Caspian: A Tunable Performance Model for Multi-core Systems

Abbas Eslami Kiasari[1,2], Hamid Sarbazi-Azad[2,1], and Shaahin Hessabi[2]

[1] IPM School of Computer Science, Tehran, Iran
[2] Sharif University of Technology, Tehran, Iran
kiasari@ipm.ir, {hessabi,azad}@sharif.edu

Abstract. Performance evaluation is an important engineering tool that provides valuable feedback on design choices in the implementation of multi-core systems such as parallel systems, multicomputers, and Systems-on-Chip (SoCs). The significant advantage of analytical models over simulation is that they can be used to obtain performance results for large systems under different configurations and working conditions which may not be feasible to study using simulation on conventional computers due to the excessive computation demands. We present Caspian[1], a novel analytic performance model, aimed to minimize prediction cost, while providing prediction accuracy. This is accomplished by using a G/G/1 priority queueing model which is used for arbitrary network topology with wormhole routing under arbitrary traffic pattern. The accuracy of this model is examined through extensive simulation results.

Keywords: Performance evaluation, Analytical model, Multi-core systems, G/G/1 queueing model.

1 Introduction

Multi-core system designers are constantly confronted with the challenge of designing high performance system while simultaneously meeting constraints such as communication latency, network throughput and design costs [4]. The problem of identifying multi-core system configurations is further exacerbated for the following reasons. First, multi-core systems are evolving into increasingly complex systems with a large number and type of components such as processors, memories, routers, and queues. As a consequence, designers must deal with a large architectural design space consisting of several interacting parameters. Furthermore, new workloads are composed of a large spectrum of programs with widely differing characteristics. System designers have addressed these problems in the past by exploring the design space using detailed simulations. However, this approach has high simulation costs due to the low speed of cycle-accurate simulators.

[1] The Caspian Sea is the largest enclosed body of water on Earth by area, variously classed as the world's largest lake or a full-fledged sea. It lies between the southern areas of the Russian Federation and northern Iran [Wikipedia].

E. Luque, T. Margalef, and D. Benítez (Eds.): Euro-Par 2008, LNCS 5168, pp. 100–109, 2008.

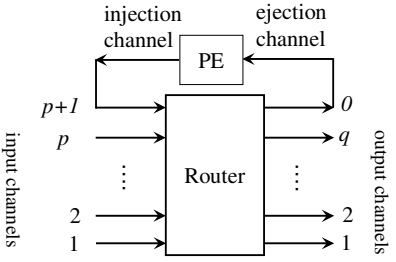


Fig. 1. A general structure for a node in a generic multi-core system

Performance models are frequently employed by multi-core system vendors in their design of future systems. Typically, engineers construct a performance model for one or two key applications, and then compare future technology options based on performance model projections. An analytical model that accurately characterizes the relationship between multi-core system performance and various implementation parameters would, in theory, obviate the need for detailed, expensive, and time consuming simulations. As an alternative to analytical models, in this research a tunable analytical modeling technique for multi-core system performance has been proposed and evaluated. The proposed approach, which is developed for wormhole flow control, provides buffer utilization, channels throughput, achievable throughput of the network, average waiting time for each channel, and average packet latency. These metrics can be conveniently used for design and optimization purposes, as well as obtaining quick performance estimates.

The main contribution of the work is a novel performance model, Caspian, for multi-core systems which can generalize the traditional delay models. Finally, the proposed model provides not only aggregate performance metrics, such as average latency and throughput, but also useful feedback about the network behavior. Hence, it can be invoked in any optimization loop for multi-core systems for fast and accurate performance estimations.

2 Performance Analysis

If the performance is measured in terms of average packet latency, then maximizing the performance is equivalent to minimizing the end-to-end packet latency. In this section, we derive an analytical performance model for multi-core systems using a G/G/1 [2] priority queueing model. It can be used for any arbitrary network topology with wormhole routing under any arbitrary traffic pattern.

2.1 Assumptions and Notations

We consider input buffered routers with $p+1$ input channels, $q+1$ output channels, and target wormhole flow control under deterministic routing algorithm. This form of

Table 1. Parameter notation

t_r	Spent time for packet routing decision (*cycles*)	*Platform specific parameters*
t_s	Delay of crossbar switch (*cycles*)	
t_w	Spent time for transmitting a flit between two adjacent router (*cycles*)	
M	The size of a packet (*flits*)	
L	The average packet latency (*cycles*)	
$\mathcal{R}$	Routing function	
R^N	The router located at address N	
PE^N	The processing element located at address N	
IC_i^N	The ith input channel of router R^N	
OC_j^N	The jth output channel of router R^N	
S_j^N	set of all source nodes for packets which pass through OC_j^N	
D_j^N	set of all destination nodes for packets which pass through OC_j^N	
$P^{S\to D}$	The probability of a packet generated by PE^S to be delivered to PE^D	*Application specific parameters*
α^N	The average packet injection rate of PE^N (*packets/cycle*)	
$\lambda_{i\to j}^N$	The average packet rate from IC_i^N to OC_j^N (*packets/cycle*)	
λ_j^N	The average packet rate to OC_j^N (*packets/cycle*) $\left(\lambda_j^N = \sum_i \lambda_{i\to j}^N\right)$	
μ_j^N	The average service rate of OC_j^N (*packets/cycle*)	
$\overline{b_j^N}$	The average service time of OC_j^N (*cycles*) $\left(\overline{b_j^N} = 1/\mu_j^N\right)$	
$\overline{\left(b_j^N\right)^2}$	The second moment of the service time of OC_j^N	
$C_{B_j^N}$	The CV (coefficient of variation) for service time of the OC_j^N	
$C_{A_{i\to j}^N}$	The 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	
$W_{i\to j}^N$	The average waiting time for a packet from IC_i^N to OC_j^N (*cycles*)	

routing results in a simpler router implementation and has been used in many practical systems [3]. So in this research we use the deterministic routing for deadlock free routing. The structure of a single node is depicted in Fig. 1. Each node contains a router and a Processing Element (PE) capable of generating and/or receiving packets.

Packets are injected into the network on input port p+1 (injection channel) and leave the network from output port 0 (ejection channel). Generally, each channel

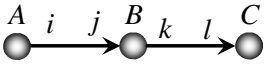


Fig. 2. A two hop packet from node A to node C

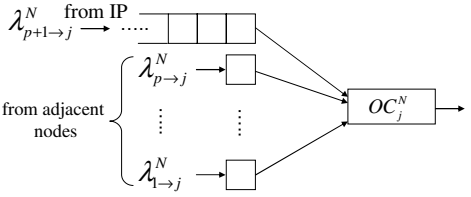


Fig. 3. Queueing model of a channel of an arbitrary topology

connects output port *j* of node *N* to input port *i* of node *M*. So, we denote this channel OC^N_j (*j*th output channel of router *N*) or IC^M_i (*i*th input channel of router *M*). Messages are broken into some packets of fixed length of *M* flits, as listed in Table 1 along with other parameters. 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 neighboring routers are t_r, t_s, and t_w, respectively. Also the transfer time of a flit across the injection and ejection channels are considered to be t_w.

Let $P^{S\to D}$ be the probability of packet transmission from the source node at router *S* (R^S) to the destination node at router *D* (R^D). Likewise, the traffic arrival rate of the header flits from IC^N_i to OC^N_j is given by $\lambda^N_{i\to j}$ packets/cycle. Also we assume that the packet injection process to the router R^N has a general distribution with mean value of α^N packets/cycle. The average packet latency (*L*) is used as the performance metric. Similar to previous works [1], [7], [8], 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, including the queuing time spent at the source. We also assume that the packets are consumed immediately once they reach their destination nodes.

2.2 Analytical Model

In Fig. 2 consider a packet which is generated in IP^A, and reaches its destination (IP^C) after traversing R^A, R^B, and R^C. The latency of this packet ($L^{A\to C}$) consists of two parts: the latency of header flit ($L_h^{A\to C}$) and the latency of body flits (L_b). In other words

$$L^{A\to C} = L_h^{A\to C} + L_b \tag{1}$$

$L_h^{A\to C}$ is the time from when the packet is created in IP^A, until when the header flit is reached to the IP^C, including the queueing time spent at the source node and intermediate nodes. In Fig. 2, $L_h^{A\to C}$ can be computed as

$$\begin{aligned} L_h^{A\to C} &= W^A_{p+1\to i} + t_w + t_r + t_s \\ &+ t_w + t_r + W^B_{j\to k} + t_s \\ &+ t_w + t_r + W^C_{l\to 0} + t_s + t_w \end{aligned} \tag{2}$$

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. 2 the channel between *B* and *C* can be addressed with OC_k^B or IC_l^C. Since the body flits follow the header flit in a pipelined fashion, L_b is given by

$$L_b = (M-1)(t_s + t_w) \tag{3}$$

The only unknown parameter for computing the latency is $W_{i\to j}^{N}$. This value can be calculated in a straightforward manner using a queuing model. The basic element in the model is a G/G/1 priority queue (the customer interarrival time and server's service time follow general distributions and queues have one server to provide the service). A router is primarily modeled based on nonpreemptive priority queuing system [11]

Now, let us consider, for instance, the *j*th output channel of R^N (OC_j^N). As can be seen in Fig. 3, this channel is modeled as a server in a priority queueing system with $p+1$ classes (IC_1^N to IC_{p+1}^N), with arrival rates $\lambda_{i\to j}^N$ $(1 \le i \le p+1)$, served by one server (OC_j^N) of service rate μ_j^N. Note that since all incoming packets are similar, the service times of all packets are equal. Both interarrival and service times are independent and identically distributed with arbitrary distributions.

Since the input channels (except injection channel) have one flit rooms, we should compute the average waiting time for the head of class *i*. Using a technique similar to that employed in literature for priority queues [2], [11] we can write

$$W_{i\to j}^{N} = \begin{cases} \overline{R_j^N} / \left(1-\rho_{i\to j}^N\right), & i = p+1, \\ \dfrac{1+\rho_{i+1\to j}^N - \sigma_{i+2\to j}^N}{1-\sigma_{i+1\to j}^N} W_{i+1\to j}^N, & 1 \le i \le p. \end{cases} \tag{4}$$

where $\overline{R_j^N}$ is the residual service time of OC_j^N seen by an incoming header flit and $\sigma_{i\to j}^N = \sum_{k=i}^{p+1} \rho_{i\to j}^N$. In a G/G/1 queueing system, $\overline{R_j^N}$ is approximated by [2]:

$$\overline{R_j^N} \approx \sum_{i=0}^{p} \rho_{i\to j}^N \frac{C_{A_{i\to j}^N}^2 + C_{B_j^N}^2}{2\mu_j^N} \tag{5}$$

Since we do not have enough insight about the first and second moments of interarrival time, we suppose that $C_{A_{i\to j}^N}$ is constant for all input channels in the network and equal to the coefficient of variation (CV) of the arrival process to network ($C_{A_{i\to j}^N} = C_A$). So, we can rewrite Eq. (5) as

$$\overline{R_j^N} \approx \frac{1}{2} \lambda_j^N \left(\overline{b_j^N}\right)^2 \left(C_A^2 + C_{B_j^N}^2\right) \tag{6}$$

Therefore, to compute $W_{i\to j}^{N}$ we must calculate the average arrival rate over OC_j^N (λ_j^N), and also first and second moments of the service time of OC_j^N. Assuming the network is not overloaded, the arrival rate over OC_j^N can be calculated using the following general equation

$$\lambda_j^N = \sum_{\forall S}\sum_{\forall D} \alpha^S \times P^{S\to D} \times \mathcal{R}\left(S \to D, OC_j^N\right) \tag{7}$$

In Eq. (7), the routing function $\mathcal{R}\left(S \to D, OC_j^N\right)$ equals 1 if the packet from PE^S to PE^D passes through OC_j^N; it equals 0 otherwise. Note that we assume a deterministic routing algorithm, thus the function of $\mathcal{R}\left(S \to D, OC_j^N\right)$ can be predetermined.

Although λ_j^N can be computed exactly for all topologies by Eq. (7), service time moments of the output channels cannot be computed in a direct manner by a general formula for any topology and any routing algorithm. To compute the moments of the service time of the output channels we first divide the channels into some groups based on their routing order and then an index is assigned to the groups opposite of the routing order, from ejection channel to injection channel. Then, we estimate the first two moments of the service time for the output channels. Determination of the channel service time moments starts at the ejection channel and works backward to the source of the packets. Therefore, the contention delay 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 must calculate the waiting time of all channels in group $k-1$. This approach is dependent to the topology and routing algorithm. Here we derive an analytical performance model for for e-cube routing [4] in a hypercube network. Due to the popularity of the hypercube network for multicomputer vendors [4], our analysis focuses on this topology but the modeling approach used here can be equally applied for other topologies after few changes in the model.

We consider a system which is composed of 2^n processing cores interconnected by an n dimensional hypercube (H_n). Packets are injected into the network on crossbar input port $n+1$ and leave on output port 0. A dimension-i channel connects output port i of a node to input port i of another node that differs only in the ith bit of its address. (In this paper, the least significant bit is bit 1). The $n+1$ input channels of each router are represented with $n+1$ (injection channel) and 1 to n (dimension 1 to n). Also these channels are assigned to priority classes from index $n+1$ (the highest priority) to 1 (the lowest priority), respectively. It is assumed that a static total ordering of input channel priorities exists. The packet arriving on the higher priority input channel will receive use of the crossbar output first. E-cube routing specifies that a packet sent between two nodes be first routed in the most significant dimension in which the addresses differ, then in the next most significant dimension in which they

differ, etc. Finally, it is fed to the destination node via ejection channel. By restricting the order in which the dimensions may be traversed, the possibility of cycles is removed, eliminating deadlock.

We divide the output channels of the hypercube network into $n+2$ groups based on their dimension numbers. Injection and ejection channels are located in group $n+1$ and 0, respectively, and physical channels are located in groups 1 to n. In the ejection channel (group 0) of R^N the header flit and body flits are accepted in t_w and L_b cycles, respectively. So, we can write $\overline{b_0^N} = t_w + L_b$ and since all packets have the same service time, there is not any variation in the service times. In other words $C_{B_0^N} = 0$. Now, we can determine the value of $W_{i\to 0}^N$ where $1 \le i \le n$.

The moments of service time for OC_j^N, $j > 0$, are obtained by tracing each of the 2^{j-1} paths from output j to the network outputs (ejection channels). Since each of these paths is not equally probable, the service time moments are weighted mean of each path service time. If S_j^N and D_j^N be the sets of all possible source and destination nodes for a packet which passes through OC_j^N, respectively, a passing packet through OC_j^N are destined to $M \in D_j^N$ with the probability of $\sum_{\forall S \in S_j^N} P^{S\to M} / \sum_{\forall S \in S_j^N} \sum_{\forall D \in D_j^N} P^{S\to D}$. The contention delays along each path are random variables; however each is only a fraction of the packet length for all packet rates that can be sustained on the hypercube.

For example, consider the output channel in dimension 2 of R^0 (OC_2^0) in an n-dimensional hypercube. One possible path from OC_2^0 to a network output is directly to ejection channel of the adjacent node R^2, for an average service time of $L_1 = t_w + \left(t_r + W_{2\to 0}^2 + t_s\right) + t_w + L_b$ where $W_{2\to 0}^2$ was already computed. The second path from OC_2^0 to a network output is through dimension 1 of R^2 and ejection channel of R^3, for an average service time of $L_2 = t_w + \left(t_r + W_{2\to 1}^2 + t_s\right) + t_w + \left(t_r + W_{1\to 0}^3 + t_s\right) + t_w + L_b$ where $W_{1\to 0}^3$ was already computed and $W_{2\to 1}^2$ can be computed with the same approach. So, the first and second moments of the service time can be estimated as

$$\overline{b_2^0} = \frac{\sum_{\forall S \in S_2^0} P^{S\to 2}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_1 + \frac{\sum_{\forall S \in S_2^0} P^{S\to 3}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_2$$

and

$$\overline{\left(b_2^0\right)^2} = \frac{\sum_{\forall S \in S_2^0} P^{S\to 2}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_1^2 + \frac{\sum_{\forall S \in S_2^0} P^{S\to 3}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_2^2$$

Now, we are able to calculate the coefficient of variation of the service time in OC_2^0, $C_{B_2^0}^2 = \overline{\left(b_2^0\right)^2} / \left(\overline{b_2^0}\right)^2 - 1$, and then the mean waiting time for OC_2^0 seen by other channels ($W_{i\to 2}^0$, $i > 2$) can be computed with Eq. (4). After computing the mean waiting time of all channels in group 2, the mean waiting time for other output channels (channels in groups 3, 4, …, $n+1$) can be calculated using the same approach. Now, we can calculate the packet latency between any two nodes in the network by Eq. (1). The average packet latency is the weighted mean of these latencies as

$$L = \sum_{S=0}^{2^n-1} \sum_{D=0}^{2^n-1} P^{S\to D} \times L^{S\to D} \tag{8}$$

3 Model Validation

The proposed analytical model has been validated through a discrete-event simulator that mimics the behavior of the network at the flit level. To achieve a high accuracy in the simulation results, we use the batch means method [9] for simulation output analysis. There are 10 batches and each batch includes up to 500,000 packets depending on the traffic injection rate 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 2% of the mean value.

For the destination address of each packet, we have considered the uniform and hotspot traffic patterns [10]. Packets are transferred to the local PE through the ejection channel as soon as they arrive at their destinations. Nodes generate packets independently of each other, and which follows a Poisson process. It means that the time between two successive packet generations in a PE is distributed exponentially, so for the first time we run the model program with $C_A = 1$. The Poisson model simplicity made it widely used in many performance analysis studies, and there are a large number of papers in very diverse application domains that are based on this stochastic assumption [5].

Using the proposed performance model, the CV of the interarrival time (C_A) can be adjusted to account for conformity between model and simulation results. Fig. 4 shows the flow chart used for tuning of the performance model. The model is run with various C_A until the predicted latency by the model is matched to its corresponding simulated value. The tuning procedure is run for the H_8 with $t_r = t_s = t_w = 1$ and C_A = 1.0498 is obtained for average packet generation rate of λ = 0.045 packets/cycle and packet length of M = 32 flits. Result of this tuned model has been presented in Fig. 5. The horizontal axis in the figure shows the traffic generation rate (packets/cycle) at each node while the vertical axis shows the mean packet latency (cycles). Tuned C_A is shown to give good quantitative agreement of model to simulation for a wide range of packet generation rate and packet length.

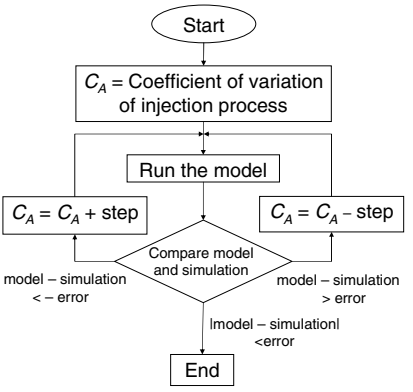


Fig. 4. Flow chart showing the strategy of the performance model tuning to simulation

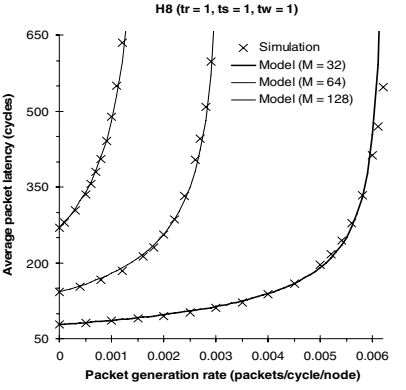
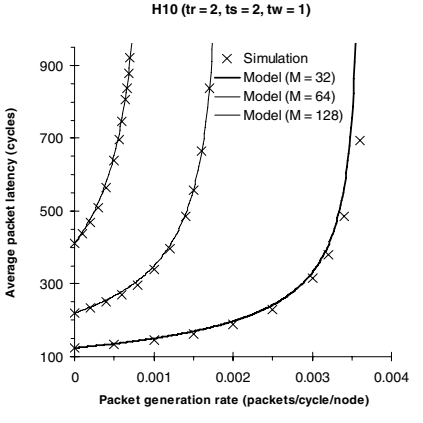
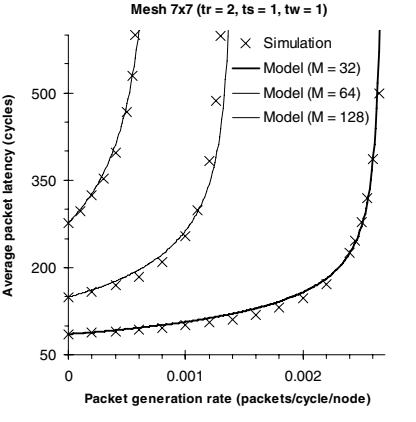


Fig. 5. The average packet latency predicted by the tuned model against simulation results for an H_8



(a)



(b)

Fig. 6. The average packet latency predicted by the tuned model against simulation results for (a) H_{10}, and (b) 7x7 mesh network with hotspot traffic

Fig. 6(a) illustrates average packet latency predicted by the tuned model, plotted against simulation results for H_{10} network. (C_A = 1.0035). Furthermore to verify the model accuracy for other topologies and non-uniform traffic pattern, we have modeled a 7x7 mesh network under hotspot traffic [10]. According to hotspot traffic pattern, there is a hot node in the network to receive the packets. Each node sends packets to the hot node with probability h, and sends packets to other nodes with probability $1- h$. In our experiments, we consider the node 24 in the center of the network as a hot node with hotspot rate $h = 0.1$. The comparison results is shown in Fig. 6(b) ($C_A = 0.8653$).

In [6], we have used Caspian and presented a performance-aware mapping algorithm which maps the IPs onto a generic System-on-Chip architecture such that the average communication delay is minimized.

4 Conclusions and Future Work

A novel methodology for predicting the communication performance of multi-core systems was proposed. The choice of the hypercube and mesh networks as the underlying interconnection architecture serves mostly as an example. In fact, our methodology can be modified to arbitrary topologies by adapting the analytical models accordingly to the target topology. Moreover, although we have evaluated our algorithm only for multi-core systems with dimension order routing, the approach is general enough to be applied to other deterministic and oblivious routing schemes.

We plan to advance this research in several directions. One possible direction is to extend this approach to multi-core systems with realistic workloads. Another important extension is to accommodate interconnection networks that support adaptive routing. The main challenge comes from the difficulty involved in the calculation of the arrival rate for each channel as multiple routing paths are possible in adaptive routing. Finally, we are extending this work for routers with finite size buffers.

References

1. Aljundi, A.C., Dekeyser, J., Kechadi, M.T., Scherson, I.D.: A Universal Performance Factor for Multi-criteria Evaluation of Multistage Interconnection Networks. Future Generation Computer Systems 22(7), 794–804 (2006)
2. Bolch, G., Greiner, S., De Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, 2nd edn. John Wiley and Sons, Chichester (2006)
3. Duato, J.: Why Commercial Multicomputers Do Not Use Adaptive Routing. IEEE Technical Committee on Computer Architecture Newsletter, pp. 20–22 (1994)
4. Duato, J., Yalamanchili, C., Ni, L.: Interconnection Networks: An Engineering Approach. IEEE Computer Society Press, Los Alamitos (2003)
5. Hu, J., Ogras, U.Y., Marculescu, R.: System-level Buffer Allocation for Application-Specific Networks-on-chip Router Design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(12), 2919–2933 (2006)
6. Kiasari, A.E., Hessabi, S., Sarbazi-Azad, H.: PERMAP: A Performance-Aware Mapping for Application-Specific SoCs. In: Proceedings of the Application-specific Systems, Architectures and Processors (2008)
7. Kiasari, A.E., Rahmati, D., Sarbazi-Azad, H., Hessabi, S.: A Markovian Performance Model for Networks-on-Chip. In: Proceedings of the Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 157–164 (2008)
8. Najafabadi, H.H., Sarbazi-Azad, H., Rajabzadeh, P.: Performance Modelling of Fully Adaptive Wormhole Routing in 2D Mesh-connected Multiprocessors. In: Proceedings of the International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 528–534 (2004)
9. Pawlikowski, K.: Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions. ACM Computing Surveys 22(2), 123–170 (1990)
10. Sarbazi-Azad, H., Ould-Khaoua, M., Mackenzie, L.M.: Analytical Modeling of Wormhole-Routed k-Ary n-Cubes in the Presence of Hot-Spot Traffic. IEEE Transaction on Computers 50(7), 623–634 (2001)
11. Takagi, H.: Queueing analysis. Vacation and Priority Systems, vol. 1. North-Holland, Amsterdam (1991)