

Analytical Modeling and Evaluation of Network-on-Chip Architectures

Suboh Suboh
The George Washington University
Washington DC. 20052, USA
suboh@gwu.edu

Jaafar Gaber
Universite de Technologie, UTBM
90010 Belfort, France
gaber@utbm.fr

Mohamed Bakhouya
Universite de Technologie, UTBM
90010 Belfort, France
bakhouya@gmail.com

Tarek El-Ghazawi
The George Washington University
Washington DC. 20052, USA
tarek@gwu.edu

ABSTRACT

Network-on-chip (NoC) architectures adopted for System-on-Chip (SoC) are characterized by different trade-offs between latency, throughput, communication load, energy consumption, and silicon area requirements. Evaluating NoC architectures is usually performed using simulations which provide little insight on how different design parameters affect the actual NoC performance metrics. Analytical models that allow rapid trade-off investigations of NoC parameters and accelerate the estimation of main metrics are required. In this paper, a Network Calculus-based methodology is presented to analyze and evaluate the performance metrics such as the latency and the cost metrics such as the energy consumption of NoC-based architectures. The WK-recursive on-chip interconnect is analyzed and results are compared against those produced using simulations. The values obtained by simulations and by analysis show the same increasing/decreasing trends in the same order of magnitude.

KEYWORDS: System-on-Chip, On-Chip Interconnect, Analytical Performance evaluation, Network calculus

1. INTRODUCTION

Emerging SoCs, such as those for mobile systems, are typically battery-powered systems and have to support a wide range of applications such as audio/video streaming. On-chip interconnect architectures, used in current SoCs to integrate hardware resources such as CPUs, DSPs, Memories, and I/O peripherals, are typically based on shared

medium or bus systems. For large SoCs, the bus-based schemes become restrictive because they are non-scalable and have higher overheads that adversely impact performance and energy consumption.

Network-on-Chip (NoC) has been recently proposed as an alternative to bus-based schemes to achieve high performance and scalability in SoC design [9, 14]. A key element in the performance and energy consumption in SoCs is the On-Chip Interconnect (OCI), which links all devices and assets (e.g., video processors, processing elements) that are created in the chip infrastructure. However, their increasing complexity makes their design extremely challenging. Different OCI-based architectures using packet-switching have been studied and adapted for SoCs [19, 21].

On-chip interconnect architectures adopted for SoCs are characterized by different trade-offs with regard to latency, throughput, communication load, energy consumption, and silicon area requirements. However, evaluating NoC architectures is usually performed using simulation [19, 23, 25]. Generally, the simulation is extremely slow for large systems and provides little insight on how different design parameters affect the actual NoC performance. Analytical models, however, allow a fast evaluation of large systems in early design process. Therefore, analytical models that allow rapid trade-off investigations of NoC parameters are required to accelerate the estimation of main metrics and ease the design process.

In this paper, we show how Network Calculus can be used to evaluate the performance metrics, energy consumptions, and area requirements of on-chip interconnects. As a case study, a detailed analysis and evaluation of WK-Recursive on-chip interconnect under different traffic loads is presented. It is worth noting that WK-Recursive topology

has received considerable attention in the parallel computing community because of its interesting properties such as extendability [10], i.e., for any specified degree, it can be expanded to an arbitrary level without reconfiguring the edges. In [20, 21], a detailed analysis has been done to compare the WK-recursive architecture with the 2D Mesh and Spidergon architectures in terms of several performance metrics such as throughput and for a variety of load scenarios. WK-recursive interconnect generally outperforms 2D Mesh and Spidergon topologies because of its regular and stable behavior at the nodes level. Other important metrics, such as energy consumption and area requirements, were investigated in [1].

The remainder of this paper is organized as follows. In section 2, we summarize the existing work on performance analysis methods proposed for evaluating on-chip interconnects. Section 3 presents the model for WK-Recursive on-chip interconnect computed using Network Calculus as well as the obtained results. Conclusions and future work are given in section 4.

2. RELATED WORK

Several studies, such as the one in [15], have demonstrated that there is a crucial need for system design tools and methodologies to compare NoC architectures for a given application. The authors pointed out that current design methodologies are unable to provide such frameworks and simulation methods because, despite their accuracy, they are very expensive and time consuming. Techniques and tools are required to extract an application communication requirement and to efficiently estimate its performance, its energy consumption, and its area requirements, under candidate communication architectures.

Recently, there has been a great deal of interest in the development of analytical performance models for NoC design [16]. Approaches proposed in the literature can be classified in four main categories: deterministic approaches, probabilistic approaches, physic-based approaches, and system theory-based approaches. In the first category, approaches are mainly based on graph theory used successfully in many software and computer engineering domains. For example, in [12], a model using a cyclo-static dataflow graph was used for buffer dimensioning for NoC applications. Deterministic approaches assume that the designer has a deep understanding of the pattern of communication among cores and switches.

Most of work to date, using probabilistic approaches, are based on queuing theory. For example, an analytical model using queuing theory was introduced in [17] to evaluate the traffic behavior in Spidergon NoC. Simulation results to

verify the model for message latency under different traffic rates and variable message lengths have been reported. In [13], a queuing-theory-based model for evaluating the average latency and energy consumption of on-chip interconnects was proposed. The results from the analytical model were validated with those obtained from a cycle-accurate simulator. Most queuing approaches consider incoming and outgoing traffics as probability distributions (e.g., Poisson traffic) and allow designers to perform a statistical analysis on the whole system in order to evaluate some network metrics such as average buffer occupancy and average buffer delay. However, NoC applications exhibit traffic patterns that are very different compared to Poisson model as in queuing theory [18, 23]. More precisely, the Poisson model fails to capture some important network characteristics like self-similarity or long-range dependence [16].

In [4], authors suggested statistical physics and information theory for NoC design and evaluation. Unlike stochastic approaches that make Markovian assumptions about the dynamical processes involved in network behavior, statistical physics can model the interactions among various components while taking into consideration the long-term memory effects. A quantum-like approach was proposed in [4] to model the information flow and buffers behavior in NoCs. The main concept in this model is that packets in the network move from one node to another in a manner that is similar to particles moving in a Bose gas and migrating between various energy levels as a result of temperature variations. Authors focused on buffers sizing issues, which is a major factor that affects the energy consumption and the silicon area requirements.

The fourth category uses system theory that is successfully applied to design electronic circuits. In particular, Network Calculus is inspired from this theory for modeling and evaluating main performance bounds (e.g., end-to-end delay) in networks such Internet [5, 8]. Based on shapes of the traffic flows (by analogy signals in system theory), designers are able to capture some dynamic features of the network. For example, in [2], we have presented a performance analysis methodology using Network Calculus to analyze and evaluate performance metrics of on-chip interconnects. Simulations are performed and results are compared with those produced from the Network Calculus based model in order to underline its usefulness for evaluating on-chip interconnects. In this paper, the Network Calculus is used to evaluate other performance metrics as well as the area and energy consumption. The WK-recursive on-chip interconnect is considered and evaluated under different traffic loads and results are reported to show the effectiveness of Network Calculus as a tool for NoC design and evaluation. Because of pages limitation, we refer readers to [2] for an overview on the basics of Network Calculus and to [5] for

more refined description.

3. OCIs EXPLORATION

In this section, the Network Calculus-based methodology for NoCs is presented. Fig. 1 shows a 16-nodes WK-recursive on-chip interconnect [1, 21], with application data flows generated as a case study to illustrate the methodology. NoC consists of IP cores, switches, and bidirectional links. Each IP core is attached to a local switch which connects the IP core to the neighboring switches via the on-chip interconnect. Data flows are represented by sequences of hops (e.g., $f_1 = (c_8, s_8, s_{12}, c_{12})$) from a source core to a destination core. As shown in this figure, five data flows are considered as follows: $f_1 = (c_8, s_8, s_{12}, c_{12})$, $f_2 = (c_8, s_8, s_3, s_2, s_5, c_5)$, $f_3 = (c_6, s_6, s_5, s_9, s_{13}, c_{13})$, $f_4 = (c_{11}, s_{11}, s_6, s_1, c_1)$, $f_5 = (c_{15}, s_{15}, s_{12}, c_{12})$. A flit in each flow, after it is injected by an IP core, moves to the local switch and is stored in the buffer waiting to be serviced and then sent to the next switch on the identified path. This process is repeated till the flit reaches the destination core. We assume that these data flows are already computed, i.e., the application tasks are already mapped to the IP cores.

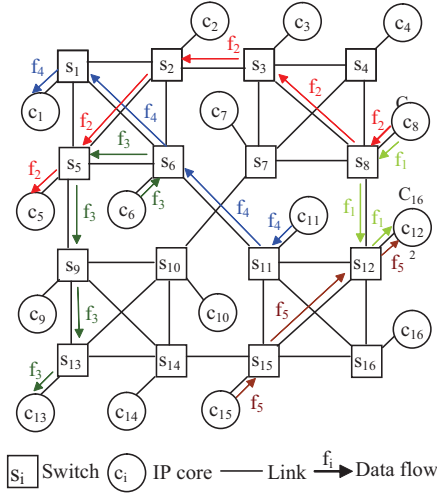


Figure 1. WK(4,2) On-chip Interconnect. f_i are Data Flows Representing the Application Workload

3.1. Network Calculus-based Model

The NoC architecture can be viewed as a distributed system composed of autonomous nodes which communicate by exchanging messages through an on-chip interconnect. The on-chip interconnect can be described as a graph $OCI(V, E)$ whose nodes $v \in V$ represent switches or cores and whose edges $\ell \in E$ represent the communication links between two neighboring nodes u and v . For

each node $v \in V$, r_v is the injection rate and for each link $\ell \in E$, R_ℓ denotes the guaranteed service rate or the link bandwidth. Similarly, an application can be represented by acyclic digraph, called Core Graph CG , where each $v \in V$ represents a core or switch and $\ell = (u, v) \in E$ is a communication flow edge having one attribute $\bar{\alpha}_\ell(t)$, the input arrival curve that represents the data flow sent through this link by a node u to a node v (node can be a core or a switch). Furthermore, at each time t , a switch s_i has an input flow $\bar{\alpha}_{s_i}(t)$ and each core c_i has an input flow $\bar{\alpha}_{c_i}(t)$.

Fig. 1 shows data flows f_1 , f_2 , f_3 , f_4 , and f_5 that can be represented by an acyclic digraph. The cores (c_6, c_8, c_{11}, c_{15}) are randomly selected to be sources. Cores (c_1, c_5, c_{12}, c_{13}) considered as sinks are selected according to the following communication locality principle in which 25% of the traffic takes place between neighboring cores and 75% of the traffic is uniformly distributed among the rest. We can see, in this traffic pattern, that c_8 is selected two times as a traffic source and c_{12} is selected two times as a traffic sink.

Having these data flows, we can express the input and output arrival curves, $\bar{\alpha}_{s_i}(t)$, $\bar{\alpha}_{c_i}(t)$, and $\bar{\alpha}_\ell(t)$ of each switch s_i , core c_i , and link ℓ respectively. The maximum data flow sent to a switch s_i is constrained by the arrival curve $\bar{\alpha}_{s_i}(t) = r_i t + b_i$, where b_i is the maximum burst size of the data flow and r_i is its average rate. Using this arrival curve, a node can send b_i bits at once, but without exceeding r_i bit/s over the long run. We consider that all cores sources have the same injection rate r and burst size b . Each switch also provides a guaranteed service constrained by the service curve $\beta_i(t) = R_i(t - T_i)^+$, where R_i denotes the guaranteed service rate and T_i is the maximum latency caused by the switch s_i . This service curve is called the rate-latency service curve in which data is delayed by a fixed time T_i and then routed out at a rate R_i . These two curves are widely used in evaluating systems [6, 11] and can be considered in our analysis. Using these curves, the arrival curves of each switch s_i , for example, can be calculated to obtain the following model:

$$\begin{aligned}
 \bar{\alpha}_{s_1}(t) &= rt + b + \frac{3}{2}rT & \bar{\alpha}_{s_9}(t) &= rt + b + \frac{13}{4}rT \\
 \bar{\alpha}_{s_2}(t) &= rt + b + 2rT & \bar{\alpha}_{s_{11}}(t) &= rt + b \\
 \bar{\alpha}_{s_3}(t) &= rt + b + rT & \bar{\alpha}_{s_{12}}(t) &= 2rt + 2b + 2rT \\
 \bar{\alpha}_{s_5}(t) &= 2rt + 2b + \frac{9}{2}rT & \bar{\alpha}_{s_{13}}(t) &= rt + b + \frac{17}{4}rT \\
 \bar{\alpha}_{s_6}(t) &= 2rt + 2b + rT & \bar{\alpha}_{s_{15}}(t) &= rt + b \\
 \bar{\alpha}_{s_8}(t) &= 2rt + 2b & &
 \end{aligned} \tag{1}$$

In the same manner, the arrival curve, $\bar{\alpha}_{c_i}(t)$, of each switch c_i , and the arrival curve, $\bar{\alpha}_\ell(t)$, of each link ℓ can be calculated. One of the main advantages of using Network Calculus is that the designer can model the data flows

of an application and their interactions (i.e., flows are dependent to each other) which are necessary for NoC design and evaluation. It's worth noting that SoC applications generally have broad computation and/or communications requirements. Understanding application communication patterns is critical for efficient use of SoC resources within a given set of constraints such as area, power and performance. Having a mathematical model to characterize the traffic pattern of a given application is an issue not described in the scope of this paper.

In the rest of this section, we will show how to evaluate the performance, the energy consumption, and the area requirements based on the OCI model describing the arrival curves of each switch, core, and link. We compare analytical and simulation results using the same traffic pattern to confirm the usefulness of Network Calculus for NoC design and evaluation. Simulations are conducted using a simulator developed in [22].

3.2. Performance Metrics

In this section, performance metrics, mainly the latency, throughput, and communication load, will be evaluated using the input and output arrival curves described by the system of equations (1).

3.2.1. Latency

Latency is defined as the time that elapses between injection start of the flits into the network at the source core and its arrival at the destination core. For a flit to reach the destination cores (processing elements), it must travel through a path consisting of a set of links and switches. Using Network Calculus, the latency \mathcal{L}_{s_i} in each switch s_i constrained by an arrival curve $r_i t + b_i$ can be calculated as follows:

$$\mathcal{L}_{s_i} = \frac{b_i}{R_i} + T_i \quad (2)$$

where R_i is the service bandwidth and T_i is the maximum latency of the service at switch s_i . Therefore, the average latency can be calculated based on equation 2. For example, since $\bar{\alpha}_{s_1}(t) = rt + b + \frac{3}{2}rT$, $\mathcal{L}_{s_1} = (\frac{3r}{2R} + 1)T + \frac{b}{R}$, if the injection rate is $r = 75Mbps$, $R = 200Mbps$, $b = 64bits$, and the flit size is $k = 8$ bytes, then $D_1 = 0.82\mu s$, where $T = k/R$. The average latency for the application can be also be calculated after calculating the latency \mathcal{L}_{s_i} of each switch s_i .

In the simulation, we consider that an application is represented as communicating parallel processes. Each process is linked with a traffic generator that injects flits according to the CBR (Constant Bit Rte) model at a deterministic

rate r which is varied from light traffic (25Mbps) to heavy traffic (100Mbps). The maximum service rate is fixed to 200Mbps in this simulation, b is fixed to be 64bits, and the flit size is 8 bytes. In this evaluation, data flows are considered dependent to each other and interact causing congestion in some switches which increases the latency at high injection rate.

Fig. 2 and Fig. 3 compares the average latency and the maximum latency of the WK(4,2) on-chip interconnect under different injection rate using Network Calculus and simulations. As shown in these figures, when increasing the injection rate, the network becomes more congested with heavy traffic and hence queues become full causing more flits to wait, therefore increasing the latency. We can see that the latency obtained using network calculus analysis is in the same order of magnitude as the latency obtained using simulations, i.e., both show a deviation of less than 18% on average.

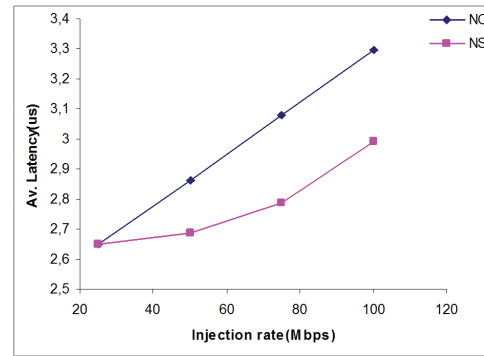


Figure 2. The Average Latency

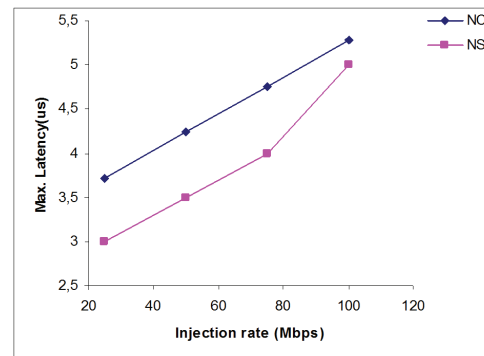


Figure 3. The Maximum Latency

3.2.2. Network Load

Communication load is a relative value of arrival rate vs. departure rate on all links. Let's consider $\mathcal{D}_r(t)$ as the maximum number of flits that can possibly, under ideal circumstances, be transmitted over all links at time t and $\mathcal{A}_r(t)$ is the actual number of flits that have arrived over all links at time t . The communication load $\mathcal{L}(t)$ can be defined as the ratio between the departure rate $\mathcal{D}_r(t)$ and the arrival rate $\mathcal{A}_r(t)$ as follows:

$$\mathcal{L}(t) = \frac{\mathcal{A}_r(t)}{\mathcal{D}_r(t)} = S_f \frac{\sum_{i=1}^{N_\ell} \bar{\alpha}_{\ell_i}(t)}{N_\ell R t} \quad (3)$$

where $\bar{\alpha}_{\ell_i}(t)$ is the number of flits arrived to the link ℓ_i , R is the bandwidth of each link ℓ_i , N_ℓ is the number of unidirectional links involved in transporting flits, and S_f is the size of each flit. We consider that all links have the same bandwidth. The results depicted in Fig. 4 show the variation of communication load under different traffic rates. The communication load obtained using Network Calculus analysis is in the same order of magnitude as the load obtained using simulations with a deviation of less than 25%.

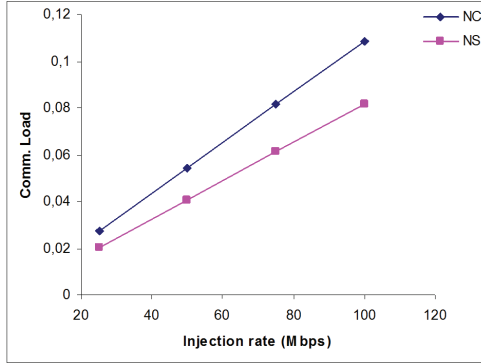


Figure 4. The Communication Load

3.2.3. Throughput

The throughput for each core represents how many bits arrive at that core per second. The aggregate throughput $\mathcal{T}(t)$ is the sum of throughput of each destination core c_i during the interval $[0, T]$. It can be calculated as follows:

$$\mathcal{T}(t) = \sum_{i=1}^{N_d} \bar{\alpha}_{c_i}(t) \quad (4)$$

where N_d is the number of cores selected as destinations (i.e., sinks), and $\bar{\alpha}_{c_i}(t)$ is the arrival curve that represents the accumulated number of flits arrived at the destination core c_i till time t . In the example depicted in Fig. 1, cores (c_1, c_5, c_{12}, c_{13}) are selected to be sinks. Using, the

OCI model described by the system of equations (1), the arrival curve $\bar{\alpha}_{c_i}(t)$ of each core c_i can be calculated. For example, $\bar{\alpha}_{c_1}(t) = rt + b + \frac{5}{2}rT$.

Fig. 5 shows the variation of aggregate throughput under different injection rates, which increases linearly when the injection rate increases because of the number of flits generated (see Fig. 6). The throughput obtained using Network Calculus analysis is in the same order of magnitude as the throughput obtained using simulations, i.e., both show a deviation of less than 5%.

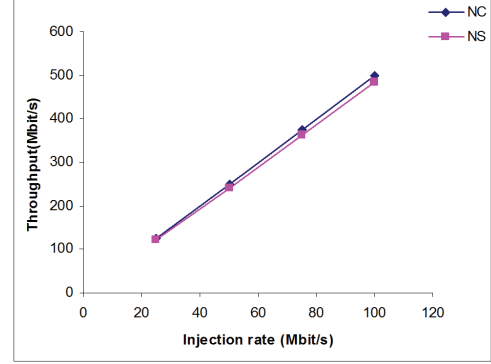


Figure 5. The Aggregated Throughput

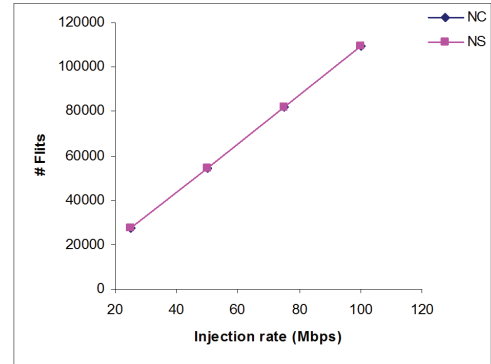


Figure 6. The Number of Flits Generated

3.3. Cost Metrics

In this section, cost metrics, mainly the average energy consumption and area overhead are evaluated using Network Calculus model and simulations.

3.3.1. Energy

The total energy can be decomposed into the energy consumed on the switches (traversal of input and output switches) and energy consumed per wire or link distance

traveled. The total energy $\mathcal{E}(t)$, can be calculated as follows:

$$\mathcal{E}(t) = \sum_{i=1}^{N_\ell} \mathcal{E}_{\ell_i}(t) + \sum_{j=1}^{N_s} \mathcal{E}_{s_j}(t) \quad (5)$$

where $\mathcal{E}_{\ell_i}(t)$ is the energy consumed, at time t , on the link ℓ_i , $\mathcal{E}_{s_j}(t)$ is the energy consumed inside the switch s_j , and N_ℓ and N_s are the number of links and switches respectively involved in transporting the application flows. Using Network Calculus arrival curves, the total energy consumption can be calculated as follows:

$$\mathcal{E}(t) = \sum_{i=1}^{N_\ell} \bar{\alpha}_{\ell_i}(t) E_\ell + \sum_{j=1}^{N_s} \bar{\alpha}_{s_j}(t) E_s \quad (6)$$

where E_ℓ is the average energy consumed during transporting one flit on a link ℓ , and E_s is the average energy consumed during buffering and routing operations inside each switch. The values of E_ℓ and E_s are constants for a given on-chip interconnect and depend mainly on the switch architecture and the link characteristic such as the width, the length, etc.

In this energy evaluation, we use the values already estimated in the energy model proposed in [24] in which the amount of energy required for a single bit to pass the switch is equal to $0.9776 pJ/bit$ and the amount of energy required for a single bit to cross a link ℓ is $(0.39 + 0.12L_\ell) pJ/bit$, where L_ℓ is the length of the link ℓ . In addition, we consider that the link between each core and its corresponding switch is of length 1mm. We consider that all links (horizontal or vertical) between neighboring switches are of length 2mm. For example, as shown in Fig. 7, WK(4,2) has 16 links of length 1mm, 20 links of length 2mm, and 10 links of length 4mm. However, only 5 links of length 1mm, 5 of length 2mm, and 5 of length 4mm are involved in transporting flits.

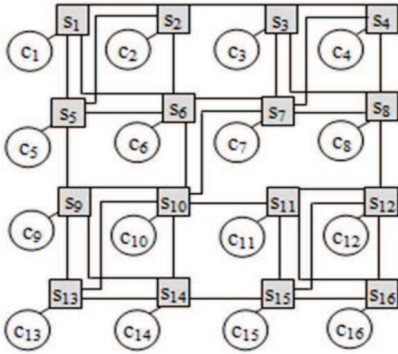


Figure 7. The layout of WK(4,2) On-chip Interconnect Considered in the Evaluation

Fig. 8 shows the energy consumption using Network Calculus and simulations. This figure shows that the energy consumption increases linearly when the injection rate increases. This increase can be explained by the big number of flits generated as the injection rate increases (see Fig. 6). We can see that the energy obtained using Network Calculus analysis is in the same order of magnitude as the energy obtained using simulations, i.e., the difference between simulation and analysis is about 21%.

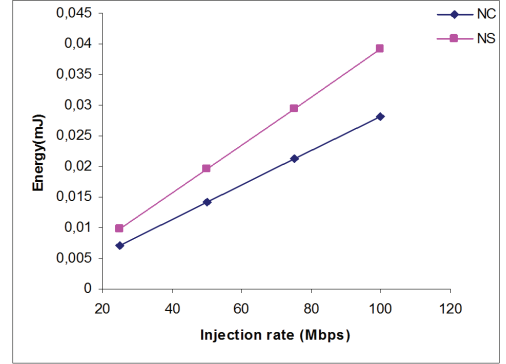


Figure 8. The Energy Consumption

3.3.2. Area

In NoC, there are three sources of area overhead, switches, cores, and links. Switches have two main components: the buffers to temporally store flits and logic to implement the routing algorithm. Area overhead of links depends on their lengths inside the chip [3]. The total area value can be calculated as follows:

$$\mathcal{A} = \sum_{i=1}^{N_s} \mathcal{A}_s(i) + \sum_{j=1}^{N_c} \mathcal{A}_c(j) + \sum_{k=1}^{N_\ell} \mathcal{A}_\ell(k) \quad (7)$$

where N_s is the number of switches, N_c is the number of IP cores, N_ℓ is the number of bidirectional links, $\mathcal{A}_s(i)$ and $\mathcal{A}_c(j)$, and $\mathcal{A}_\ell(k)$ is the area requirement for the switch i , core j and link k respectively. The average on-chip interconnect area \mathcal{A}_v will be determined by the average link area \mathcal{A}_ℓ , the average switch area \mathcal{A}_s , and the average IP core area \mathcal{A}_c . We consider the average since the resources (e.g., DSP, FPGA, Memory) are heterogeneous, the length of links are different, and the size of switches depends on their emplacement in the on-chip interconnect (e.g. degree). We use the architectures' layout depicted in Fig. 7 to determine these values in particular \mathcal{A}_s and \mathcal{A}_ℓ . So the average area \mathcal{A} can be derived from equation (7) as follows:

$$\mathcal{A}_v = N_s(R_s + a_s d_g S_f B_s) + N_c \mathcal{A}_c + a_\ell N_\ell L_\ell \quad (8)$$

where B_s is the average buffer size, a_s is the area required for one byte, S_f is the flits' size in bytes, a_ℓ is the area required for a link with length L_ℓ , R_s is other switch silicon area such as routing table and logic to implement the routing algorithm, and d_g is the average degree of the on-chip interconnect, which represents the average number of buffers inside the switch.

It was demonstrated in many papers, such as [3, 7], that a dominant part of the NoC area is due to the buffer sizes. To calculate the average buffer size B_s , we have to calculate the buffer size B_{s_i} of each switch s_i as follows. As described in (1), each switch s_i is constrained by an arrival curve in the form $\bar{\alpha}_{s_i}(t) = r_i t + b_i$ and provides a guaranteed service curve $\beta_i(t) = R_i(t - T_i)^+$ to each flow. Therefore, B_{s_i} can be calculated as follows:

$$B_{s_i} = b_i + r_i T_i \quad (9)$$

where r_i is the core injection rate and T_i is the maximum latency of the service at switch s_i . For example, since $\bar{\alpha}_{s_1}(t) = r t + b + \frac{3}{2} r T$, $B_{s_1} = \frac{5}{2} r T + b$, if the injection rate is $r = 75 \text{ Mbps}$, $R = 200 \text{ Mbps}$, $b = 64 \text{ bits}$, and the flit size is $k = 8 \text{ bytes}$, then $B_{s_6} = 16 \text{ bytes}$, where $T = k/R$.

Fig. 9 shows the area requirements (in mm^2) required for zero flits drop under different injection rates. In this evaluation, we consider that the area required to store the routing table and other related area are considered constant, $R_s = 1 \text{ mm}^2$, and $a_s = 0.005 \text{ mm}^2$, $a_\ell = 0.02 \text{ mm}^2$, $A_c = 2 \text{ mm}^2$, $R_s = 1 \text{ mm}^2$. We also consider that the chip size is of $20 \text{ mm} \times 20 \text{ mm}$. The value of L_ℓ is calculated based on the architectures layout shown in Fig. 7. As shown in Fig. 9, when injection rate increases the area requirements are also increased because the network becomes more congested with heavy traffic and so more space is needed to absorb differences in speed and burstiness between the IP cores (see Fig. 10). This figure illustrates the buffer size versus the injection rate and demonstrates that as the injection rate increases more space is needed to avoid flits being dropped.

4. CONCLUSIONS AND PERSPECTIVES

In this paper, a Network Calculus-based methodology was presented to evaluate on-chip interconnects in terms of performance and cost metrics (i.e., energy consumption and area requirements) based on a given traffic pattern. The results obtained using this methodology for WK-recursive on-chip interconnect are in the same order of magnitude as those obtained by simulations. Further work concerns the scalability issue by the development of a design space exploration software tool that will be built around Network Calculus and integrated with a simulation and exper-

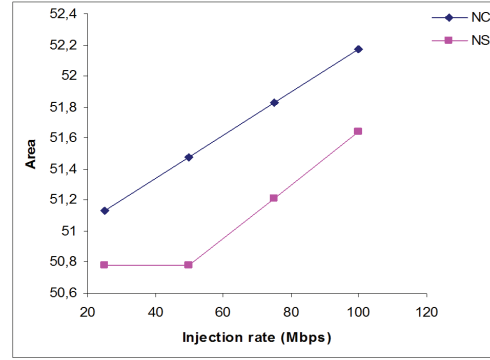


Figure 9. The Area Requirement (mm^2)

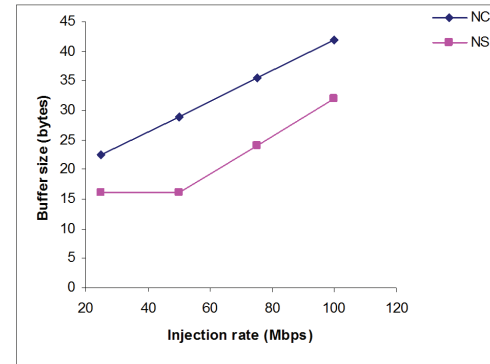


Figure 10. The Average Buffer Size

imental environment. The analytical model allows designers to quickly analyze the impact of various application-specific communication patterns on the overall system's performance. The time consuming simulation can only be used at later stages of the design and after the design space is reduced to a few configurations.

REFERENCES

- [1] M. Bakhouya, "Evaluating the energy consumption and the silicon area of on-chip interconnect architectures," *Journal of Systems Architecture*, Vol. 55, No. 7-9, pp. 387395, 2009.
- [2] M. Bakhouya and S. Suboh and J. Gaber and T. El-Ghazawi, "Analytical modeling and evaluation of on-chip interconnects using network calculus," *NoCS Proc.*, pp. 7479, 2009.
- [3] A. Balkan and G. Qu and U. Vishkin, "A mesh-of-trees interconnection network for single-chip parallel processing," *ASAP Proc.*, 2006.
- [4] P. Bogdan and R. Marculescu, "Quantum-like effects in network-on-chip buffers behavior," In *Proceedings of the*

- 44th Design Automation Conference (DAC), pp. 266-267, 2007.
- [5] J.-Y. L. Boudec and P. Thiran, "Network calculus: A theory of deterministic queuing systems for the internet," LNCS 2050, 2001.
- [6] A. Bouillard and B. Gaujal and S. Lagrange and E. Thierry, "Optimal routing for end-to-end guarantees using network calculus, performance evaluation," *Performance Evaluation*, Vol. 64, No. 11-12, pp. 883-906, 2008.
- [7] M. Coenen and S. Murali and A. Ruadulescu and K. Goossens and G. D. Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," CODES+ISSS Proc., 2006.
- [8] R. L. Cruz, "A calculus for network delay, part II: Network analysis," *IEEE Tran. on Information Theory*, Vol. 37, No. 1, pp. 132-141, 1991.
- [9] W. J. Dally and B. Towles, "Route packets, not wires: On chip interconnection networks," DAC Proc., pp. 683-689, 2001.
- [10] J.-F. Fang and Y.-R. Wang and H.-L. Huang, "The m-Pancycle-Connectivity of a WK-Recursive Network," *Inf. Sci.*, Vol. 177, No. 24, pp. 5611-5619, 2007.
- [11] V. Firoiu and J.-Y. L. Boudec and D. Towsley and Z.-L. Zhang, "Theories and models for internet quality of service," *Proceedings of the IEEE*, Vol. 99, No. 9, pp. 1565-1591, 2002.
- [12] A. Hansson and M. Wiggers and A. Moonen and K. Goossens and M. Bekooij, "Applying dataflow analysis to dimension buffers for guaranteed performance in networks on chip," NOCS Proc., pp. 211-212, 2008.
- [13] H. J. Kim and D. Park and C. Nicopoulos and V. Narayanan and C. Das, "Design and analysis of an NoC architecture from performance, reliability and energy perspective," ACM SANCS Proc., pp. 173-182, 2005.
- [14] S. Kumar and A. Jantsch and J.-P. Soininen and M. Forsell and M. Millberg and J. Berg and K. Tiensyrj and A. Hemani, "A network on chip architecture and design methodology," Proc. Int'l Symp. VLSI (ISVLSI), pp. 117-124, 2002.
- [15] K. Lahiri and S. Dey and A. Raghunathan, "Evaluation of the traffic-performance characteristics of system-on-chip communication architectures," VLSI Design Proc., pp. 29, 2001.
- [16] R. Marculescu and P. Bogdan, "The chip is the network: Toward a science of network-on-chip design," *Foundations and Trends in Electronic Design Automation*, Vol. 2, No. 4, pp. 371-461, 2007.
- [17] M. Moadeli and A. Shahrabi and W. Vanderbauwhede and M. Ould-Khaoua, "An analytical performance model for the spidergon NoC," 21st AINA Proc., pp. 1014-1021, 2007.
- [18] U. Y. Ogras and J. Hu and R. Marculescu, "Key research problems in NoC design: A holistic perspective," CODES+ISSS Proc., 2005.
- [19] P. P. Pande and C. Grecu and M. Jones and A. Ivanov and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Trans. on Computer*, Vol. 54, No. 8, pp. 1025-1040, 2005.
- [20] S. Suboh and M. Bakhouya and T. El-Ghazawi, "Simulation and evaluation of on-chip interconnect architectures: 2d mesh, spidergon, and wk-recursive networks," NoCS Proc., pp. 205-206, 2008.
- [21] S. Suboh and M. Bakhouya and J. Gaber and T. El-Ghazawi, "An interconnection architecture for network-on-chip systems," *Telecom. Sys. Journal*, Vol. 37, No. 1-3, pp. 137-144, 2008.
- [22] Y. R. Sun and S. Kumar and A. Jantsch, "Simulation and evaluation of a network on chip architecture using ns2," The IEEE NorChip Proc., 2002.
- [23] G. Varatkar and R. Marculescu, "Traffic analysis for on-chip networks design of multimedia applications," DAC Proc., pp. 510-517, 2002.
- [24] P. Wolkotte and G. Smit and N. Kavaldjiev and J. Becker and J. Becker, "Energy model of networks-on-chip and a bus," Proc. of System-on-Chip, pp. 82-85, 2005.
- [25] J. Xu and W. Wolf and J. Henkel and S. chakradhar, "A design methodology for application-specific networks-on-chip," *ACM Transactions on Embedded Computing Systems*, Vol. 5, No. 2, pp. 263-280, 2006.