "Bochs: A portable pc emulator for unix/x," *Linux Journal*, vol. 1996, no. 29es, p. 7, September 1996.

[21] B. Tsybakov and N. D. Georganas, "Self-similar processes in communications networks," *IEEE Trans. Information Theory*, vol. 44, no. 5, pp. 1713–1725, September 1998.

[22] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," *SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 5–23, April 1997.

TABLE III

MEASURED HURST PARAMETER FOR SYNTHETIC TRAFFIC TRACES IN 7×7 MESH

barnes						
0.922604	0.900864	0.929619	0.932654	0.887087	0.924292	0.924685
0.910217	0.92359	0.894023	0.892669	0.900064	0.922199	0.916118
0.913906	0.924976	0.918239	0.93298	0.878349	0.890662	0.87693
0.933494	0.873282	0.90397	0.912061	0.924956	0.891938	0.887996
0.926644	0.949166	0.894207	0.888905	0.879505	0.917606	0.898869
0.940055	0.898484	0.894776	0.911445	0.925797	0.888928	0.922339
0.831892	0.917855	0.917632	0.892862	0.879119	0.886606	0.862894
						average error ratio = 0.032

fft						
0.930469	0.901261	0.904385	0.919863	0.905215	0.920039	0.927163
0.918589	0.930192	0.932514	0.93347	0.921322	0.913654	0.953305
0.907441	0.929731	0.931737	0.91638	0.911257	0.935479	0.91756
0.927311	0.927169	0.950335	0.933219	0.934989	0.923255	0.925058
0.930802	0.92372	0.95463	0.924153	0.942931	0.920807	0.942047
0.950361	0.932204	0.937753	0.935888	0.929717	0.925567	0.928409
0.944464	0.931497	0.933646	0.927396	0.928349	0.932712	0.920281
						average error ratio = 0.041

lu						
0.920638	0.913734	0.926406	0.914993	0.912014	0.929957	0.930625
0.915871	0.913395	0.923574	0.93993	0.914495	0.925637	0.910338
0.919504	0.95263	0.937157	0.935734	0.908091	0.915819	0.916393
0.917747	0.936518	0.91593	0.915539	0.912704	0.94054	0.92322
0.919705	0.933109	0.91106	0.918234	0.944434	0.923524	0.913384
0.926757	0.904367	0.90686	0.927362	0.912072	0.916251	0.91677
0.917779	0.929412	0.920856	0.918243	0.920673	0.918731	0.924098
						average error ratio = 0.039

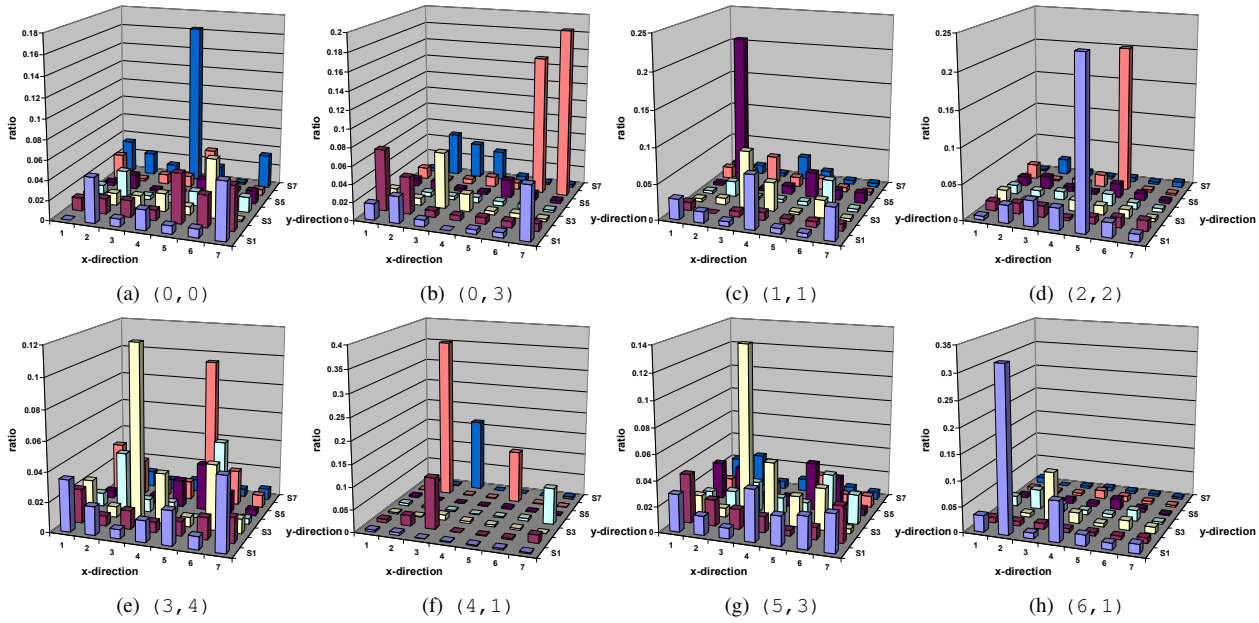
ocean						
0.8483	0.783373	0.770411	0.875487	0.766814	0.828058	0.824162
0.739303	0.77524	0.798285	0.8164	0.80788	0.719764	0.780451
0.895681	0.849855	0.709589	0.728271	0.758329	0.756766	0.705397
0.841761	0.745178	0.807255	0.721676	0.736948	0.787291	0.71473
0.837015	0.799498	0.699795	0.717817	0.815538	0.754053	0.804242
0.805689	0.780835	0.790708	0.818814	0.764959	0.742495	0.753796
0.799607	0.76778	0.853409	0.795997	0.752484	0.785473	0.752427
						average error ratio = 0.027

radix						
0.887586	0.927803	0.92179	0.927197	0.920037	0.902051	0.94445
0.912924	0.912437	0.92907	0.935315	0.930807	0.920588	0.93943
0.94104	0.929403	0.936362	0.946848	0.956438	0.937805	0.931888
0.941251	0.936482	0.925917	0.930899	0.918615	0.940462	0.915708
0.928504	0.94163	0.923772	0.921857	0.933592	0.916538	0.9224
0.892508	0.92093	0.927384	0.932629	0.915048	0.933493	0.91771
0.923702	0.926957	0.916387	0.906687	0.904082	0.917981	0.93198
						average error ratio = 0.041

raytrace						
0.90344	0.923164	0.94793	0.921658	0.919907	0.930598	0.916547
0.933545	0.915805	0.938897	0.945765	0.93872	0.899207	0.93716
0.908462	0.912697	0.931622	0.937748	0.918021	0.927636	0.909152
0.953294	0.941418	0.915519	0.924081	0.906276	0.946177	0.926767
0.919059	0.916655	0.923873	0.925912	0.949737	0.902823	0.905981
0.953268	0.903407	0.921178	0.905038	0.914827	0.890978	0.909914
0.934584	0.917112	0.917526	0.934793	0.918665	0.937771	0.920748
						average error ratio = 0.032

water-nsquared						
0.898308	0.923437	0.940954	0.922098	0.923039	0.913735	0.934255
0.937719	0.915814	0.925254	0.911514	0.913656	0.911644	0.930135
0.913194	0.919129	0.936486	0.927012	0.919423	0.930479	0.932936
0.946839	0.928766	0.920046	0.945132	0.909967	0.92056	0.91277
0.913586	0.923798	0.90929	0.913297	0.93518	0.913151	0.921679
0.917045	0.914046	0.917961	0.918059	0.917218	0.919949	0.918271
0.916328	0.948779	0.910059	0.925256	0.912018	0.919631	0.91887
						average error ratio = 0.043

water-spatial						
0.924825	0.907196	0.917813	0.935485	0.928298	0.923866	0.931682
0.91907	0.921237	0.929094	0.907	0.929373	0.920545	0.915879
0.924795	0.913452	0.938763	0.944932	0.8999	0.923134	0.906942
0.907262	0.918599	0.904109	0.905521	0.91763	0.927629	0.90155
0.901858	0.903175	0.930372	0.908927	0.917512	0.912481	0.926686
0.920825	0.909482	0.909743	0.915424	0.927488	0.922289	0.920409
0.931546	0.919134	0.919032	0.919456	0.925384	0.922665	0.92463
						average error ratio = 0.041



(a) (0,0) (b) (0,3) (c) (1,1) (d) (2,2)

(e) (3,4) (f) (4,1) (g) (5,3) (h) (6,1)

Fig. 4. Distributions of traffic ratio on selected nodes for synthetic traffic trace of barnes application in 7×7

[23] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level," *SIGCOMM Computer Communication Review*, vol. 25, no. 4, pp. 100–113, October 1995.

[24] J. H. Bahn, S. E. Lee, and N. Bagherzadeh, "On design and analysis of a feasible network-on-chip (noc) architecture," in *ITNG '07: Proceedings of the International Conference on Information Technology*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1033–1038.

[25] ——, "Design of a router for network-on-chip," *International Journal of High Performance Systems Architecture*, vol. 1, no. 2, pp. 98–105, 2007.

TABLE IV

COMPARISON OF INJECTION RATES BETWEEN ORIGINAL AND SYNTHETIC TRAFFIC TRACES IN 7×7 MESH

original traffic							synthetic traffic						
barnes													
0.059230	0.053724	0.027742	0.072139	0.042431	0.022027	0.083564	0.082931	0.055818	0.034412	0.067748	0.044069	0.032461	0.099190
0.028138	0.026441	0.091737	0.040663	0.027105	0.018616	0.157872	0.034190	0.034794	0.100266	0.042767	0.025398	0.023744	0.175239
0.021208	0.034636	0.167460	0.051668	0.056595	0.171639	0.074054	0.023813	0.043012	0.187606	0.060602	0.057168	0.164082	0.080500
0.018661	0.168760	0.039418	0.035184	0.021129	0.190589	0.027948	0.023320	0.160879	0.043212	0.038176	0.021441	0.183314	0.036472
0.015542	0.290371	0.044580	0.043946	0.154056	0.052408	0.025951	0.017888	0.324491	0.062108	0.059061	0.153222	0.052371	0.026421
0.024136	0.064680	0.028955	0.018944	0.225996	0.028720	0.025712	0.025948	0.067938	0.031171	0.026097	0.254005	0.033386	0.029015
0.112588	0.033496	0.037716	0.046235	0.030914	0.038005	0.022475	0.115044	0.045602	0.046282	0.056193	0.035571	0.060112	0.027397
average error ratio = 0.066													
fft													
0.029666	0.021464	0.006244	0.008981	0.003974	0.004094	0.007642	0.044174	0.022187	0.011228	0.013225	0.007996	0.008160	0.011472
0.004875	0.005072	0.008658	0.006987	0.004060	0.006191	0.008248	0.009111	0.007835	0.015021	0.013979	0.008017	0.011784	0.015275
0.005504	0.012254	0.013928	0.004325	0.005509	0.008301	0.005121	0.009686	0.016122	0.018494	0.008304	0.009961	0.013941	0.009420
0.005197	0.010923	0.015967	0.006519	0.006589	0.010280	0.006026	0.010128	0.021058	0.019755	0.011395	0.010854	0.019347	0.010866
0.005839	0.013510	0.007209	0.007880	0.028822	0.009014	0.009372	0.010099	0.016749	0.013635	0.014502	0.029346	0.017603	0.017984
0.024218	0.006013	0.005391	0.005813	0.007263	0.005757	0.005479	0.028910	0.011271	0.013258	0.013190	0.017048	0.009654	0.010647
0.010977	0.006434	0.006770	0.009269	0.005420	0.007020	0.005331	0.020612	0.011626	0.012168	0.016830	0.010672	0.011232	0.010410
average error ratio = 0.27													
lu													
0.031522	0.031486	0.020856	0.023430	0.015734	0.022615	0.017442	0.035190	0.033321	0.024563	0.027929	0.016210	0.027098	0.020902
0.020890	0.021314	0.019032	0.020641	0.016205	0.020843	0.024755	0.022980	0.024165	0.022151	0.022552	0.018941	0.023302	0.029638
0.020656	0.026310	0.080935	0.062681	0.022026	0.032159	0.025184	0.024147	0.031387	0.084642	0.073180	0.025317	0.037863	0.028438
0.023138	0.066632	0.025918	0.021733	0.019832	0.038455	0.022993	0.026949	0.076103	0.028331	0.023800	0.021964	0.045441	0.027337
0.029276	0.062923	0.023083	0.026969	0.124827	0.022259	0.015869	0.034131	0.067290	0.025380	0.030227	0.131003	0.026182	0.018190
0.033716	0.017579	0.016078	0.023511	0.024204	0.022110	0.021272	0.039662	0.020108	0.017948	0.027392	0.027782	0.023304	0.024658
0.021180	0.021210	0.022325	0.025828	0.022391	0.016425	0.016723	0.024829	0.024846	0.026269	0.030037	0.026304	0.018605	0.019671
average error ratio = 0.13													
ocean													
0.032656	0.052407	0.053066	0.034972	0.028506	0.029530	0.030476	0.035787	0.057154	0.057344	0.040709	0.032434	0.034281	0.034655
0.059482	0.076336	0.048041	0.047850	0.035247	0.066863	0.082566	0.063344	0.083863	0.051858	0.053600	0.039598	0.070091	0.087853
0.086836	0.035133	0.147355	0.105736	0.030472	0.067765	0.047462	0.096221	0.039389	0.150070	0.109829	0.033589	0.066591	0.050848
0.033205	0.123656	0.033226	0.037146	0.090287	0.046643	0.037019	0.036830	0.128103	0.038558	0.038684	0.093770	0.050305	0.038934
0.031955	0.035925	0.056059	0.063624	0.060362	0.034188	0.035604	0.037264	0.041204	0.057429	0.063482	0.064157	0.036824	0.038539
0.086679	0.068636	0.064304	0.067920	0.113985	0.062790	0.078873	0.098110	0.074006	0.071210	0.077321	0.119170	0.066208	0.081037
0.134312	0.096478	0.077706	0.371114	0.106355	0.055866	0.069098	0.139972	0.103311	0.085479	0.383716	0.110420	0.059998	0.076108
average error ratio = 0.067													
radix													
0.085885	0.036942	0.013971	0.020770	0.013779	0.026615	0.075784	0.088893	0.044055	0.015655	0.023982	0.014646	0.031706	0.087981
0.020545	0.032627	0.068298	0.015802	0.014502	0.016102	0.086488	0.023399	0.039103	0.071430	0.017994	0.016906	0.018508	0.098817
0.015641	0.013742	0.045850	0.088507	0.023404	0.047885	0.018831	0.017750	0.016331	0.052414	0.106077	0.027570	0.055200	0.021370
0.029354	0.122007	0.030738	0.020711	0.043255	0.096830	0.024630	0.031458	0.125732	0.036056	0.021926	0.049505	0.114297	0.028689
0.021256	0.057330	0.027209	0.028700	0.050168	0.035776	0.018404	0.022906	0.065534	0.032163	0.032332	0.060077	0.040638	0.021660
0.100576	0.014770	0.022440	0.024386	0.019568	0.015721	0.019553	0.105422	0.016417	0.026233	0.027580	0.022311	0.018466	0.023020
0.063424	0.022736	0.014779	0.049019	0.025052	0.015360	0.012427	0.074280	0.026441	0.015996	0.056243	0.028725	0.017785	0.014806
average error ratio = 0.13													
raytrace													
0.023664	0.021465	0.007970	0.013069	0.016971	0.004876	0.008253	0.026600	0.024533	0.015101	0.015422	0.018870	0.007748	0.013036
0.002765	0.002812	0.004423	0.003225	0.005511	0.003513	0.010775	0.006357	0.006265	0.008340	0.005851	0.009994	0.006027	0.012393
0.004837	0.002790	0.034439	0.006664	0.003718	0.008369	0.002179	0.006846	0.005459	0.039667	0.013511	0.006941	0.016444	0.004676
0.003348	0.006765	0.001843	0.002009	0.002106	0.004217	0.002200	0.007861	0.013654	0.004120	0.003902	0.004186	0.008048	0.004354
0.002909	0.008605	0.002654	0.002212	0.009745	0.003546	0.003611	0.006384	0.012687	0.005938	0.004182	0.012975	0.007668	0.008029
0.009046	0.002361	0.002462	0.005211	0.013867	0.002560	0.002273	0.019626	0.005373	0.005123	0.007910	0.014637	0.004454	0.004494
0.006204	0.002753	0.002233	0.005590	0.002890	0.003757	0.002269	0.013494	0.004253	0.004473	0.008239	0.005175	0.007742	0.004435
average error ratio = 0.55													
water-nsquared													
0.022009	0.016074	0.006600	0.013258	0.008507	0.006761	0.032313	0.023661	0.018801	0.009436	0.014747	0.010829	0.008875	0.035534
0.009806	0.010507	0.013351	0.009940	0.010067	0.009624	0.014857	0.013117	0.012558	0.015723	0.014398	0.010760	0.011149	0.016496
0.010534	0.012138	0.029008	0.025092	0.013357	0.027807	0.032449	0.012128	0.013980	0.034044	0.028754	0.015152	0.031198	0.038481
0.085102	0.025618	0.011492	0.051211	0.008794	0.012639	0.009679	0.096370	0.027683	0.013569	0.057958	0.010880	0.014441	0.013205
0.008935	0.018494	0.009593	0.010994	0.042124	0.009718	0.010690	0.012895	0.020918	0.011417	0.012446	0.049889	0.013163	0.012201
0.010824	0.009237	0.008589	0.010808	0.015551	0.010871	0.009337	0.012715	0.012761	0.011246	0.012833	0.016926	0.013036	0.012725
0.012572	0.009745	0.008497	0.015741	0.010567	0.009653	0.016860	0.014897	0.012041	0.009559	0.018343	0.012647	0.013809	0.019778
average error ratio = 0.18													
water-spatial													
0.039134	0.024893	0.023346	0.041009	0.027349	0.025811	0.039816	0.045553	0.029164	0.026863	0.046777	0.031364	0.030894	0.046882
0.025982	0.026905	0.038835	0.020124	0.020240	0.013104	0.015139	0.027518	0.029762	0.044945	0.023105	0.023776	0.014670	0.017940
0.019024	0.021667	0.069799	0.099872	0.012199	0.012508	0.014645	0.022233	0.024674	0.080859	0.111213	0.014298	0.014812	0.016507
0.022947	0.031320	0.012389	0.019785	0.012086	0.027014	0.012403	0.025280	0.036956	0.013214	0.022760	0.013890	0.031608	0.014042
0.012658	0.021574	0.014751	0.012530	0.023434	0.012939	0.013315	0.014053	0.023360	0.016113	0.012756	0.024740	0.014915	0.015571
0.015907	0.011776	0.011851	0.011625	0.015237	0.012631	0.019781	0.018469	0.013941	0.013507	0.013421	0.016467	0.014772	0.021343
0.023773	0.014841	0.012444	0.019787	0.012952	0.012253	0.013045	0.026972	0.017753	0.014759	0.021032	0.014798	0.013726	0.014950
average error ratio = 0.14													

TABLE V

COMPARISON OF CYCLE ACCURATE NOC SIMULATION BETWEEN ORIGINAL AND SYNTHETIC TRAFFIC TRACES IN 7×7 MESH

	original traffic		synthetic traffic		error ratio (%)	
application	offered load	avg. latency	offered load	avg. latency	offered load	avg. latency
barnes	0.06522	7.26	0.06951	7.32	6.58	0.86
fft	0.00889	8.20	0.01125	7.95	26.63	2.94
lu	0.03560	7.67	0.03240	7.52	8.99	2.17
ocean	0.06881	7.65	0.07345	7.78	6.75	1.70
radix	0.03690	7.96	0.04177	8.00	13.18	0.42
raytrace	0.00636	7.99	0.00987	7.95	55.21	0.53
water-nsquared	0.01649	7.93	0.01939	7.77	17.60	2.22
water-spatial	0.02221	7.99	0.02529	7.76	13.83	2.89