An Analytical Latency Model for Networks-on-Chip

Abbas Eslami Kiasari, *Student Member, IEEE*, Zhonghai Lu, *Member, IEEE*, and Axel Jantsch, *Member, IEEE*

Abstract—We propose an analytical model based on queueing theory for delay analysis in a wormhole-switched network-on-chip (NoC). The proposed model takes as input an application communication graph, a topology graph, a mapping vector, and a routing matrix, and estimates average packet latency and router blocking time. It works for arbitrary network topology with deterministic routing under arbitrary traffic patterns. This model can estimate per-flow average latency accurately and quickly, thus enabling fast design space exploration of various design parameters in NoC designs. Experimental results show that the proposed analytical model can predict the average packet latency more than four orders of magnitude faster than an accurate simulation, while the computation error is less than 10% in non-saturated networks for different system-on-chip platforms.

Index Terms—Modeling and prediction, network-on-chip (NoC), performance analysis and design aids, queueing theory.

I. INTRODUCTION

L ATENCY is recognized as one of the most critical design characteristics for on-chip interconnection network architectures [17]. In this work, we propose a performance model which predicts the latency of flows in a network-on-chip (NoC)-based system. Performance models are frequently employed by system designers for early architecture and design decisions. Typically, engineers construct a performance model, and then compare future technology options based on performance model projections. To this end, application and architecture models are first developed separately. Then, the application is mapped to the architecture and a performance model is used to evaluate the chosen application-architecture combination. Nowadays, most performance models of NoCs rely on simulations [2], [19]. The use of simulation experiments makes the task of searching for efficient designs computationally intensive and does not scale well with the size of networks. Therefore, it is simply impossible to use the simulation in optimization loops.

An alternative approach is an analytical model which can estimate the desired performance metrics in a fraction of time. Analytical models can be used to prune the large design space in a very short time compared to simulation. Thus, it is justified to derive accurate analytical models for performance prediction of NoCs to eliminate the need for time consuming simulations. The information provided during the performance analysis step can be used in any optimization loop for NoCs such as topology selection, application mapping, and buffer allocation. Although the use of high-level models conceals a lot of complex technological aspects, it facilitates fast exploration of the NoC design space. Accurate simulations can be setup at later steps of design process when the design space is reduced to a few practical choices.

In this research a performance queueing (PQ) model, is proposed and evaluated for NoCs. The PQ model, which is based on a G/G/1 queueing model, has been developed for deterministic routing and wormhole switching. The proposed model is topology-independent and supports any kind of spatial and temporal traffic patterns. The estimated performance metrics such as average latency and router blocking time can be conveniently used for optimization purposes to find appropriate design parameters, as well as obtaining quick performance estimates. Our results show that the PQ model calculates quickly the latency of flows in the network with less than 10% error when compared to the simulation. This gives us confidence that we can utilize the model in the early design phase of high performance on-chip networks.

The rest of this paper is organized as follows. We start by reviewing previous studies and highlighting our contribution in Section II. Since our work is based on queueing theory, we give a very brief review of G/G/1 queues and priority queues in Section III. The proposed performance model is then described in Section IV, while Section V compares the modeling results and those obtained through accurate simulations. Finally, concluding remarks and future work plans are given in Section VI.

II. RELATED WORK

Much of the previous analytical latency models in wormhole-switched off-chip networks have been formulated for a specific topology and traffic pattern [12], [13]. In [7], the authors utilized a queueing model and presented a performance model to overcome the problem of buffer allocation in NoC-based systems, but the approach cannot handle the wormhole-switched networks. The authors in [6] addressed the allocation of link capacities in NoCs through an analytical latency model. Their proposed model, however, only works for networks with single flit buffers and also ignores the queueing delays and network contentions. A more accurate analytical router model has been proposed in [16]. This work assumes that packet arrivals to the network follow the Poisson distribution. As a result, such models lack the accuracy for use in applications with bursty traffic such as multimedia application. In [11] a mathematical performance model for NoC-based systems was proposed to predict performance metrics in NoCs. However, the modeling approach was limited to k-ary n-cube networks with single flit buffers and dimension-order routing algorithm. A worst-case analysis of flow latency in the NoC-based systems was considered in [8].

This paper optimizes the traffic regulation parameters aiming for buffer optimization. Although this approach is proper for such a system with real-time requirements, many NoC-based systems have more relaxed timing constraints.

To the best of our knowledge, this work proposes the first average case analytical model for on-chip routers which takes into account the burstiness of the traffic. The proposed model can be used to develop a thorough performance analysis for arbitrary network topology with wormhole switching under arbitrary traffic pattern. Our proposed model, besides providing performance metrics such as average latency and router blocking time, gives useful feedbacks about the network behavior which can be used in an optimization loop for NoCs such as topology selection, application mapping, and buffer allocation.

III. FOUNDATION

Queueing theory is an appropriate and useful modeling tool for system analysis and performance evaluation in computer and telecommunications network [14]. Since our proposed model has been constructed on the G/G/1 priority queue [3], [22], in this section we give a quick review on the G/G/1 queue and priority queue concepts.

A. G/G/1 Queue

The G/G/1 model has a single service facility with one server, unlimited waiting room and the first-come first-served queue discipline. The service times are independent and identically distributed with a general distribution, the interarrival times of customers are also independent and identically distributed with a general distribution, and the interarrival times are independent of the service times. It is assumed that the general interarrival time and service time distributions are each partially specified by their first two moments. We should remind here that the nth moment of a random variable X is defined as the average of $X^n (\overline{X^n} = \Sigma_{i=1}^{k}(X_i)^n/k)$. All descriptions of this model thus depend only on the basic parameter 4-tuple $(\bar{a}, \overline{a^2}, \bar{s}, \overline{s^2})$, where $\bar{a}$ and $\overline{a^2}$ are the first and second moments of the customers' interarrival time, and similarly, $\bar{s}$ and $\overline{s^2}$ are the first and second moments of the service time. Also in this work we consider the arrival rate and service rate as $\lambda = 1/\bar{a}$ and $\mu = 1/\bar{s}$, respectively. The mean waiting time of a G/G/1 queueing system can be approximated by Allen-Cunneen formula [3]

$$\bar{W}_{G/G/1} \approx \frac{\rho \left(C_A^2 + C_S^2 \right)}{2\mu(1-\rho)} \tag{1}$$

where ρ is the utilization factor of the server and equal to λ/μ, and C_A and C_S are the coefficient of variation (CV) of the interarrival time and service time respectively [3]. We remind that the relationship between CV of random variable X and its moments is represented by $C_X^2 = \overline{x^2}/\bar{x}^2 - 1$.

B. Priority Queue

We consider a system with one server in which the customers have preferential treatment based on priorities associated with them. We assume that the priority of a customer is an integer fixed at arrival time, and a customer with priority $i(i = 1, 2, \ldots, p)$ belongs to class i. We say one customer has
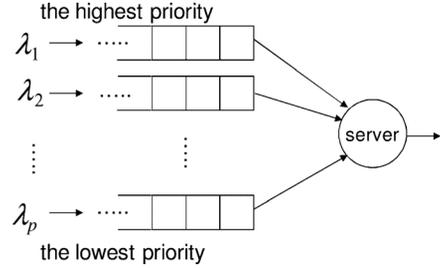
Fig. 1. Typical priority queueing system.

higher priority than another if it belongs to a priority class with lower index. In other words, the lower the index, the higher the priority. The priority queueing system to be studied is depicted in Fig. 1, where the different queue levels correspond to the different priority classes. For the service discipline, we assume that whenever a customer is completed, the server is next assigned to that customer at the head of the highest priority nonempty queue. Once a customer begins on the server, it is allowed to run to completion; i.e., the service discipline is nonpreemptive. Independent and identically distributed arrivals and service times are assumed for the ith class with the arrival and service rate denoted by λ_i and μ_i, respectively. The mean waiting time of random arrivals to the ith queue $\bar{W}_i$ can be written as [22]:

$$\bar{W}_i = \frac{\bar{R}}{\left(1 - \Sigma_{k=1}^{i-1}\rho_k \right) \left(1 - \Sigma_{k=1}^{i}\rho_k \right)} \tag{2}$$

where $\bar{R}$ is the residual service time seen by an incoming customer. In a G/G/1 queueing system, $\bar{R}$ is approximated by [3]

$$\bar{R} \approx \Sigma_{k=1}^{p} \frac{\rho_k}{2\mu_k} \left(C_{A_k}^2 + C_{S_k}^2 \right) \tag{3}$$

where μ_k and ρ_k are average service rate and utilization factor of class k, respectively. Also, C_{A_k} and C_{S_k} are CV of interarrival time and service time of class k, respectively.

In all the analysis we have reviewed so far, the queue size of each class was infinite. However, in the case of wormhole switching this is not a true assumption, because in wormhole switching each buffer can hold only finite number of flits. Later in Section V-B, we analyze such a queueing system.

IV. PERFORMANCE ANALYSIS

The following assumptions are made when developing the proposed performance model.

- The PQ model works for deterministic routing algorithms which may be minimal or non-minimal.
- The switching method is wormhole and messages are broken into packets.
- There is one finite FIFO queue per channel and channels are allocated per packet. It means that the channel is released when the whole packet has passed through the channel.
- Packets are consumed immediately by the destination node.

In order to characterize network performance, architecture and application models are essential.
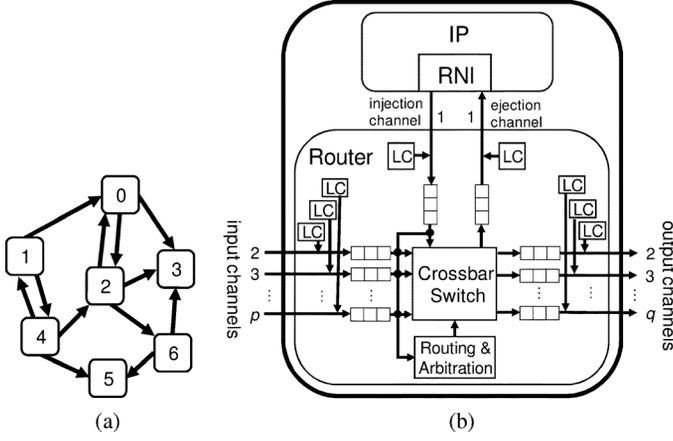
Fig. 2. (a) Graph representation of a general NoC architecture. (b) Structure of a node in an NoC-based system.

Fig. 3. Delay of a one hop flow.

A. Architecture Model

As shown in Fig. 2(a), a directed graph can represent the topology of an NoC architecture. Vertices and edges of the graph show nodes and channels of the NoC, respectively. The structure of a single node is depicted in Fig. 2(b). Every node contains an intellectual property (IP) core and a router with p input channels and q output channels. Each IP core performs its own computational, storage or I/O processing functionality, and is equipped with a resource-network-interface (RNI). The RNI translates data between IP cores and routers by packing/unpacking data packets and also manages the packet injection process. Packets are injected into the network on the injection channel (input port 1) and leave the network from the ejection channel (output port 1). Generally, each channel connects output port j of node N to input port i of node M. Therefore, we denote this channel OC_j^N (jth output channel of router N) or IC_i^M (ith input channel of router M). We consider the general reference architecture for routers in [4] and it comprises the following major components.

- *Buffer*. This is a finite FIFO buffer for storing packets in transit. In the model shown in Fig. 2(b), a buffer is associated with each input physical channel and each output physical channel. In alternative designs, buffers may be associated only with inputs (input buffering) or outputs (output buffering).
- *Link controller* (LC). The flow of packets across the physical channel between adjacent routers is implemented by the link controller. The link controllers on either side of a channel coordinate to transfer flits.
- *Crossbar switch*. This component is responsible for connecting router input channels to router output channels.
- *Routing and arbitration unit*. This component implements the routing algorithms, selects the output channel for an incoming packet, and accordingly sets the crossbar switch. Routing is only performed with the head flit of a packet. If two or more packets simultaneously request the same output channel, the arbiter must provide for arbitration among them. In this work, we suppose that input channels have a descending order of priority in a clockwise direction for each output channel. The incoming packets
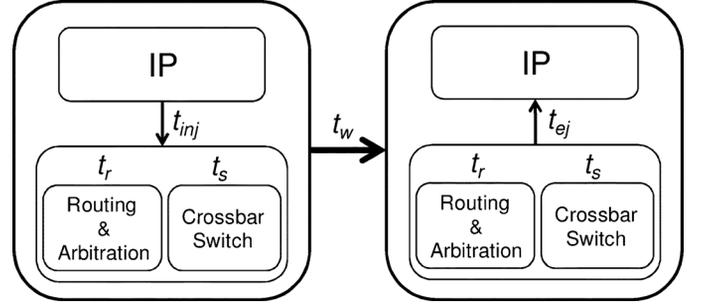
from injection channel have the highest priority in each priority group. Usually, a control mechanism prevents the network from being overloaded. Therefore, it is guaranteed that the router is never overloaded and incoming packets from lower priority channels do not face starvation. If the requested output channel is busy, the incoming head flit remains in the input buffer. It will be routed again after the channel is freed and if it successfully arbitrates for the channel.

Similar to the network model in [4], we suppose that the routing decision delay for a packet, crossing time of a flit over the crossbar switch, and transfer time of a flit across a wire between two adjacent routers are t_r, t_s, and t_w, respectively. Also the transfer times of a flit across the injection and ejection channels are considered to be t_{inj} and t_{ej}, respectively. Having looked at Fig. 3, we can infer that the latency of a head flit of a one hop packet in the absence of contention includes injection channel delay (t_{inj}), first router delay ($t_r + t_s$), inter-node wire delay (t_w), second router delay ($t_r + t_s$), ejection channel delay (t_{ej}). Therefore, we can write it as $t_{inj} + (t_r + t_s) + t_w + (t_r + t_s) + t_{ej}$.

In this study, we consider the wormhole switching under deterministic routing algorithm. Although adaptive routing algorithms avoid congested channels and result in more balanced load on the network, they may cause out-of-order packet delivery. The reorder buffers needed at the destination for ordering the packets impose large area and power on system [15]. Deterministic routers not only are more compact and faster than adaptive routers, but also guarantee in-order packet delivery. Therefore, it is not surprising that designers would like to use deterministic routing algorithms in the NoCs which desire small silicon overheads. Thus, in this research we use the deterministic routing for deadlock-free routing.

B. Application Model

The target application can be specified by the *communication graph* [18]. The communication graph is a directed graph where each vertex represents an IP core, and the directed edge represents the communication between cores. The weight of the edge represents the communication rate between source and destination. In experimental results section, we consider the communication graph of a multimedia application (see Table III). Although generation of data packets in NoC nodes has dependence, especially in application-specific platforms, the studies in [1], [21] show that compared to real traffic traces in NoCs, it

TABLE I
PARAMETER NOTATION

t_r	Time spent for packet routing decision (*cycles*)	
t_s	Time spent for switching (*cycles*)	
t_w	Time spent for transmitting a flit between two adjacent routers (*cycles*)	*Architecture parameters*
m	Average size of packets (*flits*)	
σ_m	Standard deviation of packet size (*flits*)	
$L^{S \to D}$	Average packet latency from IP^S to IP^D (*cycles*)	
L	Average packet latency in the network (*cycles*)	
IP^N	The IP core located at address N	
R^N	The router located at address N	
IC_i^N	The ith input channel in router R^N	
OC_j^N	The jth output channel in router R^N	
IB_i^N	Capacity of the buffer in IC_i^N (*flits*)	
OB_j^N	Capacity of the buffer in OC_j^N (*flits*)	
$P^{S \to D}$	Probability of a packet is generated in IP^S and is delivered to IP^D ($\sum_S \sum_D P^{S \to D} = 1$)	
λ^N	Average packet injection rate of IP^N (*packets/cycle*)	
$\lambda_{i \to j}^N$	Average packet rate from IC_i^N to OC_j^N (*packets/cycle*)	
λ_j^N	Average packet rate to OC_j^N (*packets/cycle*) $\left(\lambda_j^N = \sum_i \lambda_{i \to j}^N\right)$	*Application parameters*
$P_{i \to j}^N$	Probability of a packet entered form IC_i^N to be exited from OC_j^N	
μ_j^N	Average service rate of the OC_j^N (*packets/cycle*)	
$\bar{R}_j^N$	Residual service time of OC_j^N seen by an incoming flow (*cycle*)	
$C_{B_j^N}$	Coefficient of variation (CV) for service time of the OC_j^N	
$C_{A_{i \to j}^N}$	CV for interarrival time of packets from IC_i^N to OC_j^N	
$\rho_{i \to j}^N$	The fraction of time that the OC_j^N is occupied by packets from IC_i^N	
$W_{i \to j}^N$	Average waiting time for a packet from IC_i^N to OC_j^N (*cycles*)	

will be still accurate to model their traffic generation separately as independent bursts of packets with statistical characteristics.

We assume that the packet injection process to the router N has a general distribution with mean value of λ^N packets/ cycle and coefficient of variation of C_A. Also, the probability of packet transmission from the source node S to the destination node D is $P^{S \to D}$. This information can be easily extracted from the communication graph of application. Messages are broken into some packets with arbitrary size distribution. m and σ_m represent the average and standard deviation of the packet size, respectively, as listed in Table I along with other parameters.

V. COMMUNICATION ANALYSIS

To have a better view of the proposed model, the main idea of the analysis approach is summarized here.
  a) To estimate the average latency of flows, it is essential to estimate the packet waiting times for network channels.
  b) Each channel is modeled as a G/G/1 priority queue and the waiting time to access each channel is calculated based on the packet arrival rate and channel service time which are calculated in (c) and (d), respectively.
  c) Given the communication volume among IP cores and routing algorithm, the packet arrival rate to each channel is determined.
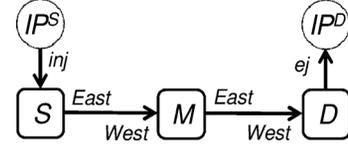
Fig. 4. Two-hops flow from IP^S (source) to IP^D (destination).

  d) The channel service time, which is part of the waiting time, is calculated recursively for each communication path starting from the destination node.

A. Latency Model

The average packet latency (L) is used as the performance metric. We assume that the packet latency spans the instant when the packet is created, to the time when the packet is delivered to the destination node. We also assume that the packets are consumed immediately once they reach their destination nodes.

In Fig. 4, consider a flow which is generated in IP^S, and reaches its destination (IP^D) after traversing R^S, R^M, and R^D. The latency of this packet $(L^{S \to D})$ consists of two parts: the latency of head flit $(L_h^{S \to D})$ and the latency of body flits (L_b). In other words

$$L^{S \to D} = L_h^{S \to D} + L_b. \tag{4}$$

$L_h^{S \to D}$ is the time since the packet is created in IP^S, until the head flit reaches the IP^D, including the queueing time spent at the source node and intermediate nodes. In Fig. 4, $L_h^{S \to D}$ can be computed as

$$
\begin{aligned}
L_h^{S \to D} = \; & t_{inj} + \left(t_r + W_{inj \to \text{East}}^S + t_s\right) \\
& + t_w + \left(t_r + W_{\text{West} \to \text{East}}^M + t_s\right) \\
& + t_w + \left(t_r + W_{\text{West} \to ej}^D + t_s\right) + t_{ej} \tag{5}
\end{aligned}
$$

where $W_{i \to j}^N$ is the mean waiting time for a packet from IC_i^N to OC_j^N. Note that in Fig. 4, the channel between S and M can be addressed with OC_{East}^S or IC_{West}^M.

Once the head flit arrives at the destination, the flow pipeline cycle time is determined by the maximum of the switch delay and wire delay. For an input-only or output-only buffered router, this cycle time would be given by the sum of the switch and wire delays [4]. In other words, in an input-output buffered router L_b is given by

$$L_b = (m - 1) \times \max(t_s, t_w) \tag{6}$$

and in an input-only or output-only buffered router it is

$$L_b = (m - 1)(t_s + t_w). \tag{7}$$

The only unknown parameter for computing the latency is $W_{i \to j}^N$. This value can be calculated using a queueing model.

B. Waiting Time Estimation

A router is primarily modeled based on nonpreemptive priority queueing system. Let us consider, for instance, the jth output channel of $R^N (OC_j^N)$. As can be seen in Fig. 5, this
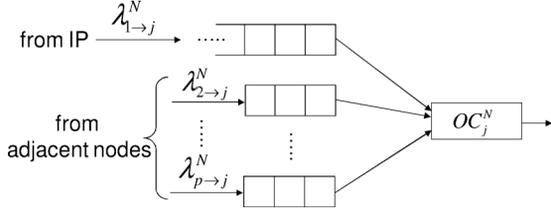
Fig. 5. Queueing model of a channel of an arbitrary topology.

channel is modeled as a server in a priority queueing system with p classes (IC_1^N to IC_p^N), the arrival rate $\lambda_{i \to j}^N (1 \le i \le p)$, and served by one server (OC_j^N) of service rate μ_j^N. Both interarrival and service times are independent and identically distributed with arbitrary distributions.

The queueing model for output channel represented in Fig. 5 is different from traditional priority queue model in Fig. 1. Since in the wormhole switching each input buffer can hold finite number of flits, we cannot use (2) and we have to compute the average waiting time for the head of class i in this special case of priority queues. Using a technique similar to that employed in the literature for general priority queues [3], [22], we can write

$$W_{i \to j}^N = \begin{cases} \bar{R}_j^N \Big/ \left(1 - \rho_{1 \to j}^N\right), & i = 1 \\ \bar{R}_j^N \Big/ \left(1 - \Sigma_{k=1}^{i-1} \rho_{k \to j}^N\right)^2, & 2 \le i \le p \end{cases} \quad (8)$$

where $\rho_{k \to j}^N$ is the fraction of time that the OC_j^N is occupied by packets from IC_k^N and equals

$$\rho_{k \to j}^N = \lambda_{k \to j}^N \big/ \mu_j^N. \quad (9)$$

Also $\bar{R}_j^N$ is the residual service time of OC_j^N seen by an incoming head flit. Based on (3), in a G/G/1 queueing system the residual service time is approximated by [3]

$$\bar{R}_j^N \approx \Sigma_{i=1}^p \rho_{i \to j}^N \frac{C_{A_{i \to j}^N}^2 + C_{S_j^N}^2}{2\mu_j^N}. \quad (10)$$

Since we do not have enough insight about the CV of interarrival time at each channel ($C_{A_{i \to j}^N}$), we suppose that $C_{A_{i \to j}^N}$ is the same for all input channels in the network and equal to the coefficient of variation of the arrival process to network ($C_{A_{i \to j}^N} = C_A$). Therefore, we can rewrite (10) as

$$\bar{R}_j^N \approx \frac{C_A^2 + C_{S_j^N}^2}{2\mu_j^N} \Sigma_{i=1}^p \rho_{i \to j}^N. \quad (11)$$

Due to the definition of $\rho_{i \to j}^N$, we can write $\Sigma_{i=1}^p \rho_{i \to j}^N = \Sigma_{i=1}^p (\lambda_{i \to j}^N / \mu_j^N)$. It is obvious that the average packet rate to an output channel of R^N is equal to sum of the average packet rate from all input channel of R^N to this output channel. Therefore, we can write $\Sigma_{i=1}^p \lambda_{i \to j}^N / \mu_j^N = \lambda_j^N / \mu_j^N = \rho_j^N$. As a result, (11) can be rewritten as

$$\bar{R}_j^N \approx \rho_j^N \left(C_A^2 + C_{S_j^N}^2\right) \Big/ 2\mu_j^N. \quad (12)$$

By substituting $\bar{R}_j^N$ in (8) we can write

$$W_{i \to j}^N = \begin{cases} \dfrac{\rho_j^N \left(C_A^2 + C_{S_j^N}^2\right)}{2\left(\mu_j^N - \lambda_{i \to j}^N\right)}, & i = 1 \\ \dfrac{\lambda_j^N \left(C_A^2 + C_{S_j^N}^2\right)}{2\left(\mu_j^N - \Sigma_{k=1}^{i-1} \lambda_{k \to j}^N\right)^2}, & 2 \le i \le p. \end{cases} \quad (13)$$

Therefore, to compute the $W_{i \to j}^N$ we have to calculate the arrival rate from IC_i^N to $OC_j^N (\lambda_{i \to j}^N)$, and also first and second moments of the service time of $OC_j^N (\bar{S}_j^N, \overline{(S_j^N)^2})$. In the following two subsections, packet arrival rate and channel service time are computed.

C. Packet Arrival Rate Calculation

Assuming the network is not overloaded, the arrival rate from IC_i^N to OC_j^N can be calculated using the following general equation

$$\lambda_{i \to j}^N = \Sigma_S \Sigma_D \lambda^S \times P^{S \to D} \times R\left(S \to D, IC_i^N \to OC_j^N\right). \quad (14)$$

In (14), the routing function $R(S \to D, IC_i^N \to OC_j^N)$ equals 1 if a packet from IP^S to IP^D passes from IC_i^N to OC_j^N; it equals 0 otherwise. Note that we assume a deterministic routing algorithm, thus the function of $R(S \to D, IC_i^N \to OC_j^N)$ can be predetermined, regardless of topology and routing algorithm. After that, the average packet rate to OC_j^N can be easily determined as

$$\lambda_j^N = \Sigma_i \lambda_{i \to j}^N. \quad (15)$$

D. Channel Service Time Estimation

After estimating the packet arrival rates, now we focus on the estimation of the moments of channel service times. At first, we assign a positive integer index to each output channel. Let D_j^N be the set of all possible destinations for a packet which passes through OC_j^N. The index of OC_j^N is equal to the maximum of distances among N and each M where $M \in D_j^N$. Obviously, the index of a channel is between 1 and diameter of the network. In addition, the index of all ejection channels is supposed to be 0. After that, all output channels are divided into some groups based on their index numbers, so that group k contains all channels with index k.

Determination of the channel service time moments starts at group 0 (ejection channels) and works in ascending order of group numbers. Therefore, the waiting time from lower numbered groups can then be thought of as adding to the service time of packets on higher numbered groups. In other words, to determine the waiting time of channels in group k, we have to calculate the waiting time of all channels in group $k - 1$. This approach is independent of the network topology and works for all kinds of deterministic routing algorithm, whether minimal or non-minimal.

In the ejection channel of R^N, the head flit and body flits are accepted in $t_s + t_w$ and L_b cycles, respectively. Therefore, we
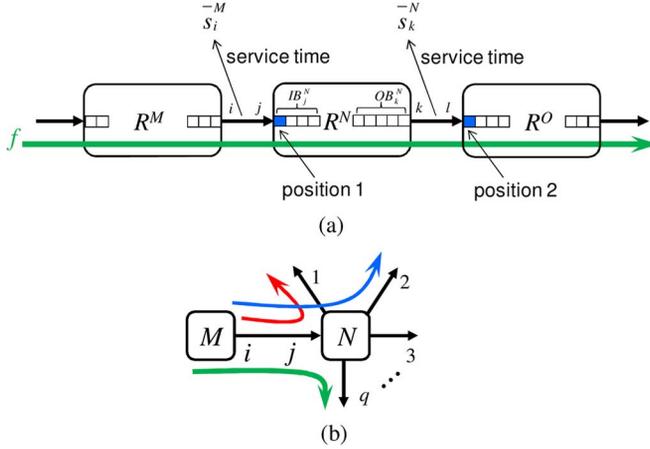
Fig. 6. (a) Passing flow from R^M, R^N and R^O. (b) Some possible path for an entering flow to R^N.

can write $\bar{s}_1^N = t_s + t_w + L_b$ and since the standard deviation of packet size is known, we can easily compute $C_{s_1^N}$. Now, by using (13), the waiting time of input channels for ejection channel $W_{i \to 1}^N$ can be determined for all nodes in the network, where $2 \le i \le p$.

Although the moments of service time can be computed simply for all ejection channels, service time moments of the other output channels cannot be computed in a direct manner by a general formula, and we have to use a more complicated approach. Consider flow f in Fig. 6(a) which passes through routers M, N and O. We suppose that the average service time of OC_k^N ($\bar{s}_k^N$) with index x has been computed before, and now we want to compute the average service time of OC_i^M ($\bar{s}_i^M$) with index $x + 1$. At the first glance, it seems that the average service time of OC_i^M is equal to $t_s + t_w + t_r + W_{j \to k}^N + \bar{s}_k^N$. However, we should ponder the effect of buffer spaces in input and output port of a router on channels service time. In the Fig. 6(a), when the tail flit of the passing packet through OC_k^N reaches position 2, the service time of OC_k^N is finished and similarly the service time of OC_i^M is finished, when the tail flit of the packet reaches position 1. Therefore, the preceding equation should be decreased by the spent time for reaching position 2 from position 1. Therefore, we can write $t_s + t_w + t_r + W_{j \to k}^N + \bar{s}_k^N - (IB_j^N + OB_k^N) \times \max(t_s, t_w)$, where IB_j^N and OB_k^N are the capacity of the buffer in IC_j^N and OC_k^N, respectively.

Although the effect of buffer size on the channel service time is considered in this equation, it does not work in all cases. Because, as shown in Fig. 6(b), there might be several paths for different flows in OC_i^M, so we should consider the possibility of using several output channels to make the next hop. Now, we can estimate the first moment or average service time of OC_i^M as

$$\bar{s}_i^M = \Sigma_{k=1}^q P_{j \to k}^N \left(\begin{array}{c} t_s + t_w + t_r + W_{j \to k}^N + \bar{s}_k^N \\ - \left(IB_j^N + OB_k^N \right) \times \max(t_s, t_w) \end{array} \right) \tag{16}$$

where $P_{j \to k}^N$ is the probability of a packet entered form IC_j^N to be exited from OC_k^N and equals

$$P_{j \to k}^N = \lambda_{j \to k}^N / \lambda_k^N. \tag{17}$$

Here, we should remind that to calculate $\bar{s}_i^M$, all values of $\bar{s}_k^N (1 \le k \le q)$ must be computed before. Likewise, the second moment of service time of OC_i^M can be approximated by

$$\overline{\left(S_i^M\right)^2} = \Sigma_{k=1}^q P_{j \to k}^N \left(\begin{array}{c} t_s + t_w + t_r + W_{j \to k}^N + \bar{s}_k^N \\ - \left(IB_j^N + OB_k^N \right) \times \max(t_s, t_w) \end{array} \right)^2. \tag{18}$$

Finally, the CV of channel service time for OC_i^M can be given by

$$C_{s_i^M}^2 = \overline{\left(S_i^M\right)^2} \Big/ \left(\bar{S}_i^M\right)^2 - 1. \tag{19}$$

Now, we are able to compute the average waiting time of all output channels using (13). After computing $W_{i \to j}^N$ for all nodes and channels, the average packet latency between any two nodes in the network, $L^{S \to D}$, can be calculated. The average packet latency is the weighted mean of these latencies

$$L = \Sigma_S \Sigma_D P^{S \to D} \times L^{S \to D} \tag{20}$$

where $P^{S \to D}$ is the probability of a packet is generated in IP^S and is delivered to IP^D.

E. Analysis Flow

To have a clear view of our proposed analysis approach, the flowchart description of the performance model is shown in Fig. 7. Average packet latency in the network is computed in following steps.

Step 1) Given the application communication graph, we can easily extract the temporal and spatial features of communication among IP cores with the computational complexity of $O(n^2)$ where n is the number of nodes in the network.

Step 2) After a mapping phase, the traffic input rates to network channels are computed. The computational complexity of this step is proportional to n^2 and d, where d is the diameter of the network. As a result the overall complexity of this step is obtained as $O(n^2 d)$.

Step 3) Statistical distribution of channel service times are partially computed using (16), (18), and (19). The computational complexity of this step is $O(nq^2)$, where q is the number of output ports per router.

Step 4) After computing the channel service times, the average waiting time of packets are computed with the complexity of $O(np^2 q)$, where p is the number of input ports.

Step 5) The complexity of the average latency calculation using (20) is $O(n^2 d)$ as with Step 2.

As a result, the overall complexity of the PQ model is obtained as $O(n^2 d) + O(np^3)$, if the number of input and output ports of routers are the same. More especially, in the case of 2-D mesh network, a router is connected to maximum four neighboring routers and also a local IP core through injection and ejection channels. Therefore, p and q equal 5 and d is proportional to $\sqrt{n}$.
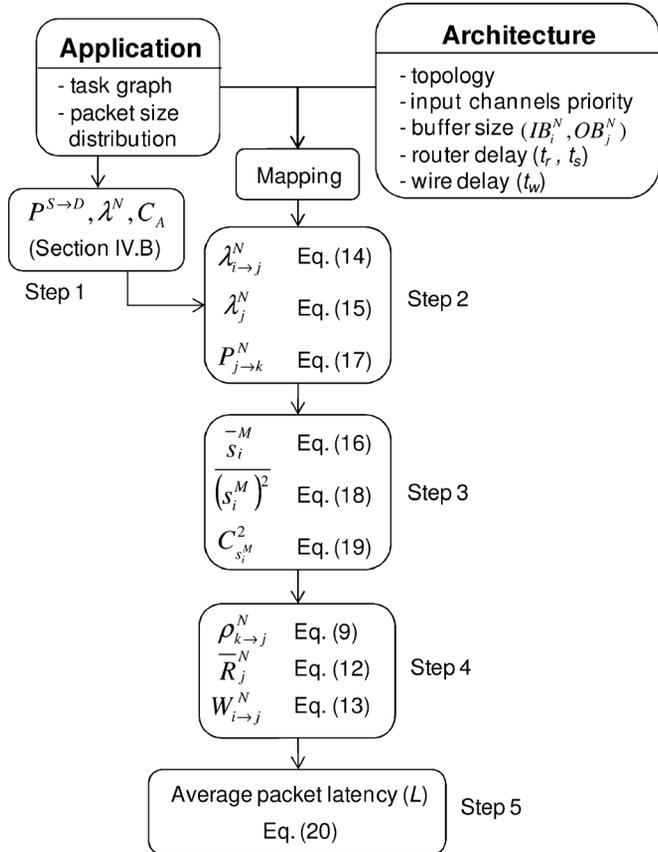
Fig. 7. Flowchart of proposed analytical model.

Fig. 8. (a) Average packet latency of all flows against simulation results. (b) Some selected flows of uniform traffic in a 9×9 mesh network. (c) The average packet latency of the flows in (b), predicted by the PQ model against simulation results.

As a result, the proposed model has time requirement $\mathrm{O}(n^{5/2})$ for 2-D mesh networks.

VI. EXPERIMENTAL RESULTS

The proposed analytical model has been validated through a discrete-event simulator that mimics the behavior of the routing algorithm in the network at the flit level. The simulator uses the same assumptions as the analytical model. To achieve a high accuracy in the simulation results, we use the batch means method [20] for simulation output analysis. There are 10 batches and each batch includes 1000 up to 80 000 000 packets depending on the workload type, traffic injection rate, packet length, and network size. Statistics gathering was inhibited for the first batch to avoid distortions due to the startup transient. The standard deviation of latency measurements is less than 1.8% of the mean value. As a result, the confidence level and confidence interval of simulation results are 0.99 and 0.02, respectively [20]. In other words, the probability of $0.98\ \mu_X \leq \bar{X} \leq 1.02\ \mu_X$ is 0.99, where μ_X is the real average value and $\bar{X}$ is the estimated average value by simulator [20].

For the sake of comprehensive study, numerous validation experiments have been performed for several combinations of workload types, network sizes and packet lengths. In what follows, the accuracy of PQ model will be assessed in multi-processor system-on-chip and application-specific system-on-chip platforms. Since their applications differ starkly in purpose, these classes of NoCs have substantially different traffic patterns.

A. Multi-Processor System-on-Chip Platform

We have considered a 9×9 mesh on-chip interconnect and input-output buffered router with four flits in each input and output channel. It takes two clock cycles to pass a flit within a router and 1 clock cycle to transmit a flit between neighboring routers. We also consider the XY routing algorithm to route the data packets among IP cores. Packet destinations are uniformly distributed across the network nodes. Following a Poisson process, nodes generate packets independently of each other. It means that the time between two successive packet generations in an IP core is distributed exponentially. The Poisson model widely used in many performance analysis studies, and there are a large number of papers in many application domains that are based on this stochastic assumption [7].

Fig. 8(a) depicts latency results predicted by the PQ model explained in the previous section, plotted against those provided by the simulator for the two different fixed packet lengths $m = 4$ and 64 flits. The horizontal axis in the figure shows the packet generation rate while the vertical axis shows the average packet latency. The figure reveals that in both cases the analytical model predicts the average latency with a good degree of accuracy. However, some discrepancies around the saturation point are apparent. These can be accounted for by the approximations made to facilitate the derivation of different variables, e.g., the approximation made to estimate CV of the interarrival time of each channels. Such an approximation greatly simplifies the model as it allows us to avoid computing the exact distribution of the interarrival time at a given channel, which is not a straightforward task due to interdependencies between successive arrival times at channels as wormhole switching relies on a
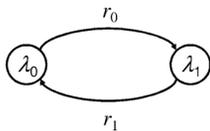
Fig. 9. Two-state MMPP model.

blocking mechanism for flow control. However, the analytical model can still predict the average latency fairly accurately in almost all traffic regions which are appropriate for network operations.

Also, we compare the average latency of some selected flows in the network predicted by the PQ model and the simulator. Fig. 8(b) shows these flows from node 0 in the corner of the network and node 40 in the centre of the network. Fig. 8(c) depicts the average latency of these flows when the flit injection rates are 0.18 and 0.12 flits/cycle/node for the packet length of 4 and 64 flits, respectively. The comparison results show that the model is in good conformity with the simulator with average relative error of 7.5%.

B. Application-Specific System-on-Chip Platform

Analyzing the multimedia applications in NoCs shows bursty patterns of traffic over a wide range of time scales [23]. Since the Poisson process cannot model the bursty traffic very well, we use Markov-modulated Poisson process (MMPP) model [5] to model the temporal burstiness of traffic. MMPP has been widely employed to model the traffic burstiness in the temporal domain [5]. Fig. 9 shows a two-state MMPP in which the arrival traffic follows a Poisson process with rate λ_0 and λ_1. The transition rate from state 0 to 1 is r_0, while the rate from state 1 to state 0 is r_1.

In this study, we use the notation $\mathrm{MMPP}(k)$ for the two-state MMPP in which $\lambda_1 = k\lambda_0$. Fig. 10 shows the number of packet arrivals in a node against time for different values of k when the mean generation rate is 0.01 packet/cycle. Fig. 10(a) vividly shows that Poisson process $(k = 1)$ cannot model the traffic burstiness and Fig. 10(b)–(f) reveal that greater k results in greater intensity of packet burstiness.

The distribution of the interarrival times in the two-state MMPP is a second order hyper-exponential distribution [9]. Therefore, it is easy to compute the coefficient of variation of the interarrival time (C_A) which is reported in Table II. We can infer that the greater the k, the greater the C_A. It means that C_A reflects the burstiness intensity very well. Here we recall that C_A is used in (13) to estimate the packet waiting time.

To evaluate the capability of the proposed model to predict the performance of application-specific applications, we applied it to a generic multimedia system (MMS), which includes an H.263 video encoder, an H.263 video decoder, an mp3 audio encoder, and an mp3 audio decoder [7]. MMS includes 40 tasks and the tasks are assigned into 16 selected IPs. The communication volume requirements (in bytes) of this application are summarized in Table III. In the next phase, we map theses 16 IPs into tiles of a 4×4 torus network randomly. Throughout the experiments, we considered an application-specific system-on-chip with 3 cycle router delay, 1 cycle wire delay and exponentially distributed packet size with average size of 16 flits. In addition,
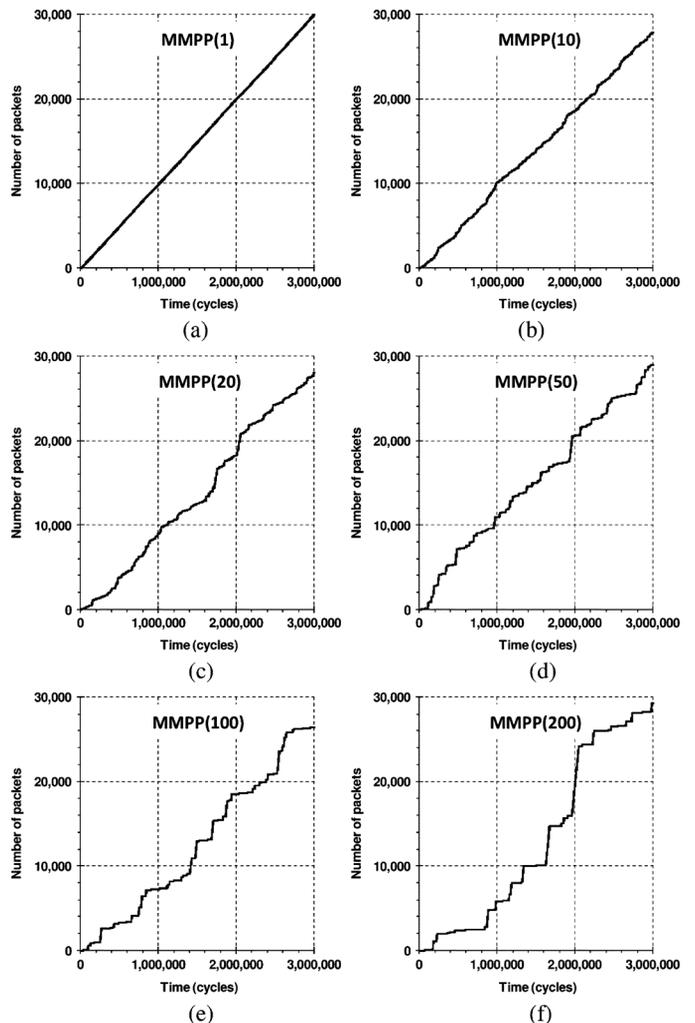
Fig. 10. Number of packets against time in the MMPP model for (a) $k = 1$ (Poisson model), (b) $k = 10$, (c) $k = 20$, (d) $k = 50$, (e) $k = 100$, (f) $k = 200$.

TABLE II
CV OF PACKET INTERARRIVAL TIME FOR DIFFERENT VALUES OF k

Traffic model	$k = \lambda_1/\lambda_0$	C_A
MMPP(1)	1	1.00
MMPP(10)	10	1.55
MMPP(20)	20	2.04
MMPP(50)	50	3.08
MMPP(100)	100	4.28
MMPP(200)	200	6.00

we supposed that input and output buffers of the routers have the capacity of 6 and 2 flits, respectively.

Latency of flows in this configuration is investigated in Fig. 11(a) and (b). Average latency of all packets generated by MMPP(10), MMPP(20), and MMPP(50) in different network throughput are compared in Fig. 11(a). Fig. 11(b) depicts the average latency of all flows generated by MMPP(50) when the network operates at 0.02 flits/cycle/node.

As can be seen again, the model has a fairly good degree of accuracy in comparison to the simulation results with average relative error of 4.7%. We also implement the proposed model in [13] and compare it with the PQ model. Fig. 11(a) also shows

(a)

(b)
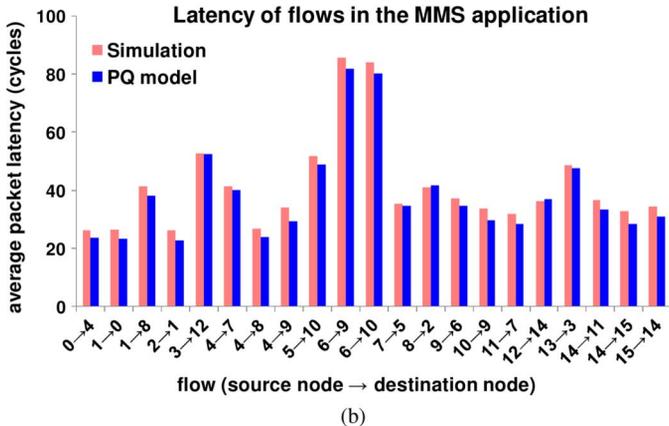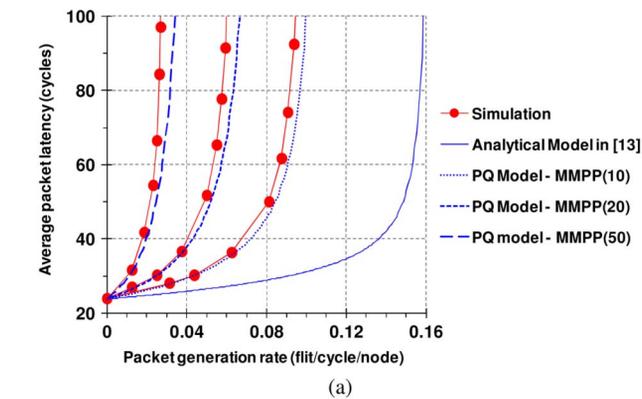
Fig. 11. (a) Average packet latency of all flows in case of bursty traffic. (b) The average packet latency of each flow predicted by the PQ model against simulation results.

(a)

(b)

Fig. 12. (a) Custom topology. (b) The average packet latency of all flows.
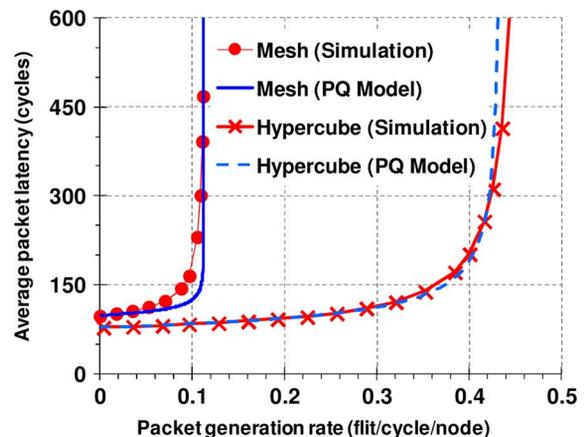
Fig. 13. Average packet latency for a 16×16 mesh network and an 8-D hypercube network with dimension-order routing.

TABLE III
MMS APPLICATION TRAFFIC REQUIREMENT [7]

src	dst	vol. (bytes)	src	dst	vol. (bytes)
ASIC1	ASIC2	25	DSP2	DSP1	20363
ASIC1	DSP8	25	DSP3	ASIC4	38016
ASIC2	ASIC3	764	DSP3	DSP6	7061
ASIC2	MEM2	640	DSP3	DSP5	7061
ASIC2	ASIC1	80	DSP4	DSP1	3672
ASIC3	DSP8	641	DSP4	CPU	197
ASIC3	DSP4	144	DSP5	DSP6	26924
ASIC4	DSP1	33848	DSP6	ASIC2	28248
ASIC4	CPU	197	DSP7	MEM2	7065
CPU	MEM1	38016	DSP8	DSP7	28265
CPU	MEM3	38016	DSP8	ASIC1	80
CPU	ASIC3	38016	MEM1	ASIC4	116873
DSP1	DSP2	33848	MEM1	CPU	75205
DSP1	CPU	20363	MEM2	ASIC3	7705
DSP2	ASIC2	33848	MEM3	CPU	75584

that using the model in [13] to design a system with bursty traffic may lead us to less trusted decisions.

C. Arbitrary Topology

To show the capability of PQ model to predict the average latency in an arbitrary network, we consider the topology shown in Fig. 12(a) with the uniform workload and 32 flits packets. We

used the CAR framework [10] to find the deadlock-free routes in this network. CAR constructs the channel dependency graph based on the network topology and application, and then deletes some edges from the channel dependency graph to guarantee the deadlock freedom. After that, CAR creates the routing space by finding all possible shortest paths for each flow. Finally, the simulated annealing heuristic is used to find congestion-aware routes. Fig. 12(b) reveals that the proposed analytical model predicts the average packet latency accurately in almost all traffic regions which are appropriate for network operation.

Furthermore, to assess the proposed model for large networks, we compare the PQ model and simulation results for 16×16 mesh network and 8-D hypercube network with 256 nodes. Dimension-order routing algorithms are used to route the data packets among IP cores. We choose the hypercube network for this experiment because it has totally different topological properties compared to the mesh network. Fig. 13 shows the comparison result when the packet length is 32 flits, input buffers of the routers have the capacity of 8 flits and there are no output buffers.

D. Execution Time Comparison

Finally, the execution time of the proposed analytical model and simulation are compared. We implement both the PQ model and the simulator in C++ and run on the same computer. The execution times of the PQ model and simulation for mesh networks with various sizes from 9 (3×3) to 400 (20×20) nodes
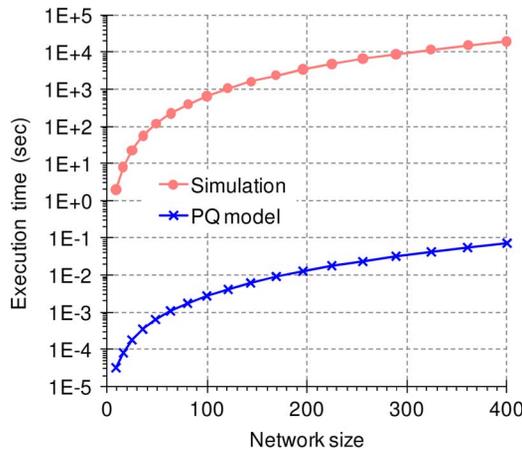
Fig. 14. Execution time comparison of the PQ model and simulation for different size of mesh networks.

are compared in Fig. 14. We simulate different size networks for 4 flits input and output buffers and 32 flits packets under uniform traffic. In such a traffic pattern, the number of flows are considerably increased with $O(n^2)$ where n is the number of nodes in the network.

As we mentioned previously in this section, a simulation run is divided into 10 batches. To reduce the simulation time we suppose that the simulator generates only three packets for each flow and averages the latency of these three packets to estimate the average latency of flows in each batch. Fig. 14 shows that the proposed approach is much faster than the simulation and the overall speed-up due to the analytical model is more than 60 000 for small networks and more than 260 000 for large networks. Also, the simulation execution time grows faster for larger buffer size, longer packet length, heavier traffic, and more bursty traffic, while the execution time of the analytical approach is constant for the same platform under different operation conditions. Furthermore, we observe that the model accuracy fluctuates randomly with the network size and does not confirm any specific trend.

VII. Conclusion and Future Work

Usually, system designers address the design problems by exploring the design space using detailed simulations. However, this approach has high run-time overhead and lacks of insights. Like in other disciplines of science and engineering, the use of analytical models can potentially address these limitations under certain assumptions. To this end, we propose the PQ model for predicting the communication performance of wormhole-switched NoC platforms. This queueing theory based model takes as input: 1) an application communication graph; 2) a topology graph; 3) a mapping vector; and 4) a routing matrix, and estimates some performance metrics of the system such as average packet latency and router blocking time. The proposed model is validated through simulation experiments, and we have shown that the proposed model achieves a good degree of accuracy ($<10\%$ error) making it a practical and useful evaluation tool that can be used by researchers in the field to gain insight into the performance behavior of the designed system. The model independency on network topology and workload

type makes it a robust tool to explore the huge design space of NoC-based systems.

In many applications such as real-time systems, the worst case execution time is of particular concern since it is important to know how much time might be needed in the worst case to guarantee that the task will always finish its jobs before the predetermined deadline. Therefore, we plan to advance this research by integrating the proposed average case model with an analytical worst case model. Finally, we would like to utilize the integrated performance model to find a near optimal solution for some design problems such as topology selection, module placement and buffer allocation problems in the network-based systems.

References

[1] J. H. Bahn and N. Bagherzadeh, "A generic traffic model for on-chip interconnection networks," in *Proc. Int. Workshop Netw.-on-Chip Arch. (NoCArc), Held in Conjunction With the IEEE/ACM Int. Symp. Microarch. (MICRO-41)*, 2008, pp. 22–29.

[2] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, Feb. 2005.

[3] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*, 2nd ed. New York: Wiley, 2006.

[4] J. Duato, C. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*. San Francisco, CA: Morgan Kaufmann, 2003.

[5] W. Fischer and K. Meier-Hellstern, "The Markov-modulated poisson process (MMPP) cookbook," *Perform. Eval.*, vol. 18, no. 2, pp. 149–171, 1993.

[6] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network delays and link capacities in application-specific wormhole nocs," *J. VLSI Design*, vol. 2007, 2007, Article ID 90941.

[7] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2919–2933, Dec. 2006.

[8] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Buffer optimization in network-on-chip through flow regulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1973–1986, Dec. 2010.

[9] S. H. Kang and D. K. Sung, "Two-state MMPP modelling of ATM superposed traffic streams based on the characterisation of correlated interarrival times," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, 1995, pp. 1422–1426.

[10] A. E. Kiasari, A. Jantsch, and Z. Lu, "A framework for designing congestion-aware deterministic routing," in *Proc. Int. Workshop Netw.-on-Chip Arch. (NoCArc), Held in Conjunction With IEEE/ACM Int. Symp. Microarch. (MICRO-43)*, 2010, pp. 45–50.

[11] A. E. Kiasari, H. Sarbazi-Azad, and S. Hessabi, "Caspian: A tunable performance model for multi-core systems," in *Euro-Par 2008 Parallel Processing*, E. Luque, T. Margalef, and D. Benitez, Eds. New York: Springer-Verlag, 2008, pp. 100–109, Lecture Notes in Computer Science.

[12] A. E. Kiasari, H. Sarbazi-Azad, and M. Ould-Khaoua, "An accurate mathematical performance model of adaptive routing in the star graph," *Future Generation Comput. Syst.*, vol. 24, no. 6, pp. 461–474, 2008.

[13] J. Kim and C. R. Das, "Hypercube communication delay with wormhole routing," *IEEE Trans. Comput.*, vol. 43, no. 7, pp. 806–814, Jul. 1994.

[14] L. Kleinrock, *Queueing Systems*. New York: Wiley, 1975, vol. 1.

[15] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design Test Comput.*, vol. 22, no. 5, pp. 434–442, Sep./Oct. 2005.

[16] U. Y. Ogras, P. Bogdan, and R. Marculescu, "An analytical approach for network-on-chip performance analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 2001–2013, Dec. 2010.

[17] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L. S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sep./Oct. 2007.

[18] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "Application specific routing algorithms for networks on chip," *IEEE Trans. Parall. Distrib. Syst.*, vol. 20, no. 3, pp. 316–330, Mar. 2009.

[19] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Trans. Comput.*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.

[20] K. Pawlikowski, "Steady-state simulation of queueing processes: A survey of problems and solutions," *ACM Comput. Surveys*, vol. 22, no. 2, pp. 123–170, 1990.

[21] V. Soteriou, H. Wang, and L.-S. Peh, "A statistical traffic model for on-chip interconnection networks," in *Proc. IEEE Int. Symp. Model., Anal., Simulation Comput. Telecommun. Syst.*, 2006, pp. 104–116.

[22] H. Takagi, "Queueing Analysis. vol. 1: Vacation and Priority Systems," Amsterdam, 1991.

[23] G. V. Varatkar and R. Marculescu, "On-chip traffic modeling and synthesis for MPEG-2 video applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 1, pp. 108–119, Jan. 2004.

Zhonghai Lu (M'05) received the B.Sc. degree in radio and electronics from Beijing Normal University, Beijing, China, in 1989, the M.Sc. degree in system-on-chip design and the Ph.D. degree in electronic and computer systems design from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 2002 and 2007, respectively.

From 1989 to 2000, he was an Engineer in the area of electronic and embedded systems. He is currently an Associate Professor with the Department of Electronic Systems, School for Information and Communication Technology, KTH. His research interests include network-on-chip/system-on-chip, many-core computing architectures, cyber-physical systems, performance analysis, and design automation. He has published about 100 papers in those areas.

Axel Jantsch (M'97) received the Dipl. Ing. and Dr. Tech. degrees from the Technical University of Vienna, Vienna, Austria, in 1988 and 1992, respectively.

He was with Siemens Austria, Vienna, Austria, as a System Validation Engineer from 1995 to 1997. Since 1997, he has been an Associate Professor with the Royal Institute of Technology (KTH), Kista, Stockholm, Sweden. Since 2000, has been a Docent, and since December 2002, a Full Professor of Electronic System Design with the Department of Electronic Systems. He has published over 200 papers in international conferences and journals, and one book in the areas of VLSI design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks on chip. At KTH, he is heading a number of research projects involving a total number of ten Ph.D. Students, in two main areas: system modeling and networks-on-chip.

Dr. Jantsch received the Alfred Schrödinger Scholarship from the Austrian Science Foundation while a Guest Researcher with KTH between 1993 and 1995. He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES+ISSS, SOC, NOCS, and others. He has been the TPC Chair of SSDL/FDL 2000, the TPC Co-Chair of CODES+ISSS 2004, the General Chair of CODES+ISSS 2005, and the TPC Co-Chair of NOCS 2009. From 2002 to 2007, he was a Subject Area Editor for the *Journal of System Architecture*.

Abbas Eslami Kiasari (S'11) received the B.Sc. degree in electrical engineering from the Ferdowsi University, Mashhad, Iran, in 2003, and the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2005. He is currently pursuing the Ph.D. degree in electronic systems under supervision of Prof. A. Jantsch from the Royal Institute of Technology (KTH), Stockholm, Sweden.

His research interests include design methodologies and performance analysis of network-based systems.

Mr. Kiasari is a member of the IEEE Computer Society.