

Mixture Models for System-Level Communication Analysis at Higher Levels of Abstraction

Ramon Mercado

Department of
Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
Email: rmercado@iastate.edu

Zhongbo Cao

Department of
Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
Email: jzbcdo@iastate.edu

Diane T. Rover

Department of
Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
Email: drover@iastate.edu

Abstract—System-level design (SLD) provides a solution to the challenge of increasing design complexity and time-to-market pressure in modern embedded system designs. In this paper, we propose a novel system-level approach to communication architecture modeling, which was not yet well addressed in existing SLD methodologies. In particular, we show how to develop statistical models for communication architectures. These new models are capable of capturing communication details at higher abstraction levels than previously possible. We demonstrate how to use the mean-square-error as a tool for developing these models, and show where to integrate these models in the design process.

I. INTRODUCTION

System complexity continues to grow according to the well-known Moore's law [1]. In contrast, designer productivity grows at a much lower rate. The International Technology Roadmap [2] indicates a productivity growth of only 21% (designed transistors/staff-month) in recent years. This disparity between system complexity and designer productivity is known as the productivity gap.

The growing productivity gap results in increasing non-recurrent engineering costs, and large time-to-market cycles. To deal with the productivity gap designers are turning to System-Level design [3]. System-level design tries to reduce the productivity gap by introducing new abstraction levels that allow designers to handle progressively more complex systems.

The higher abstraction levels, in system-level design, hide non-essential details from the designers, and allow for a coarser design space exploration. A common representation of the design space is an orthogonal composition of the computation and communication design alternatives. This orthogonal relation between computation and communication allows the designer to explore the computation architecture neglecting the communication effects on the system design.

The manner in which the design space is explored, i.e. a broader computation exploration followed by a narrow communication exploration, reveals a computation-centric design methodology. Computation-centric design is reasonable where higher computation power translates to higher system performance. However, with the proliferation of embedded communication systems (e.g. smart-phones, GPS, etc) and

multi-core systems, improvements in computation power are negated by communication delays. Today's processors are capable of processing large amounts of data much faster than the communication architecture can deliver this data. The system-level design community reacts to this paradigm change introducing new tools and methodologies to better integrate communication analysis into the early stages of the design space exploration.

For example, the works in [4], and [5] introduce SystemC [6] libraries to capture the characteristics of network-on-chip (NoC) architectures, as a collection of the services these architectures offer. On the other hand, [7] integrates the application behavior with the communication architecture details, generating a transaction based model that is used during architecture exploration. Unlike the previous works, our approach develops higher abstraction models based on the impact that communication architecture components have on the system performance. Explicitly, we base our analysis on the routing latency of NoC architectures, and develop statistical models capable of capturing the effect that different communication features have on this latency.

The rest of this paper is organized as follow: (1) The related work presents the current state of the art for communication exploration at the system-level. (2) The following section formally introduces the statistical model and shows, through an example, how to derive a statistical model from simulation data. (3) The final section discusses how the statistical model fits on the overall system-level picture, and how we plan to continue this research.

II. RELATED WORK

Refinement-based [8] methodologies dominate system-level design. The SpecC methodology [9] is probably the most widely known refinement-based system-level design methodology. The design starts at the specification abstraction level. The specification model represents the behavior of the application and includes the design constraints (e.g. maximum power, minimum performance, maximum area, etc). In the SpecC methodology the specification model is written in the SpecC language [10], other methods for defining the system specification include XML [11], and UML [12].

Through the architecture exploration the specification model is refined into an architecture model. This architecture model is the second abstraction level. At this level, the system architecture is modeled as a set of approximate-timed processing elements (PE).

The work in [13] show how the approximate-timed PE models are used for rapid design space exploration. This design space exploration depends on the performance estimates available at the architecture abstraction level. The research of [14] presents a method for performance estimation using weight-tables to represent the PE timing characteristics, and [15] shows how to improve the method in [14] by introducing more accurate low-level aware metrics. After the system architecture is decided, the next step is communication architecture exploration. The communication model is the third abstraction level. In SpecC, the communication model is the result of the communication synthesis, a process by which the architecture model is refined into the communication model. The final and lowest abstraction level is the implementation model. This model contains all the implementation details and is developed by refining and manufacturing the communication model.

As explained above, the SpecC design process flows from specification, to architecture, to communication, and finally to implementation. For communication intensive application, this refinement order may be a limiting factor. The problem lies in the architecture exploration. For communication intensive applications, the architecture exploration lacks the necessary communication architecture details to provide an accurate design space exploration. As communication intensive applications, and complex communication architectures start to dominate embedded system design, the limitation of computation-centric methodologies like SpecC becomes more apparent.

In the literature we found several approaches that address this lack of communication details during the architecture exploration. In general these approaches abstract the communication features of the target architecture, to include them earlier in the design space exploration. For example, [7] presents a model that (1) integrates the application behavior in terms of the communication transactions, and (2) includes cycle accurate figures for each transaction type on the target architecture.

In a more direct approach to communication architecture abstraction, [16] describes NoC architectures as a collection of communication resources and computation placeholders. The communication resources are further abstracted through the use of communication layers based on the OSI model. This layering approach to communication architecture abstraction has been adopted by others [4], [17], [18]. A similar approach is found in [5], where a C++ library is introduced to facilitate the modeling of layered interconnection networks.

In this paper we introduce a new modeling scheme capable of capturing low level communication architecture characteristics, and move these characteristics to the highest abstraction levels. Using random variables, we show how an statistical model is capable of capturing the timing behavior for a

network-on-chip (NoC) architecture. Further, we demonstrate how the mean square error (MSE) can be used to build statistical models from previous simulation results.

III. STATISTICAL MODEL

The goal of system-level design is to provide tools to measure the impact that low-level design decisions have on the system performance. In the case of the communication design alternatives, system performance is affected by the communication throughput and latency. The work presented in this paper shows how statistical models are developed, that capture the low-level latency relations and bring this information into the realm of system-level design.

A. Modeling Communication Components

The key modeling issue in system-level design is the relation between the design alternatives, performance metrics, and implementation cost. For latency a well known relation is the path latency [19]. For a path with n nodes and k channels, the path latency, T_n , is defined as:

$$T_n = \sum_{i=0}^n t_{ir} + \sum_{j=0}^k t_{jt} \quad (1)$$

Eq (1) has two components, routing time and traveling time. Routing time, t_r , refers to the time a packet resides in an intermediate routing node. Traveling time, t_t , is the time a package utilizes the communication channel between two nodes, and can be expressed in terms of the packet size and channel bandwidth.

Eq (1) shows how the design alternatives on topology, routing, and flow control have a direct effect on latency through their impact on routing and traveling time. For example, for a typical implementation of deterministic routing with wormhole flow control, topology and routing greatly affect routing time for the head flit at every routing node. It is shown in [20] that routing time may vary between 40 to 50 cycles, depending on the number of available output channels and buffers. In fact, most components of eq (1) are the result of one or more design alternatives. The path length, n , is as much influenced by topology as by the routing protocol. Routing time, t_r , is a function of the topology, routing, flow control, and even traffic. Traveling time, t_t , is heavily influenced by the packet size which is determined by the flow control.

Path latency provides a starting point for evaluating the relations between communication design alternatives and system performance. However, path latency, routing, and traveling time are still low level performance metrics. For these metrics to be meaningful at the system-level, it is necessary to provide models and methods to estimate the performance impact system-level design decisions have on these metrics.

Statistical models are widely used in the networking and communication research fields. For communication components, statistical models are most suitable for bridging the gap from the low-level implementation metrics to the system-level. Based on random variables, statistical models capture the behavior of communicating processes, and along with

statistical methods, can provide the tools necessary to move the communication exploration into the higher levels of abstraction.

Using eq (1) as the basis for this research, the challenge is how to properly introduce random variables into the path latency. The two candidates to model as random variables are, routing time (t_r) and traveling time (t_t). It is useful to see these components as they fit in the system. Fig. 1 shows the relation between the routing and traveling times, and the communication components on the system model.

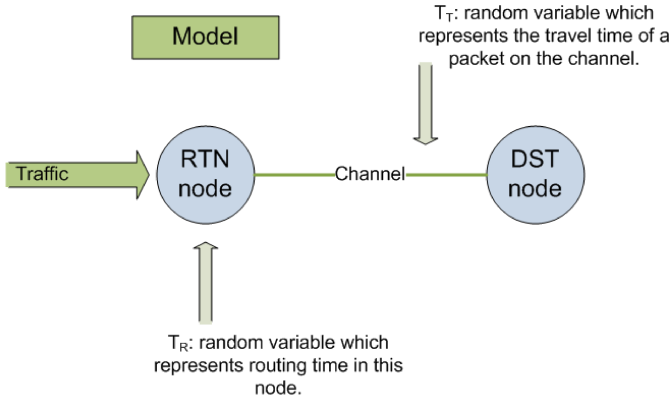


Fig. 1. Communication Components in System

From Fig. 1 it is clear that routing time captures the system design alternatives that define the node architecture. These alternatives include number of IN/OUT ports, buffer sizes, routing scheme implementation, flow control implementation, among others. On the other hand traveling time encapsulates the design alternatives related to the communication medium, including , but not limited to, bandwidth and packet size. Also present in Fig. 1 is the effect of dynamic phenomena in the system model.

Simple random variable models for routing and traveling times are not sufficient for system-level design. The new statistical models must capture both the static design alternatives and the dynamic phenomena due to load changes. The rest of this section presents the challenges and methods for developing the a statistical model for routing time.

B. Developing a Statistical Model

Modeling the routing time using an statistical model can provide the abstraction necessary to bring the communication information into the higher abstraction levels of system design. However, the process for developing such statistical model is not trivial. The initial challenge is to determine the behavior of routing time. To this end, we've set up the following system:

- 64 nodes arranged in an 8x8 torus,
- XY deterministic routing, and
- wormhole flow control

This system was developed in SystemC using a modified version of the NOXIM [21] simulator.

The characteristics of these systems were carefully selected. Deterministic routing is still the most commonly routing type

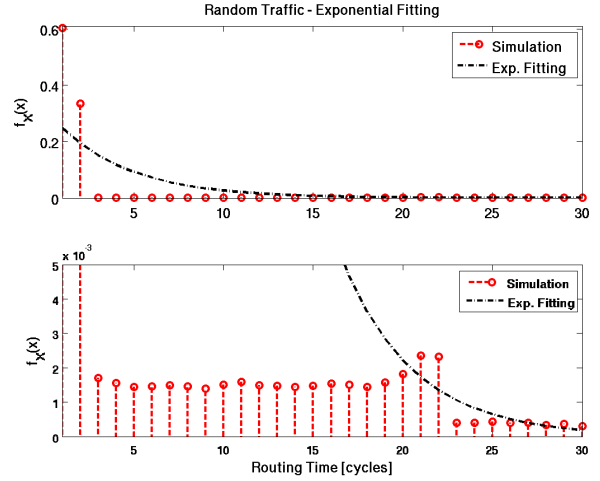


Fig. 2. Routing Time Distribution and Fitting for Random Traffic

used [22]–[25]. The advantages of deterministic routing are many, simple design and analysis, ease of implementation, low latency for well behaved loads, etc. Likewise, wormhole flow control dominates the NoC architectures [24], [26]–[28]. Flit-based flow control schemes, such as wormhole, are popular because they provides a level of data management that is ideal for the kind of transfers and resource constrains commonly found in today's NoC.

Two traffic loads are considered: random, and hotspot. Fig. 2 summarizes the simulation results for random traffic. In this simulation the system is subjected to random traffic for 10000 cycles. Each node injects random traffic made up of packets of exactly 10 flits.

Fig. 2 shows the empirical distribution of routing time, labeled Simulation. This distribution is the basis for the random model. As seen in Fig. 2, routing time in this system follows an exponential distribution. Several characteristics of this distribution are worthy to point out:

- 1) One and two cycles are the most frequent latencies.
- 2) About 97% of the area of Fig. 2 lay within the first 25 cycles.
- 3) The average simulation path latency is 28.68 cycles/flit.
- 4) The average simulation path length is 6 nodes.

Path latency is computed as the end-to-end delay of the packet, as it travels from source to destination.

The most common method for developing random models from empirical data is to fit the data to a known distribution. Table I shows the average routing time from simulation, along with the mean, standard distribution, and mean square error (MSE) for an exponential fitting of the simulation data for both random and hotspot traffic. This table confirms what can be seen graphically in Fig. 2, that is that an exponential fitting of the empirical distribution does not produce a good model.

The problem with the exponential fitting is that it can't model the behavior of the routing time for the first data points. The routing time distribution decreases very quickly during the

TABLE I
EXPONENTIAL FITTING FOR RANDOM TRAFFIC

Traffic	$Routing_{ave}$	Mean	Std. Dev.	MSE
Random	4.78	4.02	16.19	0.0821
Hotspot	30.99	127.52	16261.32	0.1632

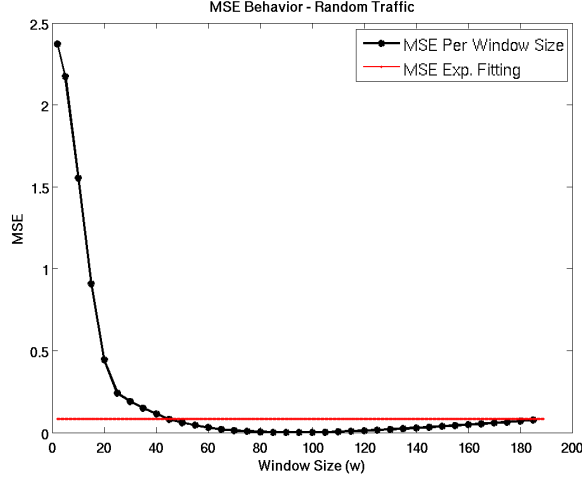


Fig. 3. MSE Behavior Across Window Sizes (Random)

first three data points, however it still has a long and narrow tail. These properties are difficult to match with an exponential fitting alone. The solution is to use a mixed distribution.

C. A Mixture Model

This section presents the steps we took to develop mixture models of our system under random and hotspot traffic. A mixture model [29] is of the form

$$f_X(x) = \sum_{i=0}^n a_i f_{Y_i}(x) \quad (2)$$

Where the density functions $f_{Y_i}(x)$ are the mixture components, and a_i are the mixture proportions. With the constraints that $0 \leq a_i \leq 1$ and $a_1 + a_2 + \dots + a_i = 1$. Upon further inspection of Fig. 2 we developed a the mixture model following a piecewise approach.

To develop the mixture model the routing time distribution of Fig. 2 is separated in two sets, or windows. For a simulation set of n data points, the first window contains the points from 0 to w ; and the second set has the rest of the data points, from $w + 1$ to n . These two sets become our mixture components. Based on this analysis our mixture model has two parameters:

- 1) the mixture proportion (a) from eq 2, and
- 2) window size (w).

The mixture proportions are determined using eq 3. $T_{r_{ave}}$ is the average routing time from simulation, and μ_w, μ_{w+1} are the means of the mixture components $f_{Y_0}(x)$ and $f_{Y_1}(x)$. Equation 3 holds only for distributions belonging to the exponential family, which is the case for routing time.

$$T_{r_{ave}} = \mu_w * a + \mu_{w+1} * (1 - a). \quad (3)$$

Window sizes, on the other hand, are not as clearly defined as the mixture proportions. We studied different window sizes and evaluated their mean square error (MSE). Fig. 3 shows the MSE as a function of the window size. The red horizontal line represents the MSE of the exponential fitting of Fig. 2. There are three distinct sections in Fig. 3: $w \leq 25$, $25 < w < 90$, and $w \geq 90$.

The first section, $w \leq 25$, is characterized by the largest MSE reduction rate. Knowing that 97% of the routing time distribution is stored on the first 25 cycles, it is expected that the MSE would decrease significantly faster as these data points are integrated into the $f_{Y_0}(x)$ mixture component. After this point, $w = 25$, MSE decreases at a lower rate because any more data moved into $f_{Y_0}(x)$ adds very little information. It is important to note that even with the 97% of the routing time distribution in $f_{Y_0}(x)$, this mixture model with $w = 25$ still has a worst MSE than the exponential fitting of Fig. 2.

On the second section, $45 < w < 90$, the MSE continues to decrease, but at a lower rate. MSE reaches a minimum at $w = 90$. At $w = 90$, 99% of the routing distribution is now in $f_{Y_0}(x)$. The resulting mixture model at $w = 90$ is

$$\begin{aligned} f_{T_R}(x) &= a_0 f_{Y_0}(x) + a_1 f_{Y_1}(x), \text{ where} \\ f_{Y_0}(x) &\sim \text{Exp}(\mu = 0.77), a_0 = 0.9513 \\ f_{Y_1}(x) &\sim \text{Exp}(\mu = 81.13), a_1 = 1 - a_0. \end{aligned}$$

Looking at the mixture proportions a_0 and a_1 , it is clear why this mixture model produces a better MSE. It is given a higher weight to the mixture component representing 99% of the routing time distribution, $a_0 f_{Y_0}(x)$.

The last section, $w \geq 90$, is characterized by an increase in MSE. This last section represents when the information on the tail of Fig. 2 starts to impact the mixture model. Finally, as the window size increases, $w \rightarrow \infty$, our mixture model becomes the same as an exponential fitting over the entire simulation set.

In summary, the MSE analysis above shows that using mix-

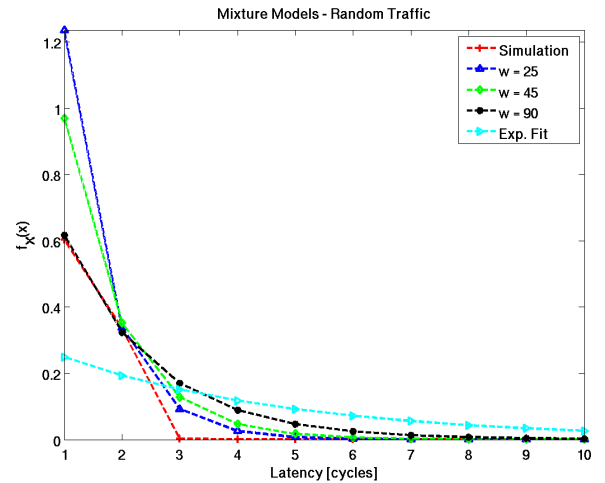


Fig. 4. Mixture Models for Random Traffic

ture models it is possible to find a better fit than simply fitting the simulation data to an exponential distribution. Furthermore, Fig. 3 shows that for all windows sizes $45 < w < 185$, the mixture model has a lower MSE than the exponential fit of Fig. 2. Graphically the mixture models for windows $w_{[25\ 45\ 90]}$ are shown in Fig. 4.

The same MSE analysis is used to derive the system model, when loaded with hotspot traffic. When using hotspot traffic the shape of the routing time distribution is very similar to the case with random traffic, with some key differences. (1) The tail of the routing distribution is much longer for hotspot. (2) The average routing time is 6 times that of the random traffic case. However, like with random traffic, the most meaningful information of the routing time distribution is located within the first 25 cycles. Figures 5 and 6 show the MSE curve and resulting mixture models for hotspot traffic, respectively.

IV. SYSTEM-LEVEL DESIGN WITH MIXTURE MODELS

The previous section demonstrated how to derive a mixture model from simulation. These mixture models capture the communication architecture characteristics as they behave in a loaded system. For the random traffic example, we showed that a window size of $w = 25$ produces a good mixture model. The resulting model is of the form:

$$\begin{aligned} f_X(x) &= a_0 f_{Y_0}(x) + a_1 f_{Y_1}(x), \text{ where} \\ f_{Y_0}(x) &\sim \text{Exp}(\mu = 0.77), a_0 = 0.9513 \\ f_{Y_1}(x) &\sim \text{Exp}(\mu = 81.13), a_1 = 1 - a_0 \end{aligned}$$

The mixture model $f_X(x)$ becomes the representation of the routing elements on future high level models that include the same communication architecture. Different mixture models are derived to represent different architecture alternatives. Much useful information may be gathered from the mixture model directly, e.g. the mean routing time, etc. However, the statistical mixture model becomes exceptionally useful when exposed to dynamic loads.

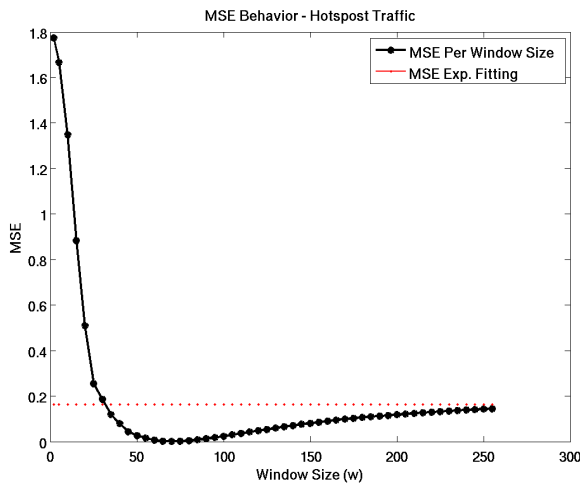


Fig. 5. MSE Behavior Across Window Sizes (Hotspot)

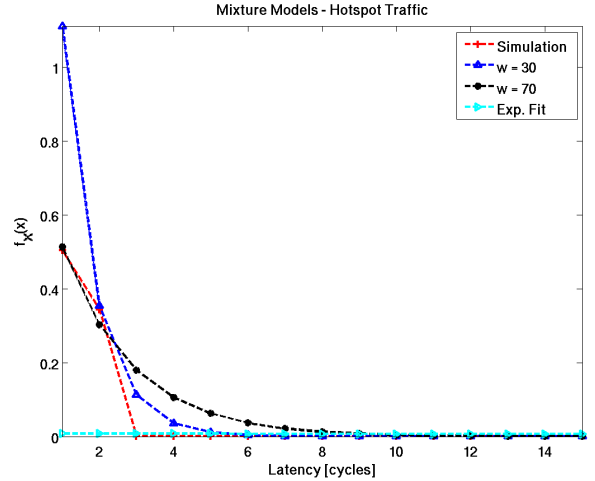


Fig. 6. Mixture Models for Hotspot Traffic

For instance, given the application communication characteristics, it is possible to use the mixture model to evaluate the collision behavior of a system at the highest abstraction level. Knowing that the routing time has a distribution $f_X(x)$, it is possible to combine the application communication characteristics with the mixture model to produce the distribution of the collision probability.

The collision distribution represents how the entire system reacts to the dynamic communication behavior of the application, and directly relates to the system performance. Dynamic analysis, such as the one for collision, are possible with mixture models because of the tools and methods available to statistical models.

V. FUTURE WORK

Using statistical models to represent communication components gives the designers access to a large assortment of tools from the statistics field. These tools provide several methods to combine the statistical model with the dynamic elements of the design. The next step in our research is to evaluate different statistical tools available for design space exploration.

Design space exploration is driven by performance estimation. Mixture models directly provide punctual statistics that are useful for space exploration, but these statistics alone are not enough for the in-depth exploration required by today's complex embedded systems. This research will continue to evaluate different statistical tools available to integrate the mixture model with the dynamic components of the architecture model, and to provide the communication performance estimates necessary for better architecture exploration.

VI. CONCLUSION

In this research we showed that mixture models are a viable tool for including communication information at higher levels of abstraction. We further illustrate a method for developing mixture models based on the mean square error. A typical system is modeled in SystemC and subjected to random and

hotspot traffic, and the data collected from the SystemC model was used to demonstrate how the mean square error is used to generate the mixture model.

ACKNOWLEDGMENT

This work was partially supported under NSF grant No. 0431924 and a GAANN grant from the U.S. Dept. of Education to the Information Infrastructure Institute at Iowa State.

REFERENCES

- [1] G. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan 1998.
- [2] D. Edenfeld, A. B. Kahng, M. Rodgers, and Y. Zorian, "2003 technology roadmap for semiconductors," *Computer*, vol. 37, no. 1, pp. 47–56, 2004.
- [3] O. Hammami and M. O. Cheema, "Graduate education to fight system level design productivity gap in soc design," in *MSE '07: Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 45–46.
- [4] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli, "Addressing the system-on-a-chip interconnect woes through communication-based design," in *DAC '01: Proceedings of the 38th conference on Design automation*. New York, NY, USA: ACM, 2001, pp. 667–672.
- [5] M. Coppola, S. Curaba, M. D. Grammatikakis, G. Maruccia, and F. Papariello, "Ocn: A network-on-chip modeling and simulation framework," in *DATE '04: Proceedings of the conference on Design, automation and test in Europe*. Washington, DC, USA: IEEE Computer Society, 2004, p. 30174.
- [6] P. R. Panda, "Systemc: a modeling platform supporting multiple design abstractions," in *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*. New York, NY, USA: ACM, 2001, pp. 75–80.
- [7] S. Pasricha, N. Dutt, and M. Ben-Romdhane, "Fast exploration of bus-based communication architectures at the ccatb abstraction," *Trans. on Embedded Computing Sys.*, vol. 7, no. 2, pp. 1–32, 2008.
- [8] S. Edwards, L. Lavagno, E. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: formal models, validation, and synthesis," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 366–390, Mar 1997.
- [9] D. D. Gajski, R. Dömer, J. Peng, and A. Gerstlauer, *System Design: A Practical Guide with Specc*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.
- [10] M. Fujita and H. Nakamura, "The standard specc language," in *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*. New York, NY, USA: ACM, 2001, pp. 81–86.
- [11] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "Mapping and configuration methods for multi-use-case networks on chips," in *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*. Piscataway, NJ, USA: IEEE Press, 2006, pp. 146–151.
- [12] W. Mueller, A. Rosti, S. Bocchio, E. Riccobene, P. Scandurra, W. Dehaene, and Y. Vanderperren, "Uml for esl design: basic principles, tools, and applications," in *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. New York, NY, USA: ACM, 2006, pp. 73–80.
- [13] L. Cai, A. Gerstlauer, and D. Gajski, "Retargetable profiling for rapid, early system-level design space exploration," in *DAC '04: Proceedings of the 41st annual conference on Design automation*. New York, NY, USA: ACM, 2004, pp. 281–286.
- [14] L. Cai, "Estimation and exploration automation of system level design," Ph.D. dissertation, 2004, chair-Gajski., Daniel.
- [15] J. P. Schneider, "Low-level estimation at high-levels of abstraction in system-level design," MS in Computer Engineering, Iowa State University, Ames, IA, USA, 2007.
- [16] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani, "A network on chip architecture and design methodology," in *ISVLSI '02: Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. Washington, DC, USA: IEEE Computer Society, 2002, p. 117.
- [17] L. Benini and G. D. Micheli, "Networks on chips: A new soc paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.
- [18] A. Gerstlauer, D. Shin, R. Dömer, and D. D. Gajski, "System-level communication modeling for network-on-chip synthesis," in *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*. New York, NY, USA: ACM, 2005, pp. 45–48.
- [19] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann Publishers, 2004.
- [20] H. Hashemi-Najafabadi, H. Sarbazi-Azad, and P. Rajabzadeh, "An accurate performance model of fully adaptive routing in wormhole-switched two-dimensional mesh multicomputers," *Microprocess. Microsyst.*, vol. 31, no. 7, pp. 445–455, 2007.
- [21] F. Fazzino, M. Palesi, and D. Patti. (2008) Noxim: Network-on-chip simulator. [Online]. Available: <http://noxim.sourceforge.net/>
- [22] S. Kundu and S. Chattopadhyay, "Mesh-of-tree deterministic routing for network-on-chip architecture," in *GLSVLSI '08: Proceedings of the 18th ACM Great Lakes symposium on VLSI*. New York, NY, USA: ACM, 2008, pp. 343–346.
- [23] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, and M. Pedram, "An empirical investigation of mesh and torus noc topologies under different routing algorithms and traffic models," in *DSD '07: Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 19–26.
- [24] A. K. Mishra, R. Iyer, R. Das, N. Vijaykrishnan, S. Eachempati, and C. R. Das, "A case for dynamic frequency tuning in on-chip networks," in *Micro-42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. New York, NY, USA: ACM, 2009, pp. 292–303.
- [25] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular noc architectures," in *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*. Washington, DC, USA: IEEE Computer Society, 2003, p. 10688.
- [26] F. A. Samman, T. Hollstein, and M. Glesner, "Multicast parallel pipeline router architecture for network-on-chip," in *DATE '08: Proceedings of the conference on Design, automation and test in Europe*. New York, NY, USA: ACM, 2008, pp. 1396–1401.
- [27] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware prioritization mechanisms for on-chip networks," in *Micro-42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. New York, NY, USA: ACM, 2009, pp. 280–291.
- [28] T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel, "Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures," in *DSD '07: Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 527–534.
- [29] A. Feldmann and W. Whitt, "Fitting mixtures of exponentials to long-tail distributions to analyze network performance models," in *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*. Washington, DC, USA: IEEE Computer Society, 1997, p. 1096.