Performance Evaluation and Design Tradeoffs of On-Chip Interconnect Architectures

Mohmed Bakhouya, Suboh Suboh, Jaafar Gaber, Tarek El-Ghazawi, Smail Niar

Performance Evaluation and Design Tradeoffs of On-Chip Interconnect Architectures

M. Bakhouya, S. Suboh, J. Gaber, T. El-Ghazawi, S. Niar

PII: S1569-190X(10)00220-0
DOI: 10.1016/j.simpat.2010.10.008
Reference: SIMPAT 1023

To appear in: *Simulation Modeling Practices and Theory*

Received Date: 2 July 2010
Revised Date: 8 October 2010
Accepted Date: 14 October 2010

Please cite this article as: M. Bakhouya, S. Suboh, J. Gaber, T. El-Ghazawi, S. Niar, Performance Evaluation and Design Tradeoffs of On-Chip Interconnect Architectures, *Simulation Modeling Practices and Theory* (2010), doi: 10.1016/j.simpat.2010.10.008

Performance Evaluation and Design Tradeoffs of On-Chip Interconnect Architectures

M. Bakhouya[1], S. Suboh[2], J. Gaber[1], T. El-Ghazawi[2], S. Niar[3]

[1] *Universite de Technologie de Belfort/Montbeliard*
Rue Thierry Mieg, 90010 Belfort Cedex, France
{mohamed.bakhouya,gaber}@utbm.fr
[2] *The George Washington University*
Washington DC. 20052, USA
{suboh,tarek}@gwu.edu
[3] *Universite de Valenciennes et du Hainaut-Cambresis*
59313 VALENCIENNES Cedex 9, France
smail.niar@univ-valenciennes.fr

Abstract

Network-on-Chip (NoC) has been proposed as an alternative to bus-based schemes to achieve high performance and scalability in System-on-Chip (SoC) design. Performance analysis and evaluation of on-chip interconnect architectures are widely based on simulations, which become computationally expensive, especially for large-scale NoCs. In this paper, a Network Calculus-based methodology is presented to analyze and evaluate the performance and cost metrics, such as latency and energy consumption. The 2D Mesh, Spidergong, and WK-recursive on-chip interconnect architectures are analyzed using this methodology and results are compared with those produced using simulations. The values obtained by simulations and by analysis show similar trends in the same order of magnitude. Furthermore, WK outperforms the other on-chip interconnects in all considered metrics.

Key words: Network-on-Chip, On-chip interconnect, Analytical modeling and evaluation, Design Tradeoffs, Network calculus

1. Introduction

System-On-chip (SoC) has recently emerged as a key technology behind most embedded and smart miniaturized systems to provide high flexibility

and better performance. These systems must provide high-performance while meeting system requirements, such as a low energy consumption and small area. For example, future mobile communication terminals should support many applications, which range from web browsing/navigation, to real-time multimedia applications such as audio and video communication. Therefore, the design of these systems should be highly flexible, adaptable, and meet stringent time-to-market constraints, while providing high-performance and lower energy consumption.

A key element in the performance and energy consumption in SoCs is the On-Chip Interconnect (OCI), which allows different SoC components to communicate efficiently. Network-on-chip has been proposed as an alternative to bus-based schemes to achieve high performance and scalability in SoC design. Different OCI-based architectures using packet-switching have been recently studied and adapted for SoCs. Examples of these architectures are Fat-Tree (FT)[1], 2D mesh[2], Ring[3], Butterfly-Fat Tree (BFT)[4], Torus[5], Spidergon[6], Octagon[7], WK-Recursive[8,9]. However, their increasing complexity makes their design extremely challenging. Furthermore, understanding and studying traffic generated between components and traverse the OCI is a crucial task[10]. Therefore, it is useful to perform a traffic analysis in early stages of the design process, such that the designer can select appropriate parameters for the on-chip interconnect architecture. Indeed, the selection of the on-chip interconnect architecture, based on traffic patterns that an application specific SoC generates, allows designers to detect and locate network contentions and bottlenecks.

Evaluating the performance of NoC architectures are usually performed using simulations[1,11,12,13,14,15]. Generally, the simulation is extremely slow for large systems and provides little insight on how different design parameters affect the actual NoC performance[16]. Analytical models, however, allow fast evaluation of performance metrics in early stages of the design process. This paper extends the work we have done by evaluating the performance (e.g., latency) of three on-chip interconnect architectures using Network Calculus[17]. We show how Network Calculus can be used to evaluate the performance metrics, energy consumption and area requirements of on-chip interconnects and their design tradeoffs. The main objective is to illustrate the effectiveness of this methodology in evaluating on-chip interconnect architectures. As a case study, a detailed analysis and evaluation of three on-chip interconnect architectures, the 2D mesh, WK-Recursive, and Spidergon, under different traffic loads is presented.

The rest of this paper is structured as follows. In section 2, we summarize the existing work on performance analysis methods proposed for evaluating on-chip interconnects. Section 3 provides a brief overview of Network Calculus concepts and features. In section 4, we present the on-chip interconnect modeling methodology, and the results obtained using both simulations and Network Calculus. Conclusions and future work are given in section 5.

2. Related Work

On-chip interconnect architectures adopted for SoCs are characterized by trade-offs between latency, throughput, communication load, energy consumption, and silicon area requirements. Several works, such as presented in[18], have demonstrated that there is a crucial need for system design tools and methodologies to analytically evaluating and comparing NoC architectures. The authors in[18] have pointed out that the current design tools and methodologies are not suitable for NoC evaluation, and simulation methods, despite their accuracy, are very expensive and time consuming. Therefore, techniques and tools are required to extract application communication characteristics and to efficiently estimating their performance and energy consumption in addition to area requirements for candidate communication architectures.

Recently, there has been a great deal of interest in the development of analytical performance models for NoC design. Approaches proposed in the literature can be classified in four main categories: deterministic approaches, probabilistic approaches, physics 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[19], a model using a cyclo-static dataflow graph was used for buffer dimensioning for NoC applications. Deterministic approaches assume that the designer has thorough understanding of the pattern of communication among cores and switches.

Most of the work to date using probabilistic approaches are based on queuing theory. For example, an analytical model using queuing theory was introduced in[20] 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. A queuing-theory-based model for evaluating the average latency and energy consumption of on-chip interconnects was proposed in[21]. The results from the analytical model were

validated with those obtained when using a cycle-accurate simulator. Most queuing approaches consider incoming and outgoing traffic as probability distributions (e.g., Poisson traffic) and allow designers to perform a statistical analysis on the whole system in order to evaluate certain network metrics, such as average buffer occupancy and average buffer delay in an equilibrium state. However, NoC applications exhibit traffic patterns that are very different compared to Poisson distribution used in queuing model[22,12]. More precisely, the Poisson model fails to capture some important network characteristics like self-similarity or long-range dependence[23].

In[24], the authors suggested statistical physics and information theory for NoC design and evaluation. Unlike stochastic approaches that make Markovian assumptions about the network behavior, statistical physics can model the interactions among various components while considering the long-term memory effects. A quantum-like approach was proposed in[24] 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 consequence of temperature variations. The authors have focused on the buffer sizing issue, 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. Network Calculus features are derived from system theory so that performance bounds (e.g., end-to-end delay) in networks such as the Internet can be modeled and evaluated[25,26]. The attractive feature of Network Calculus is its ability to capture all traffic patterns with the use of bounds. In other words, 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[27], we have presented a performance analysis methodology using Network Calculus to analyze and evaluate performance metrics of 2D Mesh on-chip interconnect. Simulations are performed and results are compared with those from the Network Calculus-based methodology in order to underline its usefulness for evaluating on-chip interconnects.

In this paper, the Network Calculus-based methodology is used to evaluate other performance metrics (e.g., load and throughput) as well as cost metrics (e.g., energy consumption and area overhead). Three on-chip interconnects, that are the 2D mesh, WK-Recursive, and Spidergon, are evaluated and compared under different traffic loads. Results show the effectiveness of Network Calculus as a useful tool for NoC design and evaluation. It's worth

noting that we have selected 2D mesh, WK-Recursive, and Spidergon because they outperform other on-chip interconnects, such as FT and Ring, in all performance and cost metrics[9,28].

3. Network Calculus: an Overview

Network Calculus[25,26] is a modeling framework that allows designers to specify a system as a mathematical model and evaluate main performance bounds such as end-to-end delay. This theory is based on $(min, +)$ algebra for deterministic network performance analysis, especially for worst-case analysis[25]. Based on shapes of the traffic flows, designers are able to capture some dynamic features of the network. In this section, we briefly introduce Network Calculus, in particular *service and arrival curves* that represent traffic patterns, as well as some performance bounds.

We consider that any system can be composed of one or several components that exchange traffic in order to accomplish a given task. The traffic pattern of the system can be defined by arrival curves of incoming traffic flows to each component of the system. Let's consider f a data flow characterized by an input function denoted by $R(t)$, which represents the cumulative data units (e.g., packets, bits) of f arriving at the component C within the time interval $[0, t]$. Let's consider $R^*(t)$ the output function (see Figure 1), which represents the cumulative amount of data that leaves the component during the time interval $[0, t]$, $R(t) \geq R^*(t)$. Having the input and output functions, we can derive the following two quantities of interest, the *backlog* and the *virtual delay*[25]. The *backlog* $x(t)$ is the amount of data units that are held inside the system, $x(t) = R(t) - R^*(t)$. The *virtual delay* $d(t)$ is the delay that would be experienced by a data unit arriving at time t if all units received before it is served before it, $d(t) = inf\{\tau \geq 0, R(t) = R^*(t + \tau)\}$.

In order to calculate the delay and the backlog, the input and output functions have to be defined. Their definition is based on $(min, +)$ convolution and deconvolution principles defined as follows. Given f and g wide-sense increasing functions and $f(0) = g(0) = 0$, their *convolution* is defined as $(f \otimes g)(t) = inf_{0 \leq s \leq t}\{f(t - s) + g(s)\}$ and their *deconvolution* is defined as $(f \oslash g)(t) = sup_{s \geq 0}\{f(t + s) - g(s)\}$.

Each input function can be characterized by an arrival curve as follows. An arrival curve $\alpha(t)$ characterizes a traffic flow $R(t)$, *iff* it upperbounds the amount of arriving data of this traffic flow during any time interval $[0, t]$. More formally, given a wide-sense increasing function $\alpha(t)$ defined

for $t \geq 0$, we say that a flow $R(t)$ is constrained by α *iff* for all $s \leq t$: $R(t) - R(s) \leq \alpha(t-s)$. It is also said that R has α as an arrival curve, or also that R is α-smooth[25]. Using $(min, +)$ convolution, α is an arrival curve of an input function R *iff* $R \leq R \otimes \alpha$. An example of the arrival curve is a *leaky bucket controller*, which enforces an arrival curve constraint $\alpha(t) = rt + b$. It means that no more than b data units can be sent at once and $r$bit/s on long-term.

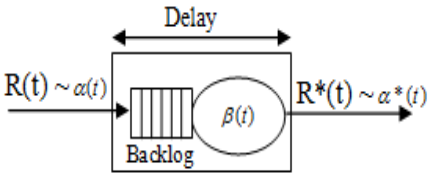

Figure 1: Arrival and service curves in Network Calculus with delay and backlog bounds

The output function $R^*(t)$ can be calculated after the modification of the input function $R(t)$ by the component C described by the service curve $\beta(t)$ of that component. We say that C offers to the flow R a service curve β (non-decreasing function such that $\beta(0) = 0$) *iff*: $\forall t \geq 0$, $R^*(t) \geq inf_{0 \leq s \leq t}\{R(s) + \beta(t-s)\}$. Using $(min, +)$ convolution of these two functions, β is a service curve of flow R *iff* $R^* \geq R \otimes \beta$. An example of the service curve is *rate latency function* $\beta(t) = R(t-T)^+$, where R denotes a guaranteed service rate and T is the maximum latency caused by the component[29]. The expression $(x)^+$ equals to x when $x > 0$ and 0 otherwise. Figure 1 shows a component with input/output curves, service curve, delay and backlog.

Knowing the service curve $\beta(t)$ offered by a component C, the output curve $\alpha^*(t)$ of $R^*(t)$, can be calculated as follows: $\alpha^*(t) = (\alpha \oslash \beta)(t)$. For example, assuming that a flow is constrained by an arrival curve $\alpha(t) = rt + b$ and C provides a guaranteed service curve $\beta(t) = R(t-T)^+$ to the flow, the output bound can be calculated as follows: $\alpha^*(t) = \alpha(t) + rt$. These curves, $\alpha(t)$ and $\alpha^*(t)$, act like bounds on the input and output traffic flows respectively, and are used to compute the delay bound D and the backlog bound B as follows. The delay D for a data flow $R(t)$ constrained by an arrival curve $\alpha(t)$ that receives the service $\beta(t)$ to produce a data flow $R^*(t)$ constrained by the arrival curve $\alpha^*(t)$ is upper-bounded by: $d(t) \leq sup_{s \geq 0}(inf\{\tau \geq 0 : \alpha(s) \leq \beta(s+\tau)\})$. The backlog $x(t)$ can be upper-bounded by: $x(t) \leq sup_{s \leq 0}\{\alpha(s) - \beta(s)\}$, $\forall t$.

An example is illustrated in Figure 2 that shows the delay and the backlog bounds of a component receiving a traffic flow characterized by an arrival curve $\alpha(t) = rt + b$ and providing a service curve $\beta(t) = R(t - T)^+$, where $R \geq r$ is the guaranteed bandwidth, and T is the maximum latency of the service. Using these curves, the backlog B and delay bounds D can be expressed as follows: $B = b + rT$ and $D = b/R + T$.

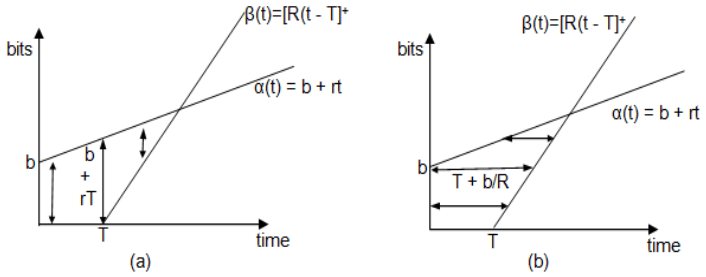

Figure 2: Example of backlog bound B (a) and delay bound D (b)

4. OCIs exploration

In this section, three on-chip interconnect architectures are selected for analysis and evaluations, 16-node configurations are used. Figure 3 shows these configurations with application data flows generated as a case study (e.g., in 2D Mesh, $f_1 = (c_8, s_8, s_{12}, c_{12})$). As shown in this figure, there are three important elements in NoC: cores, routers (or switches), and bidirectional links. Each core can be either a source or a sink, in which flits are constructed or consumed. Each ingress port in a switch has a buffer for temporary storage of information. When a flit arrives at a switch, it must go into the buffer that corresponds to a Drop-tail queue with an FIFO queue management mechanism. The rest of this section presents the Network Calculus-based model and how it is used to evaluate the performance and cost metrics.

4.1. Network Calculus-based Model

In order to analyze and evaluate the performance of each OCI, we need to build a model for the entire system. The NoC architecture can be viewed as a distributed system composed of autonomous nodes that communicate by

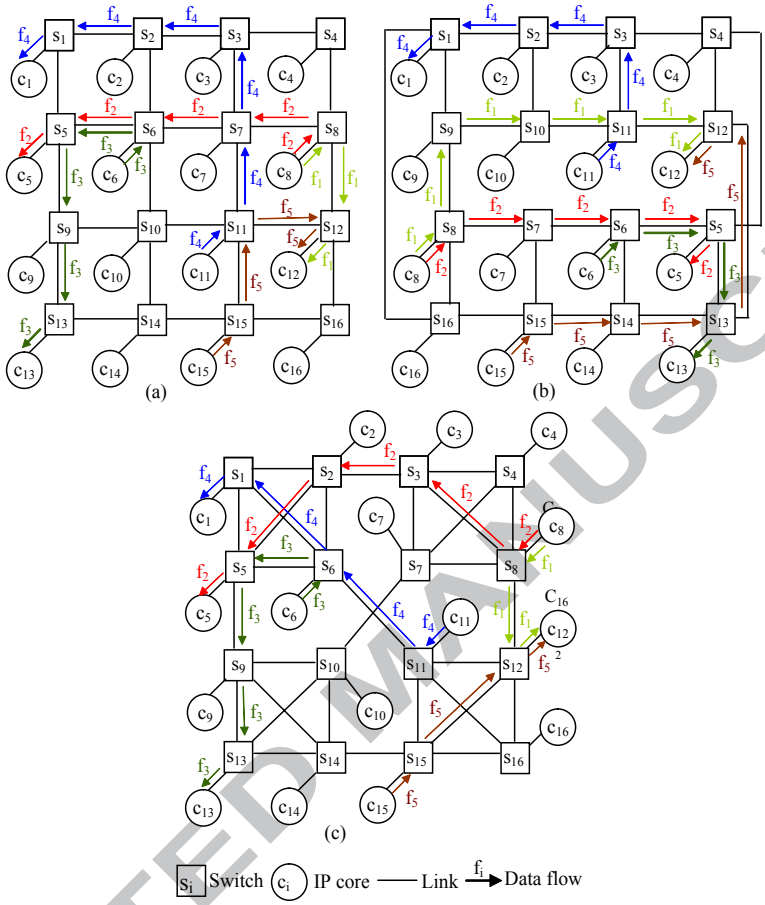

Figure 3: On-chip interconnects with data flows: (a) 2D Mesh, (b) Spidergon, (c) WK(4,2)-recursive.

exchanging messages through an on-chip interconnect[30]. 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 an acyclic digraph, called Task Graph TG, where each $v \in V$ represents a task and each $\ell = (u, v) \in E$ is a communication flow edge having one attribute $\overline{\alpha}_\ell(t)$, the input arrival curve that represents the data flow sent by u to v.

After a random mapping of the TG on the OCI, as illustrated in Fig-

ure 3, the cores $(c_6, c_8, c_{11}, c_{15})$ are selected to be traffic 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 to be a traffic sink. Data flows are represented by sequences of hops from a source core c_i to a destination core c_j. These data flows are computed using a deterministic routing protocol to direct flits between switches.

Having these data flows, we can express the input and output arrival curves, $\overline{\alpha}_{s_i}(t)$, $\overline{\alpha}_{c_i}(t)$, and $\overline{\alpha}_{\ell}(t)$ 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 $\overline{\alpha}_i(t) = r_i t + b_i$, 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. 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[31,32,33,34]. We use these curves to evaluate and compare the considered OCIs.

After defining data flows and nodes participating in transmitting and/or receiving data, the entire network can be described to obtain the performance model by merging all arrival and output flows. For example, Figure 3 (b) shows the 16-nodes configuration of the Spidergon on-chip interconnect. As shown in this figure, five data flows are selected as follows: $f_1 = (c_8, s_8, s_9, s_{10}, s_{11}, s_{12}, c_{12})$, $f_2 = (c_8, s_8, s_7, s_6, s_5, c_5)$, $f_3 = (c_6, s_6, s_5, s_{13}, c_{13})$, $f_4 = (c_{11}, s_{11}, s_3, s_2, s_1, c_1)$, $f_5 = (c_{15}, s_{15}, s_{14}, s_{13}, s_{12}, c_{12})$.

Based on these data flows, the input and output curves of each switch are *iteratively* calculated. For example, $\overline{\alpha}_{15}(t)$ and $\overline{\alpha}^*_{15}(t)$ respectively have to be calculated first. We have then, $\overline{\alpha}_{15}(t) = rt + b$ and $\overline{\alpha}^*_{15}(t) = rt + b + rT$. The output bound of the switch s_{15} is an input to the switch s_{14}, so $\overline{\alpha}_{14}(t) = rt + b + rT$ and $\overline{\alpha}^*_{14}(t) = rt + b + 2rT$. In the second iteration, input and output curves $\overline{\alpha}_8(t)$ are calculated as follows, $\overline{\alpha}_8(t) = 2rt + 2b$ and $\overline{\alpha}^*_8(t) = 2rt + 2b + 2rT$. In the third iteration, the input and output curves of $\overline{\alpha}_7(t)$ and $\overline{\alpha}_9(t)$ respectively have to be calculated in the same manner according to data flows. The calculation will be repeated with nodes s_6, s_5,

s_{13}, s_{10}, s_{11}, s_{12}, s_3, s_2, and s_1, till we obtain the following equations:

$$\begin{array}{ll}
\overline{\alpha}_1(t) = rt + b + \frac{9}{2}rT & \overline{\alpha}_9(t) = rt + b + rT \\
\overline{\alpha}_2(t) = rt + b + \frac{7}{2}rT & \overline{\alpha}_{10}(t) = rt + b + 2rT \\
\overline{\alpha}_3(t) = rt + b + \frac{5}{2}rT & \overline{\alpha}_{11}(t) = 2rt + 2b + 3rT \\
\overline{\alpha}_5(t) = 2rt + 2b + 4rT & \overline{\alpha}_{12}(t) = 2rt + 2b + 6rT \\
\overline{\alpha}_6(t) = 2rt + 2b + 2rT & \overline{\alpha}_{13}(t) = 2rt + 2b + 5rT \\
\overline{\alpha}_7(t) = rt + b + rT & \overline{\alpha}_{14}(t) = rt + b + rT \\
\overline{\alpha}_8(t) = 2rt + 2b & \overline{\alpha}_{15}(t) = rt + b
\end{array} \quad (1)$$

In the same manner, the arrival curve, $\overline{\alpha}_{c_i}(t)$, of each core c_i, and the arrival curve, $\overline{\alpha}_\ell(t)$, 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.

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. 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. Analytical and simulation results are compared 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[14].

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 Rate) model at a deterministic rate r, which is varied between 25Mbps and 100Mbps. It's worth noting that, in this evaluation, we have used Network Calculus theory, which is mainly proposed to study lossless system, i.e., with the assumption that no flits are ever lost. Once a flit is injected in the NoC, it will eventually reach its destination. When the injection rate is above 100Mbps, a lot of flits are lost. This is the reason why at this rate the network becomes congested and router start dropping flits. The maximum service rate R is fixed to 200Mbps in this simulation and same for each switch. In NoCs, the maximum service rate was expected to be in the order of Gigabits/s. However, because of the limitations from real conditions and since an event-simulator not cycle accurate simulator (event can represent many cycles that allow this high

bandwidth) is used, and processor power limitation, the maximum service rate can only add up to 200Mbps.

In the analytical evaluation, the arrival curve we have used for each node i is a leaky bucket controller which enforces an arrival curve constraint $\alpha(t) = rt + b$. Using this arrival curve, a node i can send b bits at once, but without exceeding $r$bit/s over the long run. One of the applications using arrival curve is in the Generic Cell Rate Algorithm (GCRA) with two parameters, target inter-arrival time of packets T, and τ the tolerance that quantifies how early packets may arrive with respect to the ideal spacing T[25]. A CBR connection is defined by one GCRA with parameters (T, τ), in which $b = S_f(\frac{\tau}{T} + 1)$ and $r = \frac{S_f}{T}$, where S_f is the flit size. In the simulator we have used, the CBR was implemented with τ equal to 0, therefore, b is all time equal to the flit size fixed.

The flit is an elementary unit of information exchanged in the communication network in a unit of time (e.g., clock cycle), but a packet is an element of information that an IP core sends to another core, which consists of a variable number of flits. The size of flits can be 8, 16, 32 or 64 bits, but in our evaluation, we keep the flit size to 8 bytes. The size has influence on the performance and cost metrics but not on the comparison results between on-chip interconnects. It is worth noting that, the length of packet, number and size of flits and the buffer size are all parameterized during the design space exploration. More precisely, after comparing different on-chip interconnects the designer can customize the suitable one by selecting appropriate parameters, such as the maximum service rate, the buffer size for each input port, and flit size, given a specific application.

In this evaluation study, we have considered latency, throughput, and communication load, which are the most important performance metrics used in evaluating on-chip interconnects[1,3,11,13]. Another performance metric is the loss rate not considered in this study because we are analyzing lossless NoCs. In addition to these performance metrics, cost metrics that are energy consumption and area requirements are considered.

4.2. Performance Metrics

In this section, performance metrics, mainly the latency, throughput, and communication load, will be evaluated using the input and output arrival curves $\overline{\alpha}_{c_i}(t)$, $\overline{\alpha}_{\ell}(t)$, and $\overline{\alpha}_{s_i}(t)$.

4.2.1. Latency

Latency is defined as the time that elapses between the 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 (e.g., 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[25]:

$$\mathcal{L}_{s_i} = \frac{b_i}{R_i} + T_i \tag{2}$$

where R_i is the service bandwidth and T_i is the maximum latency of the service at a switch s_i. Therefore, the average latency can be calculated based on equation 2. For example, as shown in the previous section (see eq.1), in Spidergon, since $\overline{\alpha}_7(t) = rt + b + rT$, $D_7 = (\frac{r}{R} + 1)T + \frac{b}{R}$, if the injection rate is $r = 100Mbps$, $R = 200Mbps$, b= 64bits, and the flit size is $S_f = 8$ bytes, then $D_6 = 0.8\mu s$, where $T = S_f/R$. After computing the delay bound of each switch, the total delay, called end-to-end delay bound, D_{f_i} of each data flow f_i (from the source to the sink) can be calculated by summing up the delay of each participating switch. It is defined as the time that elapses between the injection start of the flit into the network at the node source and its arrival to the destination node. For example, since $D_{f_3} = D_5 + D_6 + D_{13}$, if $r = 75Mbps$, then $D_{f_3} = 4.2\mu s$. The calculation continues in the same manner with D_{f_1}, D_{f_2}, D_{f_4}, and D_{f_5} to find the average end-to-end delay.

Figure 4 compares the average latency of the three on-chip interconnect architectures under different injection rate using Network Calculus (analysis) and simulation. As shown in this figure, when increasing the injection rate, the network becomes more congested with heavy traffic and hence queues become full causing more flits to wait, and therefore increasing the latency. We can also see that the latency obtained using network calculus analysis (i.e., a worst case analysis) is in the same order of magnitude as the latency obtained using simulations, i.e., both show a deviation of less than 14% on average. Furthermore, regardless of the injection rate used and in both simulation and analysis results, the Spidergon has higher average latency compared to the Mesh and WK because of high average number of hops flits traversed. We can also see that WK is less sensitive to the injection rate increases and has lower average latency.

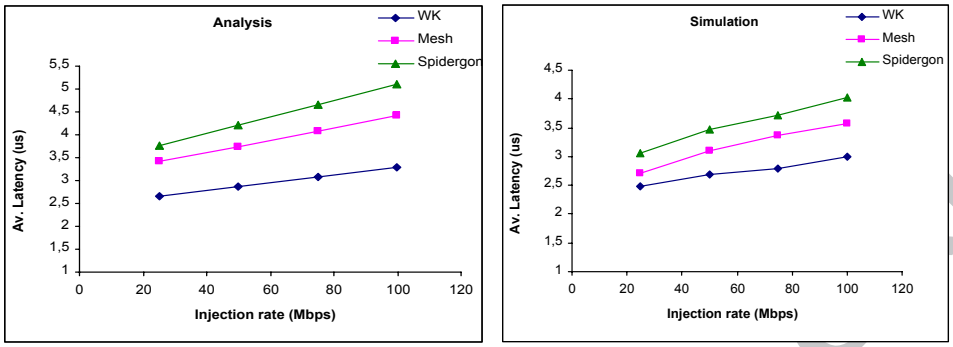

Figure 4: The average latency

4.2.2. Network Load

Communication load is a relative value of arrival rate versus departure rate on all links. Let's consider $\mathcal{D}_r(t)$ is 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[14]. 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:

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

where $\overline{\alpha}_{\ell_i}(t)$ is the number of flits arrived in the link ℓ_i, R is the bandwidth of each link ℓ_i, and N_ℓ is the number of unidirectional links involved in transporting flits. We consider that all links have the same bandwidth, R.

The results depicted in Figure 5 show the variation of communication load under different traffic rates for the three OCIs. 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 28%. Furthermore, regardless of the injection rate used, in both simulation and analysis results, the Spidergon has a higher communication load compared to the Mesh and WK. Furthermore, WK is less sensitive to the injection rate increases and has a slightly lower load.

4.2.3. Throughput

The throughput for each core c_i represents how many bits arrive at that core per second (bps). The aggregate throughput $\mathcal{T}(t)$ is the sum of through-

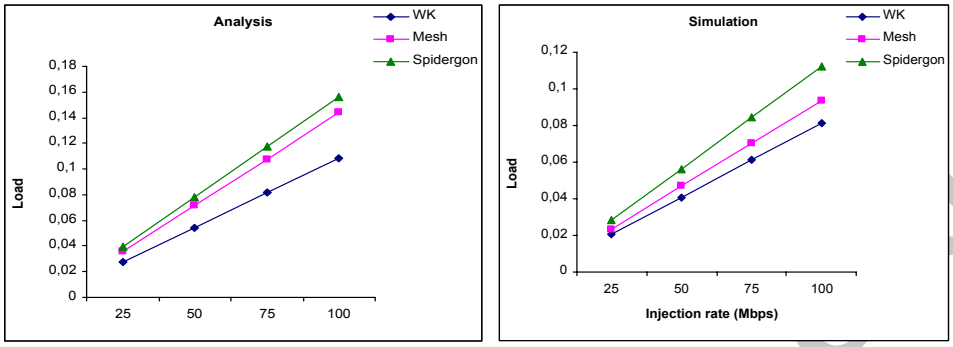

Figure 5: The communication load

put of each destination core c_i during the interval $[0, t]$. It can be calculated as follows:

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

where N_d is the number of cores selected as destinations (i.e., sinks), and $\overline{\alpha}_{c_i}(t)$ is the arrival curve that represents the accumulated number of bits arrived (i.e., accumulated) at the destination core c_i until time t.

In the example depicted in Figure 3, cores $(c_1, c_5, c_{12}, c_{13})$ are selected to be sinks. Using, the OCI model of the Spidergon, the arrival curve $\overline{\alpha}_{c_i}(t)$ of each core c_i can be calculated, for example, $\overline{\alpha}_{c_1}(t) = rt+b+\frac{11}{2}rT$ and $\overline{\alpha}_{c_5}(t) = rt + b + 3rT$. Figure 6 shows the variation of aggregate throughput under different injection rates for the three OCIs. The throughput increases linearly when the injection rate increases because of the number of flits generated. Furthermore, the throughput obtained using analysis is slightly similar to all OCIs and is in the same order of magnitude as the throughput obtained using simulations with a deviation of less than 5%.

4.3. Cost Metrics

This section presents the analytical evaluation of cost metrics, mainly the average energy consumption and area overhead. Analytical results are also compared to those obtained using simulations.

4.3.1. Energy

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

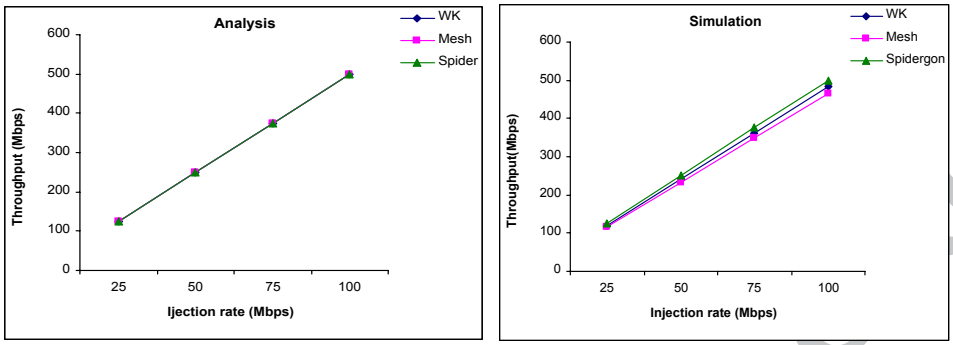

Figure 6: The aggregated throughput

wires or links between cores and switches. The total energy $\mathcal{E}(t)$, can be calculated as follows:

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

where $\overline{\alpha}_{\ell_i}(t)$ and $\overline{\alpha}_{s_j}(t)$ are the number of bits arrived until time t to the link ℓ_i and s_j respectively. N_ℓ and N_s are the number of links and switches involved in transporting the application flows. Therefore, the first term represents the energy consumed, at time t, on all links involved, and the second term represents the energy consumed inside the switches[28]. E_{ℓ_i} is the energy consumed during transporting one bit on a link ℓ_i, and E_{s_j} is the energy consumed during buffering and routing operations of one bit inside each switch s_j.

The values of E_{ℓ_i} and E_{s_j} depend mainly on the switch architecture and the link characteristic such as the width, the length, etc. In this evaluation, we use the values already estimated in the energy model proposed in[35] in which the average amount of energy required for a single bit to pass a switch is equal to $0.9776pJ/bit$ and the average 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. To calculate L_ℓ, 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 Figure 3, 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 shows the energy consumption using analytical evaluation 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. Furthermore, regardless of the injection rate used, in both simulation and analysis results, the Spidergon has higher average energy consumption compared to the Mesh and WK. This increase can be explained by the higher number of hops traversed by flits. We can also see that the energy obtained using analysis is in the same order of magnitude as the energy obtained using simulations, i.e., the difference between simulation and analysis is about 1%.

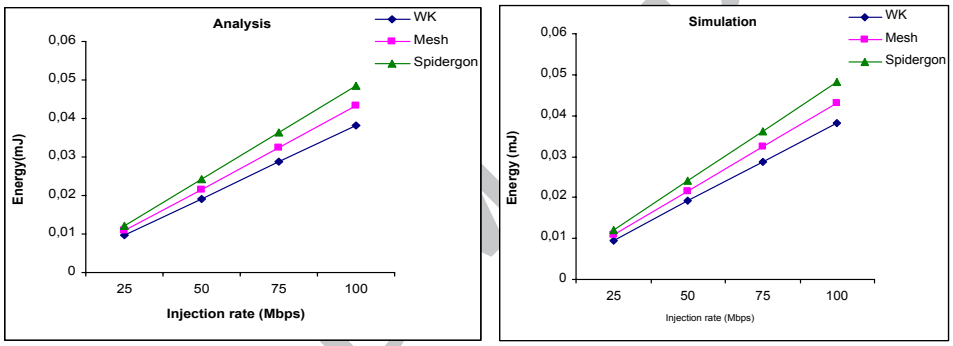

Figure 7: The average energy consumption

4.3.2. Area

In NoC design, three sources of area overhead can be identified, 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[36]. The total area value can be then calculated as follows:

$$\mathcal{A} = \sum_{i=1}^{N_s} \mathcal{A}_s(i) + \sum_{j=1}^{N_c} \mathcal{A}_c(j) + \sum_{k=1}^{N_\ell} \mathcal{A}_\ell(k) \tag{6}$$

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 $\mathcal{A}_\ell$, the

average switch area $\mathcal{A}_s$, and the average IP core area A_s. 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 presented in[28] to determine these values, in particular $\mathcal{A}_s$ and $\mathcal{A}_\ell$. So the average area $\mathcal{A}$ can be derived from eq.6 as follows:

$$\mathcal{A}_v = N_s(R_s + a_s d_g S_f B_s) + N_c \mathcal{A}_c + a_\ell N_\ell L_\ell \tag{7}$$

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 and L_ℓ is the average width and the average length of each link ℓ, R_s is another 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 previous works, for example in[36,37], 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 above, each switch s_i is constrained by an arrival curve in the form $\overline{\alpha}_{s_i}(t) = r_i t + b_i$ 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[25]:

$$\mathcal{B}_{s_i} = b_i + r_i T_i \tag{8}$$

where r_i is the core injection rate and T_i is the maximum latency of the service at the switch s_i. For example, in Spidergon, since $\overline{\alpha}_{s_1}(t) = rt + b + \frac{9}{2}rT$, $B_{s_1} = \frac{11}{2}rT + b$, if the injection rate is $r = 75Mbps$, $R = 200Mbps$, b= 64bits, and the flit size is $S_f = 8$ bytes, then B_{s_1} =24.5 bytes (∼3 flits), where $T = S_f/R$.

Figure 8 shows the area requirements (in mm^2) for zero flits drop (i.e., lossless system) under different injection rates. In this evaluation, the area required to store the routing table and other related area are considered constant, $R_s = 1mm^2$, and $a_s = 0.005mm^2$, $a_\ell = 0.02mm$, $A_c = 2mm^2$, $R_s = 1mm^2$. We also consider that the chip size is of $20mm \times 20mm$. The value of L_ℓ is calculated based on the architectures layout[28]. As shown in Figure 8, when injection rate increases, the area requirement increases 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. In other words, as the injection rate increases more space is needed

to avoid flits from being dropped. We can see, that the WK and Spidergon require more area because of the additional links and more buffer size respectively, when compared with the Mesh. Furthermore, area obtained using analytical evaluation is in the same order of magnitude as the area obtained using simulations, i.e., the difference between simulation and analysis is about 1.5%.

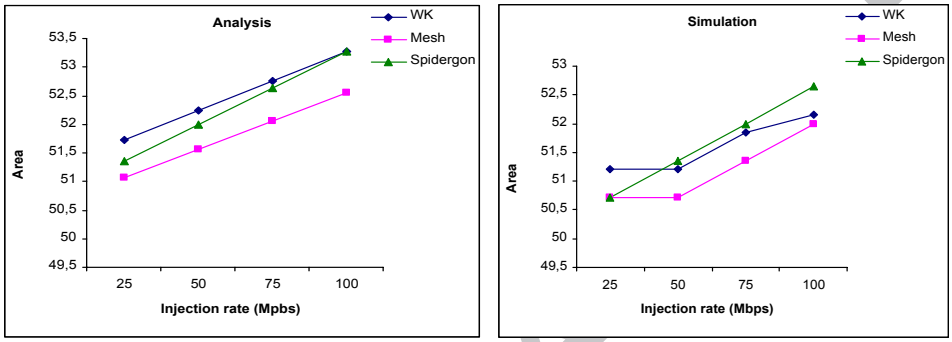

Figure 8: The average silicon area

5. Conclusions and Future Work

In this paper, a Network Calculus-based methodology is presented to evaluate on-chip interconnects in terms of performance (i.e., latency, communication load, throughput) and cost metrics (i.e., energy consumption and area requirements) based on a given traffic pattern. The main objective is to illustrate the practical use of the Network Calculus approach to analytically evaluating on-chip interconnects. The 2D regular Mesh, Spidergon, and WK on-chip interconnect architectures are compared and evaluated using a given traffic pattern. The results show that this approach can provide the designer with initial insight on on-chip interconnects and the relationship between application traffic and performance. The results show that WK-Recursive outperforms the 2D Mesh and Spidergon on-chip interconnects in all considered metrics.

Further work concerns the development of a design space exploration software tool that will be built around Network Calculus and integrated with a simulation and experimental environment. This software tool allows designers to rapidly explore design options over a wide range of energy budget and performance requirements. The utility of this tool will be demonstrated

via several prototypes that are created using reconfigurable platforms based on the FPGA technology where actual performance can be measured. Combining applications characterization, performance simulation and analysis, and implementation in one software tool allows filling the gap between pure simulation that may be too slow and analytic methods that are not accurate enough to be used in a design space exploration of SoCs.

References

[1] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, Performance evaluation and design tradeoffs for network-on-chip interconnect architectures, IEEE Trans. on Computer 54 (8) (2005) 1025-1040.

[2] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. berg, K. Tiensyrj, A. Hemani, A network on chip architecture and design methodology, Proc. Int't Symp. VLSI (ISVLSI) (2002) 117-124.

[3] L. Bononi, N. Concer, Simulation and analysis of network on ship architectures: Ring, spidergon, and 2d mesh, DATE Proc. (2006) 6 pages.

[4] P. Guerrier, A. Greiner, A generic architecture for on-chip packet-switched interconnections, DATE Proc. (2000) 250-256.

[5] W. J. Dally, B. Towles, Route packets, not wires: On chip interconnection networks, DAC Proc. (2001) 683-689.

[6] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, A. Scandurra, Spidergon: a novel on-chip communication network, Proc. International Symposium on System-on-Chip (2004) 250-256.

[7] F. Karim, A. Nguyen, S. Dey, An interconnection architecture for networking systems on chip, IEEE Microprocessors 22 (5) (2002) 36-45.

[8] S. Suboh, M. Bakhouya, T. El-Ghazawi, Simulation and evaluation of on-chip interconnect architectures: 2d Mesh, Spidergon, and WK-recursive networks, NoCS Proc. (2008) 205-206.

[9] S. Suboh, M. Bakhouya, J. Gaber, T. El-Ghazawi, An interconnection architecture for network-on-chip systems, Telecom. Systems 37 (1-3) (2008) 137-144.

[10] K. Lahiri, A. Raghunathan, S. Dey, System-level performance analysis for designing on-chip communication architectures, IEEE Trans. On CAD of Ics and Systems 20 (6) (2001) 768-783.

[11] S. Suboh, M. Bakhouya, S. Lopez-Buedo, T. El-Ghazawi, Simulation-based approach for evaluating on-chip interconnect architectures, SPL Proc. (2008) 75-80.

[12] G. Varatkar, R. Marculescu, Trac analysis for on-chip networks design of multimedia applications, DAC Proc. (2002) 510-517.

[13] A. Hegedus, G. M. Maggio, L. Kocarev, A ns-2 simulator utilizing chaotic maps for network-on-chip traffic analysis, Proc. of IEEE International Symposium on Circuits and Systems (2005) 3375-3378.

[14] Y. R. Sun, S. Kumar, A. Jantsch, Simulation and evaluation of a network on chip architecture using ns2, IEEE NorChip Proc.

[15] J. Xu, W. Wolf, J. Henkel, S. Chakradhar, A design methodology for application-specific networks-on-chip, ACM Transactions on Embedded Computing Systems 5 (2) (2006) 263-280.

[16] U. Y. Ogras, R. Marculescu, Analytical router modeling for networks-on-chip performance analysis, DATE Proc. (2007) 1-6.

[17] M. Bakhouya, S. Suboh, J. Gaber, T. El-Ghazawi, Analytical performance comparison of 2d mesh, wk-recursive, and spidergon nocs, The 19th IPDPS Conference, PMEO-UCNS workshop.

[18] K. Lahiri, S. Dey, A. Raghunathan, Evaluation of the traffic performance characteristics of system-on-chip communication architectures, VLSI Design Proc. (2001) 29.

[19] A. Hansson, M. Wiggers, A. Moonen, K. Goossens, M. Bekooij, Applying dataow analysis to dimension buffers for guaranteed performance in networks on chip, NOCS Proc. (2008) 211-212.

[20] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, M. Ould-Khaoua, An analytical performance model for the spidergon NoC, 21st AINA Proc. (2007) 1014-1021.

[21] H. J. Kim, D. Park, C. Nicopoulos, V. Narayanan, C. Das, Design and analysis of an NoC architecture from performance, reliability and energy perspective, ACM SANCS Proc. (2005) 173-182.

[22] U. Y. Ogras, J. Hu, R. Marculescu, Key research problems in NoC design: A holistic perspective, Proc. of International Conference on Hardware/Software Codesign and System Synthesis (2005) 69-74.

[23] R. Marculescu, P. Bogdan, The chip is the network: Toward a science of network-on-chip design, Foundations and Trends in Electronic Design Automation 2 (4) (2007) 371-461.

[24] P. Bogdan, R. Marculescu, Quantum-like effects in network-on-chip buffers behavior, Proc. of the 44th Design Automation Conference (2007) 266-267.

[25] J.-Y. L. Boudec, P. Thiran, Network calculus: A theory of deterministic queuing systems for the internet, Book, LNCS 2050 (2001) 265 pages.

[26] R. L. Cruz, A calculus for network delay, part ii: Network analysis, IEEE Tran. on Information Theory 37 (1) (1991) 132-141.

[27] M. Bakhouya, S. Suboh, J. Gaber, T. El-Ghazawi, Analytical modeling and evaluation of on-chip interconnects using network calculus, Proc. of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (2009) 74-79.

[28] M. Bakhouya, Evaluating the energy consumption and the silicon area of on-chip interconnect architectures, Journal of Systems Architecture 55 (7-9) (2009) 387-395.

[29] D. Stiliadis, A. Varma, Latency-rate servers: a general model for analysis of traffic scheduling algorithms, IEEE/ACM Trans. Networking 6 (5) (1998) 611-624.

[30] S. Suboh, M. Bakhouya, J. Gaber, T. El-Ghazawi, Analytical modeling and evaluation of network-on-chip interconnect architectures, International Conference on High Performance Computing and Simulation (HPCS) (2010) 491-497.

[31] A. Bouillard, B. Gaujal, S. Lagrange, E. Thierry, Optimal routing for end-to-end guarantees using network calculus, performance evaluation, Performance Evaluation 64 (11-12) (2008) 883-906.

[32] V. Firoiu, J.-Y. L. Boudec, D. Towsley, Z.-L. Zhang, Theories and models for internet quality of service, Proceedings of the IEEE 90 (9) (2002) 1565-1591.

[33] J.-P. Georges, E. Rondeau, T. Divoux, Evaluation of switched ethernet in an industrial context by using the network calculus, 4th IEEE Workshop on Factory Communication Systems (2002) 19-26.

[34] J. B. Schmitt, F. A. Zdarsky, U. Roedig, Sensor network calculus with multiple sinks, Proceedings of the Performance Control in Wireless Sensor Networks Workshop at the IFIP Networking Conference (2006) 6-13.

[35] P. T. Wolkotte, G. Smit, N. Kavaldjiev, J. Becker, J. Becker, Energy model of networks-on-chip and a bus, Proc. of International Symposium on System-on-Chip (2005) 82-85.

[36] A. O. Balkan, G. Qu, U. Vishkin, A mesh-of-trees interconnection network for single-chip parallel processing, Proc. of International Conference on Application-specific Systems, Architectures and Processors (2006) 73-80.

[37] M. Coenen, S. Murali, A. Ruadulescu, K. Goossens, G. D. Micheli, A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control, Proc. of International Conference on Hard- ware/Software Codesign and System Synthesis (2006) 130-135.