Buffer Optimization in Network-on-Chip Through Flow Regulation

Fahimeh Jafari, Zhonghai Lu, *Member, IEEE,* Axel Jantsch, *Member, IEEE,* and Mohammad Hossein Yaghmaee, *Member, IEEE*

***Abstract*—For network-on-chip (NoC) designs, optimizing buffers is an essential task since buffers are a major source of cost and power consumption. This paper proposes flow regulation and has defined a regulation spectrum as a means for system-on-chip architects to control delay and backlog bounds. The regulation is performed per flow for its peak rate and burstiness. However, many flows may have conflicting regulation requirements due to interferences with each other. Based on the regulation spectrum, this paper optimizes the regulation parameters aiming for buffer optimization. Three timing-constrained buffer optimization problems are formulated, namely, buffer size minimization, buffer variance minimization, and multiobjective optimization, which has both buffer size and variance as minimization objectives. Minimizing buffer variance is also important because it affects the modularity of routers and network interfaces. A realistic case study exhibits 62.8% reduction of total buffers, 84.3% reduction of total latency, and 94.4% reduction on the sum of variances of buffers. Likewise, the experimental results demonstrate similar improvements in the case of synthetic traffic patterns. The optimization algorithm has low run-time complexity, enabling quick exploration of large design spaces. This paper concludes that optimal flow regulation can be a highly valuable instrument for buffer optimization in NoC designs.**

***Index Terms*—Buffer size, buffer variance, interior point method, network-on-chip (NoC), optimization problem.**

I. Introduction

THE advance of the technology is raising the level of integration of intellectual property (IP) and scalability issue for communication architectures in very large-scale integration systems. Since traditional buses do not scale well in the system-on-chip (SoC) platforms, this trend has driven bus-based architecture toward networks-on-chip (NoCs) [1]. Current achievements in integrating more processor cores on a single chip enable to employ these many-core systems as real time multimedia servers. Thus, it is imperative to provide quality of service (QoS) in these systems which have been well available in traditional Internet servers. IPs for a SoC are typically developed concurrently using a standard interface, e.g., advanced extensible interface or open core protocol. Despite the standard interfaces, integrating IPs into a SoC infrastructure presents challenges because: 1) traffic flows from IPs are diverse and typically have stringent performance constraints; 2) the impact of interferences among traffic flows is hard to analyze; and 3) of the cost and power constraint, buffers in the SoC infrastructure must not be over-dimensioned while still satisfying performance requirements even under worst-case conditions.

Fig. 1 illustrates the approach that we have proposed and investigated in [2] for addressing the IP integration problem. Master IPs send read and write requests to slave IPs which respond with read data and write acknowledgments. The admission of traffic flows from master IPs into the SoC infrastructure can be controlled by a regulator rather than injecting them as soon as possible. Thus, we can control QoS and achieve cost-effective communication. To lay a solid foundation of the approach, our flow regulation has been based on network calculus [3]–[6]. By importing and extending the analytical methods from network calculus, we can obtain worst-case delay and backlog bounds. In [7], we implemented the microarchitecture of the regulator and quantified its hardware speed and cost. The aim of this paper is to optimize the regulator parameters including peak rate and traffic burstiness of flows by formulating optimization problems.

Silicon area and power consumption are two critical design challenges for NoC architectures. The network buffers take up a significant part of the NoC area and power consumption [8]; consequently, the size of buffers in the system should be minimized. On the contrary, buffers should be large enough to improve communication performance. This means that there is a tradeoff between buffer size and performance metrics. Hence, we address an optimization problem of minimizing the total number of buffers subject to the performance constraints of the applications running on the SoC. Moreover, since reusing similar or identical switches facilitates the design process of NoC-based systems, we formulate another optimization problem to minimize the variances of buffer size in the respective output buffers of switches. As both of the mentioned objective functions are worthwhile for the design process, we formulate them as a multiobjective problem under QoS constraints. Finally, we show the benefits of the proposed method and quantify performance improvement and buffer size and variance reduction.

The remainder of this paper is organized as follows. Section II gives an account of related works. In Section III, we

Manuscript received December 1, 2009; revised May 12, 2010; accepted July 3, 2010. Date of current version November 19, 2010. This paper was recommended by Associate Editor V. Narayanan.

F. Jafari, Z. Lu, and A. Jantsch are with the Department of Electronic Systems, Royal Institute of Technology, SE-164 40 Kista, Stockholm, Sweden (e-mail: fjafari@kth.se; zhonghai@kth.se; axel@kth.se).

M. H. Yaghmaee is with the Computer Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad 91775-111, Iran (e-mail: hyaghmae@ferdowsi.um.ac.ir).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2063130

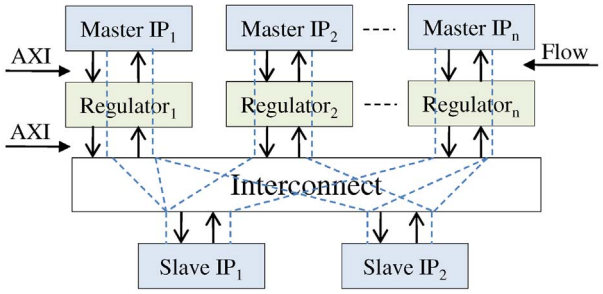

Fig. 1. IP integration in SoCs.

introduce the flow regulation concept along with the basics of *Network Calculus* [3]–[6]. Section IV discusses the underlying system model. Section V is devoted to the discussion about the buffer optimization problems. In Section VI, we present the solution method using an iterative approach. Our simulation results are described in Section VII. We discuss the scope and assumptions in Section VIII. Finally, Section IX gives the conclusion and future work.

II. Related Work

A. Network Calculus

Cruz [4] and Chang [6] have pioneered the network calculus [4], which is a mathematical framework to derive worst case bounds on maximum latency, backlog, and minimum throughput. In [5], a general latency-rate server model was proposed for analyzing traffic scheduling algorithms. Based on this model, they derived deterministic delay and backlog bounds. Le Boudec and Thiran [3] summarized the results of network calculus and their applications in Internet and ATM. Real-time calculus [9], close to network calculus, was developed for platform-based embedded systems. It generalizes standard event models via upper and lower arrival curves, and processing-element models via upper and lower service curves. Based on these curves, it derives delay and backlog bounds. The authors in [2] proposed a network calculus-based flow regulation and defined a regulation spectrum as a design instrument for SoC architects to control QoS. In this paper, we use the concept of regulation and regulation spectrum in [2] and address the issue of optimal regulation for buffer optimization. We optimize the regulator parameters including peak rate and traffic burstiness of flows by formulating optimization problems.

B. Application Specific Design

NoC-based SoC architectures are often designed for a specific application or a class of applications. Thus, designers customize it for a specific application to achieve best performance and cost trade-offs. The authors in [10] and [11] show the advantages of the topological mapping of IPs on the NoC architectures. In [12], the network topology customization and its effects on the system are considered. In [13] and [14], the authors investigate the customized allocation of buffer resources to different channels of routers. Actually, these works strived to distribute a given budget of buffering space among channels. Also, they are based on the average-case analysis which is not sufficient for a system with hard real-time requirements.

In [15], we followed a different direction by addressing an optimization problem to find the minimum total buffering requirements while satisfying acceptable communication performance in NoCs with round robin arbitration. In this paper, we have significantly extended the work in [15]. We address not only the buffer size minimization problem but also the buffer variance minimization problem. Moreover, since both objectives are desirable for NoC designs, we formulate a multiobjective optimization problem to minimize both buffer size and buffer variance. We give a systematic account of all the three problems, i.e., the buffer size minimization, the buffer variance minimization, and the multiobjective optimization. Furthermore, we construct the model for weighted round robin arbitration which outperforms round robin policy. It is worth mentioning that our method is presented based on tight worst-case bounds derived by network calculus. Therefore, it is suitable for real-time system designs.

C. Optimization Method

In this paper, we formulate optimization problems to optimize the regulator parameters with respect to buffer requirements.

In the literature, the proposed constrained problems are called nonconvex nonlinear programming (NLP) problems [16]. The general aim in constrained optimization is to transform the problem into an easier subproblem that can then be solved and used as the basis of an iterative process [16]. A characteristic of a large class of early methods is the translation of the constrained problem to a basic unconstrained problem by using a penalty function for constraints that are near or beyond the constraint boundary. In this way, the constrained problem is solved using a sequence of parameterized unconstrained optimizations, which in the limit converge to the constrained problem. These methods are now considered relatively inefficient and have been replaced by methods that have focused on the solution of the Karush-Kuhn-Tucker (KKT) equations [16], [17]. The KKT equations are necessary conditions for optimality for a constrained optimization problem.

The solution of the KKT equations forms the basis to many nonlinear programming algorithms. These algorithms attempt to compute the Lagrange multipliers directly. In particular, we will solve the proposed optimization problems using interior point method for constrained NLP problems [16], [17].

III. Concepts of Flow Regulation

A. Network Calculus Basics

A flow f is an infinite stream of unicast traffic (packets) sent from a source node and flow j is denoted as f_j. In network calculus [3], a flow $f_j(t)$ represents the accumulated number of bits transferred in the time interval $[0, t]$. To obtain the average and peak characteristics of a flow, traffic specification (TSPEC) is used. With TSPEC, f_j is characterized by an *arrival curve* $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ in which L_j is the maximum transfer size, p_j the peak rate ($p_j \geq \rho_j$), σ_j the burstiness

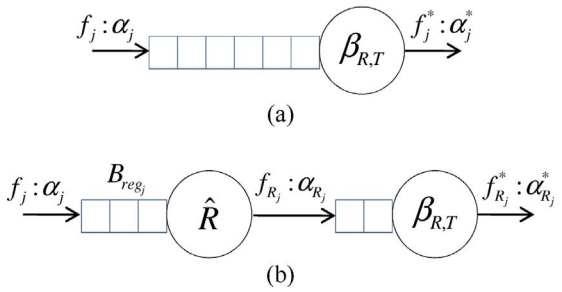

Fig. 2. Flow served by a latency-rate server. (a) Flow served without regulation. (b) Flow served after regulation.

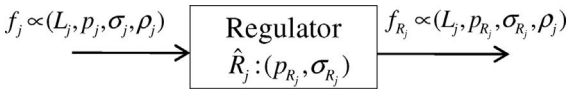

Fig. 3. Flow regulation.

($\sigma_j \geq L_j$), and ρ_j the average (sustainable) rate. We denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. The burstiness also is an important case among these parameters because a flow with low average rate and unlimited burst size can incur an unlimited delay on its own packets.

Network calculus uses the abstraction of service curve to model a network element (node) processing traffic flows. A service curve reflects the processing latency and service capability of the node. A well-formulated service model is the latency-rate function $\beta_{R,T} = R(t-T)^+$, where R is the minimum service rate and T is the maximum processing latency of the node [5]. Notation $x^+ = x$ if $x > 0; x^+ = 0$, otherwise.

As depicted in Fig. 2(a), a TSPEC flow $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$ (denoted as $f_j : \alpha_j$) is served by a node guaranteeing a latency-rate service $\beta_{R,T}$. According to [3], the maximum delay and the buffer required for flow j are bounded by (1) and (2), respectively

$$\bar{D}_j = \frac{L_j + \theta_j (p_j - R)^+}{R} + T \tag{1}$$

$$\bar{B}_j = \sigma_j + \rho_j T + (\theta_j - T)^+ [(p_j - R)^+ - p_j + \rho_j] \tag{2}$$

where $\theta_j = (\sigma_j - L_j)/(p_j - \rho_j)$. The output flow f_j^* is bounded by another affine arrival curve $\alpha_j^*(t) = (\sigma_j + \rho_j T) + \rho_j t,\ \theta_j \leq T$; $\alpha_j^*(t) = min((T+t)(min(p_j, R)) + L_j + \theta_j (p_j - R)^+, (\sigma_j + \rho_j T) + \rho_j t),\ \theta_j > T$.

B. Regulation Spectrum

TSPEC can be used to characterize flows. It can also be used to define a traffic regulator. Fig. 3 shows that an input flow f_j reshaped by a regulation component $\hat{R}_j(p_{R_j}, \sigma_{R_j})$ results in an output flow f_{R_j}. We assume the regulator has the same input and output data unit, *flit*, and the same input and output capacity *C flits/cycle*. We also assume that f_j's average bandwidth requirement must be preserved. The output flow f_{R_j} is characterized by the four parameters $(L_j, p_{R_j}, \sigma_{R_j}, \rho_j)$, where $p_{R_j} \in [\rho_j, p_j]$, $\sigma_{R_j} \in [L_j, \sigma_j]$. f_j can be losslessly reshaped by the regulator, meaning that f_{R_j} has the same L and average rate ρ as f_j. The two intervals $p_{R_j} \in [\rho_j, p_j]$ and

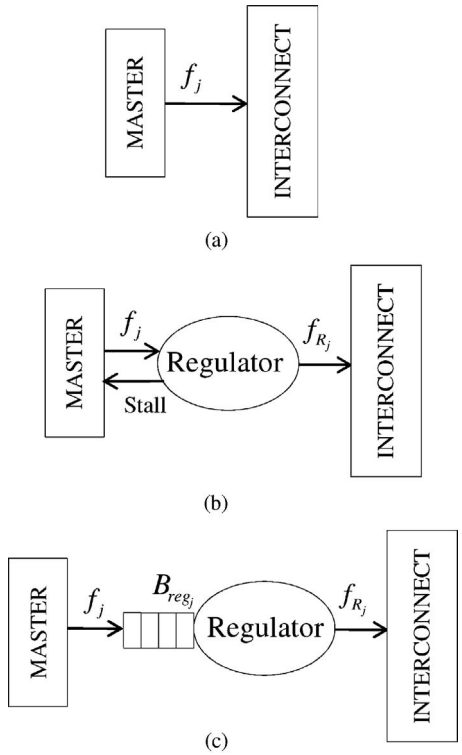

Fig. 4. Mechanisms of flow regulation. (a) Self-regulating master. (b) IPs are stalled: no queuing buffer. (c) IPs are not stalled: queuing buffer.

$\sigma_{R_j} \in [L_j, \sigma_j]$ are called the *regulation spectrum*, where the former is for the regulation of peak rate and the latter for the regulation of traffic burstiness.

The regulation spectrum defines the upper and lower limits of regulation. Fig. 2(b) shows how the flow is served after regulation. We implemented microarchitecture of the regulator and quantified its hardware speed and cost in [7]. Selecting appropriate p_{R_j} and σ_{R_j} is very effective in performance and cost of communications. In the next sections, we formulate three optimization problems that consider these regulation parameters as decision variables.

C. Mechanism and Cost of Flow Regulation

There are three different ways to realize the flow regulation, each of which incurs different costs.

1) *Regulation by design methodology:* as shown in Fig. 4(a), no regulator is implemented in the system. The IP or the application is designed such that it meets the regulation requirements. If that can be guaranteed, there is no additional cost in the network or the network interface. Also, there is no buffers and no delay due to regulation; consequently, there is no hardware cost for designing the regulator. However, the design structure of master should be changed to have a self-regulating master. This means that the workload is pushed to the master and application design. Thus, it applies to new IPs, but not applicable to legacy IPs.
2) *Regulation by a hardware regulator:* a hardware regulator is implemented which enforces traffic regulation at the network interfaces. There are two ways that the hardware regulator may affect the behavior of IPs as

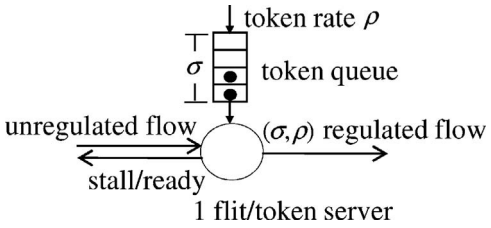

Fig. 5. (σ, ρ)-based regulation mechanism.

follows.

a) As shown in Fig. 4(b), the regulator does not buffer the packets, but stalls the traffic producers or IPs. In this case, no buffer due to regulation is required, but the behavior of masters should also be modified. This may be a good idea if the traffic producer is a multitasking CPU that can do something else while waiting. In this case the traffic generation is simply delayed and no buffering costs occur in the system.

b) The traffic producers or IPs are not stalled but the regulators use buffers to store transactions as depicted in Fig. 4(c). This can reduce back-pressure at the expense of buffering cost. Thus, this scheme allows any legacy IPs to be directly used in the system.

In principle, which option is best will depend on the context (application, IPs, architecture, and so on). The significant benefit of case 2b in comparison with others is simplicity of design process because no changes are required for the master structure. In this paper, we have implemented our proposed method based on case 2b concepts, but it can be easily extended for other cases.

To evaluate the overhead in silicon area due to the use of regulators, we designed and synthesized a multi-flow regulator with Synopsys tools using 180 nm technology [7]. When optimized for area, the multi-flow regulator using three regulators consumes $5K$ gates, running up to 730 MHz. Buffers and packet latency due to regulation depend on the values of the regulation parameters including peak rate and traffic burstiness which will be calculated in our case study in Section VII. The regulation mechanism in this paper is described as follows.

The regulator is implemented using the token-bucket mechanism [18] as shown in Fig. 5. The token queue has a size of σ. Initially the token queue is full. The 1-flit/token server admits one flit by de-asserting the "stall" signal as long as the token queue is not empty. The token queue is realized by a saturating credit counter that increments at rate ρ and saturates when it reaches a count of σ. A flit can be transmitted if and only if the credit counter is positive (at least one token available). Each time a flit is sent, the counter is decremented by 1.

IV. System Model and Delay/Backlog Bounds

We aim at optimizing buffer requirements while satisfying acceptable latency in on chip communications. We shall formulate optimization problems based on an analytical performance model. At first, we shall derive the per-flow worst-case delay and backlog bounds.

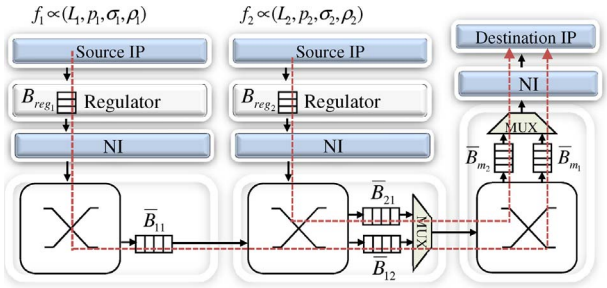

Fig. 6. Example of required buffers for two flows.

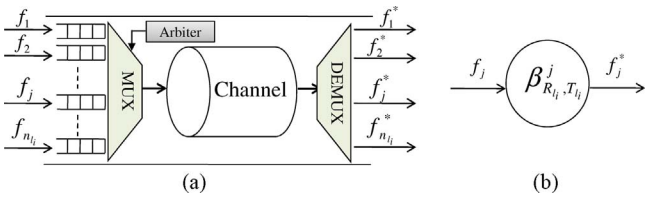

Fig. 7. (a) Channel sharing among set of flows. (b) Channel service model for flow j.

A. Assumptions and Notations

We consider a NoC architecture which can have different topologies. Every node contains an IP core and a router with $p+1$ input channels and $q+1$ output channels. Each IP core performs its own computational, storage or input/output processing functionality, and is equipped with a network interface (NI). NIs provide an interface between IPs and the network and they are responsible for packetization/depacketization of messages. Note that the presence of NIs is the consequence of using a network rather than using regulators. Regulators are inserted between the source IP and the NI. We presume the number of virtual channels for each physical channel is the same as the number of flows passing through that channel. Fig. 6 shows required buffers of flows f_1 and f_2 from different sources to the same destination. The following analysis on buffer requirements of flows is illustrated by this figure. We also assume that the NoC architecture is lossless, and packets traverse the network in a best-effort fashion using a deterministic routing. This means that the path of a flow is statically determined.

To facilitate our discussions, we turn the aforementioned NoC architecture into a mathematically modeled network. In this respect, we consider a NoC as a network with a set of bidirectional channels L, and a set of flows F. Each physical channel $i \in L$ has a fixed capacity of c_{l_i} *flits/cycle*. We denote the set of flows that share channel i by F_{l_i} and their number is denominated as n_{l_i}. Similarly, the set of channels that flow j passes through, is denoted by L_{f_j} and their number is denominated as n_{f_j}. By definition, $j \in F_{l_i}$ if and only if $i \in L_{f_j}$.

B. Channel Service Model

To compute the flow traversal delay and backlog bounds using the equations, we first need to build a channel service model. The network channel and the ejection channel at the destination node are treated in the same way since both types of channels are multiplexed by multiple flows with an arbitration policy.

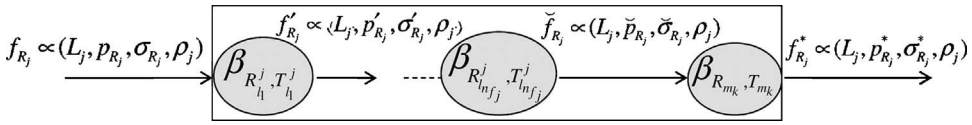

Fig. 8. Modeling each network element as a latency-rate server.

Fig. 7(a) depicts a channel l_i allocated to n_{l_i} flows. Since the arbitration policy determines how much the flows influence each other, it has to be known. We assume that, while serving multiple flows, the routers employ weighted round robin scheduling to share the link bandwidth. Assuming a fixed word length of L_w in all of flows, weighted round robin arbitration means that each flow j gets at least a $\frac{\rho_j}{\sum_{\forall f_k \in F_{l_i}} \rho_k} c_{l_i}$ of the channel bandwidth. A flow may get more if the other flow uses less, but we now know a worst-case lower bound on the bandwidth. Since network calculus uses the abstraction of service curve to model a network element processing traffic flows [3], we can also model a weighted round robin arbiter of channel l_i for flow j as a latency-rate server [19] that its function is as $\beta_{R_{l_i}^j, T_{l_i}^j} = R_{l_i}^j (t - T_{l_i}^j)^+$, where $R_{l_i}^j$ is the minimum service rate and $T_{l_i}^j$ is the maximum processing latency of the arbiter of channel l_i for flow j. $R_{l_i}^j$ and $T_{l_i}^j$ are defined as follows:

$$R_{l_i}^j = \frac{\rho_j}{\sum_{f_k \in F_{l_i}} \rho_k} c_{l_i} \tag{3}$$

$$T_{l_i}^j = \frac{(\sum_{f_k \in F_{l_i}} N_{l_i}^k - N_{l_i}^j) L_w}{c_{l_i}} \tag{4}$$

where $N_{l_i}^j$ is the minimum positive integer for flow j passing through channel i provided that

$$\frac{\rho_j}{\sum_{\forall f_k \in F_{l_i}} \rho_k} = \frac{N_{l_i}^j}{\sum_{\forall f_k \in F_{l_i}} N_{l_i}^k} \qquad \forall f_j \in F_{l_i}.$$

For (3), $R_{l_i}^j$ denotes the minimum weight-proportional bandwidth that flow j can take from channel i. For (4), $T_{l_i}^j$ denotes the maximum blocking time for flow j when passing through channel i. The channel service model for flow j is shown in Fig. 7(b).

With the channel service model, we can now model a flow passing through a series of channels including the ejection channel as a series of concatenated latency-rate servers. Fig. 8 shows a traffic flow f_j after regulation which is called f_{R_j} and is passing through adjacent channels. We construct an analytical model with the network elements depicted in this figure. Every channel $l_i \in L_{f_j}$ that flow j passing through can be modeled as a latency-rate server for flow j with service curve $\beta_{R_{l_i}^j, T_{l_i}^j}$, and also the ejection channel in the destination node of flow j, node k, can be modeled as a latency-rate server with service curve $\beta_{R_{m_k}, T_{m_k}}$.

C. Tight Worst-Case Bounds for Each Flow

Consider that flow j passes through the regulator and several network channels offering each a latency-rate service curve. For each flow, the delay and backlog bounds have two components: one incurred at the regulator and the other the network.

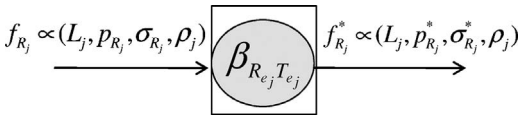

Fig. 9. Modeling all network elements as a latency-rate server.

1) *Delay and Backlog Bounds at Regulators:* To determine the delay and backlog due to the regulation, its impact on the behavior of IPs should be considered. As discussed in Section III-C, one is that IPs are *stalled* and therefore, there is no queuing buffer at the regulator. In the other case which is adopted in this paper, IPs are *not stalled* and the regulators use buffers to store transactions. This can decrease back-pressure at the expense of buffering cost. Let D_{reg_j} and B_{reg_j} be the delay and backlog for flow j due to regulation, respectively. We have $B_{reg_j} = \Delta\sigma_j = \sigma_j - \sigma_{R_j}$, which is the difference between the input and output burstiness of the regulator, and $D_{reg_j} = \Delta\sigma_j / \rho_j$ [2].

2) *Delay and Backlog Bounds in the Network:*

a) *Delay bound:* To compute the delay bound for a flow passing a series of nodes, one simple way is to calculate the summation of delay bounds at each node. However, this results in a loose total delay bound. To tighten the worst-case delay bound along the network, we use the theorem of *concatenation of network elements* [3]. Given are two nodes sequentially connected and each is offering a latency-rate service curve β_{R_i, T_i}, $i = 1$ and 2. These nodes can be represented as a single latency-rate server as follows:

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2}. \tag{5}$$

As depicted in Fig. 9, we can model all network elements on a given flow as a single latency-rate server $\beta_{R_{e_j}, T_{e_j}}$ with the following characteristics:

$$R_{e_j} = min\left(min_{l_i \in L f_j} \left(\frac{\rho_j}{\sum_{f_k \in F_{l_i}} \rho_k} c_{l_i} \right), \frac{\rho_j}{\sum_{f_r \in F_{d_k}} \rho_r} c_{m_k} \right) \tag{6}$$

$$T_{e_j} = \sum_{l_i \in L f_j} \left(\frac{(\sum_{f_k \in F_{l_i}} N_{l_i}^k - N_{l_i}^j) L_w}{c_{l_i}} \right) + \frac{(\sum_{f_r \in F_{d_k}} N_{d_k}^r - N_{d_k}^j) L_w}{c_{m_k}} \tag{7}$$

where R_{e_j} denotes the minimum service rate among channels through which flow j passes and T_{e_j} the sum of maximum processing latency of the mentioned channels.

Based on a corollary of this theorem which is known as *Pay Bursts Only Once* [3], the equivalent latency-rate server is used for obtaining worst-case delay bound. Therefore, according to

(1), (6), and (7), the maximum delay for flow j in network is bounded by

$$\bar{D}_j = \frac{L_j + \theta_{R_j}(p_{R_j} - R_{e_j})^+}{R_{e_j}} + T_{e_j} + n_{f_j} d_p \tag{8}$$

where d_p is delay for propagation in a channel which is assumed identical for all channels. Therefore, $n_{f_j} d_p$ is propagation delay in whole network for flow j and $\theta_{R_j} = \frac{\sigma_{R_j} - L_j}{p_{R_j} - \rho_j}$. Hence, the total maximum delay for the flow j is bounded as $D_{reg_j} + \bar{D}_j$.

b) *Backlog bound:* For calculating tight worst-case bound on backlog along the network, the sum of the individual bounds on every element is computed [3]. Thus, the required buffer in network for flow j is bounded by

$$\bar{B}_j = \sum_{i \in L_{f_j}} \bar{B}_{ji} + \bar{B}_{m_j} \tag{9}$$

where $\bar{B}_{ji}$ is the upper bound on the buffer for flow j for each $i \in L_{f_j}$ and $\bar{B}_{m_j}$ is the maximum required buffer for the ejection channel multiplexer of the destination node of flow j. $\bar{B}_{ji}$ and $\bar{B}_{m_j}$ can be easily obtained by (2). For example, directly applying (2) for flow j in Fig. 8, $\bar{B}_{m_k}$ can be calculated by

$$\bar{B}_{m_k} = \breve{\sigma}_{R_j} + \rho_j T^j_{m_k} + (\breve{\theta}_j - T^j_{m_k})^+ [(\breve{p}_{R_j} - R^j_{m_k})^+ - \breve{p}_{R_j} + \rho_j]. \tag{10}$$

Finally, the total buffer requirements for flow j are bounded by $B_{reg_j} + \bar{B}_j$.

V. Buffer Optimization Problems

A. Buffer Size Optimization

As stated before, our objective is to choose output peak rate and traffic burstiness of regulators for each flow so as to minimize the buffer requirements while satisfying acceptable performance in the network. Thus, the buffer size minimization problem, *Minimize-Size*, can be formulated as follows.

Given a set of flows $F = \{f_j \propto (L_j, p_j, \sigma_j, \rho_j)\}$, routing matrix R, the maximum delay that each flow can suffer in the network $d = \{d_j\}$ for $\forall f_j \in F$, *find* the regulator parameters, peak rate p_{R_j} and traffic burstiness σ_{R_j} for $\forall f_j \in F$, such that

$$\min_{p_{R_j}, \sigma_{R_j}} \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) \tag{11}$$

subject to

$$D_{reg_j} + \bar{D}_j \le d_j \qquad \forall f_j \in F \tag{12}$$
$$\rho_j \le p_{R_j} \le p_j \qquad \forall f_j \in F \tag{13}$$
$$L_j \le \sigma_{R_j} \le \sigma_j \qquad \forall f_j \in F \tag{14}$$
$$\bar{B}_j > 0 \qquad \forall f_j \in F \tag{15}$$

where p_{R_j} and σ_{R_j} for $\forall f_j \in F$ are optimization variables.

Equation (11) is the objective function of this optimization problem which minimizes total buffer requirements. Constraint (12) says that the maximum delay of each flow j cannot exceed the maximum delay that it can suffer in the network d_j. Since we measured the flow performance in terms of its latency, we can consider d_j as a criterion of minimum guaranteed performance for flow j. Constraints (13) and (14) are related to two intervals $p_{R_j} \in [\rho_j, p_j]$ and $\sigma_{R_j} \in [L_j, \sigma_j]$ which called the regulation spectrum as described in Section III-B.

It is clear that by following the above mentioned equations, we can understand the effect of optimization variables on the objective function and all constraints of the defined problem.

In the literature, (11) is called a nonconvex NLP problem [16]. There are different methods for solving this kind of optimization problems. In particular, we will solve the optimization problem (11) using interior point method for constrained NLP problems [16], [17].

B. Buffer Variance Optimization

To reuse IP modules, designers would like to use similar switches as far as possible. However, flow requirements differ from each other in terms of buffer size; consequently, we would like to find appropriate peak rate and traffic burstiness of each flow so that variances of buffer size in the respective output buffers of switches are minimized. For example in a 2-D mesh network, we would like to minimize the variance of buffer size in northern output port of switches, as well as other output ports. Using general variance formula, we can easily calculate variances of the required buffer on each output port i which is denoted by var_i. Hence, we formulate another optimization problem to minimize the sum of required buffers variances while satisfying QoS requirements in the network. Thus, the buffer variance minimization problem, *Minimize-Variance*, can be formulated as follows.

Given a set of flows $F = \{f_j \propto (L_j, p_j, \sigma_j, \rho_j)\}$, routing matrix R, the maximum delay that each flow can suffer in the network $d = \{d_j\}$ for $\forall f_j \in F$, *find* the regulator parameters, peak rate p_{R_j} and traffic burstiness σ_{R_j} for $\forall f_j \in F$, such that

$$\min_{p_{R_j}, \sigma_{R_j}} \sum_i var_i \tag{16}$$

subject to

$$D_{reg_j} + \bar{D}_j \le d_j \qquad \forall f_j \in F \tag{17}$$
$$\rho_j \le p_{R_j} \le p_j \qquad \forall f_j \in F \tag{18}$$
$$L_j \le \sigma_{R_j} \le \sigma_j \qquad \forall f_j \in F \tag{19}$$
$$\bar{B}_j > 0 \qquad \forall f_j \in F. \tag{20}$$

Optimization variables are p_{R_j} and σ_{R_j}, $\forall f_j \in F$, that can be detected in the objective function and constraints by the following equations. Similar to (11), (16) also is a nonconvex NLP that can be solved via the interior point method.

C. Multiobjective Optimization Problem

As both of the aforementioned objective functions are worthwhile for designing the network, we formulate a multiobjective optimization problem which minimizes both total buffers and variances, *Multiobjective*, as follows.

Given a set of flows $F = \{f_j \propto (L_j, p_j, \sigma_j, \rho_j)\}$, routing matrix R, the maximum delay that each flow can suffer in the

network $d = \{d_j\}$ for $\forall f_j \in F$, *find* the regulator parameters, peak rate p_{R_j} and traffic burstiness σ_{R_j} for $\forall f_j \in F$, such that

$$\min_{p_{R_j},\sigma_{R_j}} f_1 = \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) \tag{21}$$

$$\min_{p_{R_j},\sigma_{R_j}} f_2 = \sum_{i} var_i \tag{22}$$

subject to

$$D_{reg_j} + \bar{D}_j \le d_j \qquad \forall f_j \in F \tag{23}$$

$$\rho_j \le p_{R_j} \le p_j \qquad \forall f_j \in F \tag{24}$$

$$L_j \le \sigma_{R_j} \le \sigma_j \qquad \forall f_j \in F \tag{25}$$

$$\bar{B}_j > 0 \qquad \forall f_j \in F. \tag{26}$$

In multiobjective optimizations, there is not even a universally accepted definition of *optimum* as in single-objective optimization, which makes it difficult to even compare results of one method to another, because normally the decision about what the *best* answer corresponds to the so-called *decision maker* [23]. Overall, there are different ways for solving multiobjective optimizations. One of them is combining objectives into a single function which normally denominated *Weighted Sum Approach*. Since objective functions in this paper are in the same direction and they are not in conflict with each other, we adopt this approach. The results in Section VII also confirm that the obtained solution of the proposed multiobjective problem is very close to optimal points of *Minimize-Size* and *Minimize-Variance* problems. This means that it is an appropriate method for solving this problem. The main advantage of this approach is the simplicity of its implementation and its computational efficiency. This method consists of adding all the objective functions together using weighting coefficients for each one of them. Specifically, our multiobjective problem is transformed into a scalar optimization problem of the form

$$min(w_1 f_1 + w_2 f_2) \tag{27}$$

where w_1 and w_2 are the weighting coefficients representing the relative importance of the objectives. In this paper, they are assumed the same. This approach has a low run-time complexity because of its simplicity and efficiency and therefore, can be applied for complex SoC designs. We solve the mentioned problem still using the interior point method.

VI. Optimization Method

A. Optimization Algorithm

As stated before, the proposed optimization problems are called nonconvex NLP problems [16] and solved by the interior point method. There are different packages for solving this kind of optimization problems and we particularly use the MATLAB optimization package in this paper.

To exemplify the optimization approach, we will solve the buffer size optimization problem (11), using the interior point method for constrained NLP problems [16], [17].

The interior point approach to constrained minimization is to solve a sequence of approximate minimization problems called *barrier* problem [17]. Due to (11), for each $\mu > 0$, the barrier problem is

$$\min_{p_{R_j},\sigma_{R_j},s_i} \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) - \mu \sum_{i=1}^{6|F|} ln(s_i) \tag{28}$$

subject to

$$D_{reg_j} + \bar{D}_j - d_j + s_i = 0 \quad \forall f_j \in F \quad i = 1, ..., |F| \tag{29}$$

$$\rho_j - p_{R_j} + s_i = 0 \quad \forall f_j \in F \quad i = |F| + 1, ..., 2|F| \tag{30}$$

$$p_{R_j} - p_j + s_i = 0 \quad \forall f_j \in F \quad i = 2|F| + 1, ..., 3|F| \tag{31}$$

$$L_j - \sigma_{R_j} + s_i = 0 \quad \forall f_j \in F \quad i = 3|F| + 1, ..., 4|F| \tag{32}$$

$$\sigma_{R_j} - \sigma_j + s_i = 0 \quad \forall f_j \in F \quad i = 4|F| + 1, ..., 5|F| \tag{33}$$

$$s_i - \bar{B}_j = 0 \quad \forall f_j \in F \quad i = 5|F| + 1, ..., 6|F| \tag{34}$$

where $|F|$ is the cardinality of set F.

There are as many slack variables s_i as inequality constraints (12)–(15). The s_i are restricted to be positive to keep $ln(s_i)$ bounded. As μ decreases to zero, the minimum of f_μ should approach the minimum of f. The approximate problem (28) is a sequence of equality constrained problems. These are easier to solve than the original inequality-constrained problem (11).

To facilitate our discussion, we define $p_R = (p_{R_1}, ..., p_{R_{|F|}})^T$, $\sigma_R = (\sigma_{R_1}, ..., \sigma_{R_{|F|}})^T$, $s = (s_1, ..., s_{6|F|})^T$ and assume $g(p_R, \sigma_R) = (g_1(p_R, \sigma_R), ..., g_{6|F|}(p_R, \sigma_R))^T$ so that $g(p_R, \sigma_R) + s$ is a vector that its elements are constraints (29)–(34). Thus, the barrier problem (28) can be rewritten as

$$\min_{p_R,\sigma_R,s} f_\mu(p_R, \sigma_R, s) = \min_{p_R,\sigma_R,s} f(p_R, \sigma_R) - \mu \sum_{i=1}^{6|F|} ln(s_i) \tag{35}$$

subject to

$$g(p_R, \sigma_R) + s = 0. \tag{36}$$

In the following, we shall find an approximate solution to (35), for fixed μ. Then, the used method is applied repeatedly to (35), for decreasing values of μ, to approximate the solution of the original problem (11).

Using the optimization methods [16], the Lagrangian of the problem (35) can be written as

$$L(p_R, \sigma_R, s, \lambda) = f(p_R, \sigma_R) - \mu \sum_{i=1}^{6|F|} ln(s_i) + \lambda^T (g(p_R, \sigma_R) + s) \tag{37}$$

where $\lambda = (\lambda_1, ..., \lambda_{6|F|})^T$ is the vector of Lagrange multipliers. Regarding the first-order optimality conditions, at an optimal solution (p_R, σ_R, s) of the barrier problem, we have

$$\nabla_{p_R} L(p_R, \sigma_R, s, \lambda) = \nabla_{p_R} f(p_R, \sigma_R) + A(p_R, \sigma_R)\lambda = 0 \tag{38}$$

$$\nabla_{\sigma_R} L(p_R, \sigma_R, s, \lambda) = \nabla_{\sigma_R} f(p_R, \sigma_R) + \acute{A}(p_R, \sigma_R)\lambda = 0 \tag{39}$$

$$\nabla_s L(p_R, \sigma_R, s, \lambda) = -\mu S^{-1} e + \lambda = 0 \tag{40}$$

where $A(p_R, \sigma_R) = (\nabla_{p_R} g_1(p_R, \sigma_R), ..., \nabla_{p_R} g_{6|F|}(p_R, \sigma_R))$ and $\acute{A}(p_R, \sigma_R) = (\nabla_{\sigma_R} g_1(p_R, \sigma_R), ..., \nabla_{\sigma_R} g_{6|F|}(p_R, \sigma_R))$ are the

matrixes of constraint gradients with respect to p_R and σ_R, respectively, and where

$$e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, S = \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_{6|F|} \end{pmatrix}.$$

To solve the approximate problem, we should generate a step d for displacement at an iterate z, where

$$d = \begin{pmatrix} d_{p_R} \\ d_{\sigma_R} \\ d_s \end{pmatrix}.$$

One of the two main types of steps is used at each iteration.

1) A *direct* step in (p_R, σ_R, s). This step attempts to solve the KKT equations for the approximate problem via a linear approximation. This is also called a Newton step [20].
2) A *conjugate gradient (CG)* step, using a trust region [21].

The algorithm first attempts to take a direct step. If it cannot, it attempts a CG step. One case where it does not take a direct step is when the approximate problem is not locally convex near the current iteration.

Afterward, it is necessary to decide if the step obtained from the abovementioned methods is acceptable. For this purpose, a merit function is introduced. The merit function is given by

$$\phi = \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) - \mu \sum_{i=1}^{6|F|} ln(s_i) + \nu \, \| g(p_R, \sigma_R) + s \| \quad (41)$$

where $\nu > 0$ is a *penalty parameter* and can increase with iteration number in order to force the solution toward feasibility.

The step is accepted if it gives sufficient reduction in the merit function; otherwise it is rejected. More details of the *direct* and *CG* steps are described in the following.

According to the above discussions, we present an iterative algorithm as the solution to (11). Algorithmic realization of the solution method is listed as Algorithm 1. In this respect, optimal peak rate and traffic burstiness for traffic flows can be found while minimizing total buffer requirements under performance constraints.

Algorithm 1: Buffer Size Minimization Algorithm

Initialization:

1. Choose a *penalty parameter* $\nu > 0$ and a *barrier parameter* $\mu > 0$.
2. Initialize trust region radius $R > 0$ and Lagrange multipliers λ.
3. Set an appropriate initial value for peak rate and burstiness of flows for problem (11) denoted as $p_R(0)$, $\sigma_R(0)$.
4. Specify an appropriate value for ϵ, $\acute{\epsilon}$ ($\acute{\epsilon}$ denote value of expectable reduction in merit function).

1. *Loop* 1: Do until $(\max \mid p_R(t+1) - p_R(t) \mid < \epsilon) \& (\max \mid \sigma_R(t+1) - \sigma_R(t) \mid < \epsilon)$

2. Set an appropriate initial value for peak rate and burstiness of flows and slack variables for barrier problem (35) denoted as $\acute{p}_R(0)$, $\acute{\sigma}_R(0)$, $s(0)$.

3. *Loop* 2: Do until $(\max \mid \acute{p}_R(k+1) - \acute{p}_R(k) \mid < \epsilon) \& (\max \mid \acute{\sigma}_R(k+1) - \acute{\sigma}_R(k) \mid < \epsilon)$

4. if H is not definite positive go to 5
   4.1. Calculate d based on *Direct Step* as described in Section VI-B
   4.2. Go to 6.

5. Calculate d based on *CG Step* as described in Section VI-C

6. $p_{temp} = \acute{p}_R(k) + d_{p_R}$;
7. $\sigma_{temp} = \acute{\sigma}_R(k) + d_{\sigma_R}$;
8. $s_{temp} = \acute{s}(k) + d_s$

9. Calculate $\phi(k+1)$ by substituting p_{temp}, σ_{temp}, s_{temp} in merit problem (41).

10. if $(\phi(k+1) - \phi(k) \geq \acute{\epsilon})$
    10.1. Decrease R;
    10.2. Go to 4;

11. $p_R(k+1) = p_{temp}$; $\sigma_R(k+1) = \sigma_{temp}$; $s(k+1) = s_{temp}$

12. Compute new Lagrange multipliers λ.
13. End of loop 2.

14. Decrease *barrier parameter* μ.
15. End of loop 1.

Output:
Communicate optimal peak rates and traffic burstinesses to the corresponding regulators.

B. Direct Step

This step attempts to solve the KKT equations for the barrier problem via a linear approximation. Regarding the KKT conditions for the equality constrained barrier problem (35), we have

$$\begin{pmatrix} \nabla_{p_R} f(p_R, \sigma_R) + A(p_R, \sigma_R)\lambda \\ \nabla_{\sigma_R} f(p_R, \sigma_R) + \acute{A}(p_R, \sigma_R)\lambda \\ -\mu S^{-1} e + \lambda \\ g(p_R, \sigma_R) + s \end{pmatrix} = 0. \quad (42)$$

After applying Newton's method to this system, we have

$$\begin{pmatrix} \nabla^2_{p_R,p_R} L & \nabla^2_{p_R,\sigma_R} L & 0 & A(p_R,\sigma_R) \\ \nabla^2_{\sigma_R,p_R} L & \nabla^2_{\sigma_R,\sigma_R} L & 0 & \acute{A}(p_R,\sigma_R) \\ 0 & 0 & \mu S^{-2} & I \\ A(p_R,\sigma_R) & \acute{A}(p_R,\sigma_R) & I & 0 \end{pmatrix} \begin{pmatrix} d_{p_R} \\ d_{\sigma_R} \\ d_s \\ \lambda^+ \end{pmatrix}$$

$$= \begin{pmatrix} \nabla_{p_R} f(p_R, \sigma_R) \\ \nabla_{\sigma_R} f(p_R, \sigma_R) \\ \mu S^{-1} e \\ -g(p_R, \sigma_R) - s \end{pmatrix} \quad (43)$$

where $\lambda^+ = \lambda + d_\lambda$. Thus, steps d_{p_R}, d_{σ_R} and d_s can be calculated by solving (43). Letting H be the Hessian of the Lagrangian of the barrier problem, we have

$$H = \begin{pmatrix} \nabla^2_{p_R,p_R} L & \nabla^2_{p_R,\sigma_R} L & 0 \\ \nabla^2_{\sigma_R,p_R} L & \nabla^2_{\sigma_R,\sigma_R} L & 0 \\ 0 & 0 & \mu S^{-2} \end{pmatrix}. \quad (44)$$

If the barrier problem is locally convex near the current iteration, i.e., H is positive definite, the algorithm uses this step; otherwise, it uses a CG step, described in the next section.

C. Conjugate Gradient (CG)

The CG approach to solving the approximate problem (35) is similar to other CG calculations. In this case, the algorithm adjusts p_R, σ_R, and s, keeping the slacks s positive.

The approach is to minimize a quadratic approximation to the barrier problem in a trust region, subject to linearized constraints.

The algorithm obtains Lagrange multipliers by approximately solving the KKT equations, subject to λ being positive. Then it takes a step $d = (d_{p_R}, d_{\sigma_R}, d_s)^T$ to approximately solve

$$\min_d \nabla f_\mu^T d + \frac{1}{2} d^T H d \tag{45}$$

subject to

$$(A(p_R, \sigma_R)^T I) d_{p_R} + (\acute{A}(p_R, \sigma_R)^T I) d_{\sigma_R} + g(p_R, \sigma_R) + s = 0$$

where ∇f_μ is the gradient of the barrier problem and is given by

$$\nabla f_\mu = \begin{pmatrix} \nabla_{p_R} f(p_R, \sigma_R) \\ \nabla_{\sigma_R} f(p_R, \sigma_R) \\ -\mu S^{-1} e \end{pmatrix}. \tag{46}$$

To obtain convergence from remote starting points, we introduce a trust region constraint in (45) of the form

$$\left\| \begin{pmatrix} d_{p_R} \\ d_{\sigma_R} \\ S^{-1} d_s \end{pmatrix} \right\| \leq R \tag{47}$$

where $R > 0$ denotes the trust region radius and is updated at every iteration.

To solve (46), the algorithm tries to minimize a norm of the linearized constraints inside a region with radius scaled by R. Then (45) is solved with the constraints being to match the residual from solving (46), staying within the trust region of radius R, and keeping s strictly positive. Since it is not desirable to impede progress of the iteration by employing small trust regions, the slack variables are bounded away from zero by imposing the well-known fraction to the boundary rule [22]

$$s + d_s \geq (1 - \tau) s$$

where the parameter $\tau \in (0, 1)$ is chosen close to 1. Therefore, (45) can be rewritten as follows:

$$\min_d \nabla f_\mu^T d + \frac{1}{2} d^T H d \tag{48}$$

subject to

$$A(p_R, \sigma_R)^T I d_{p_R} + \acute{A}(p_R, \sigma_R)^T I d_{\sigma_R} + g(p_R, \sigma_R) + s = 0 \tag{49}$$

$$\left\| (d_{p_R}, d_{\sigma_R}, S^{-1} d_s) \right\| \leq R \tag{50}$$

$$d_s \geq -\tau s. \tag{51}$$

Although, (48) could be difficult and complex to solve exactly, but we intend to only compute approximate solutions which are sufficiently good solutions [17].

Further details about the optimization method can be found in [17], [20], and [21].

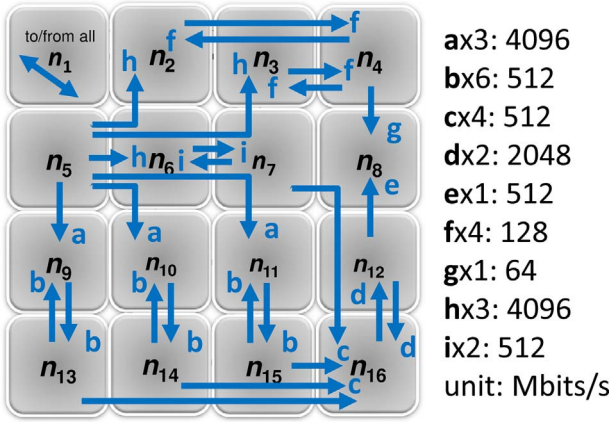

Fig. 10. Ericsson radio systems application.

VII. Experimental Results

A. Experimental Setup

To evaluate the capability of our method, we applied it to a realistic traffic pattern and two synthetic traffic patterns including hot-spot and bit-complement which are mapped to a 4×4 2-D mesh network. Although the experiments are performed on a mesh, our method is topology independent.

In this paper, the proposed analytical model is implemented in MATLAB and throughout the experiments, we consider an SoC with 500 MHz frequency, 32-flit packets, and 32-bit flits. We also assume that packets traverse the network on a shortest path using the dimension order XY routing, which is deadlock free.

B. Realistic Traffic Pattern

We used a real application provided by Ericsson Radio Systems [1] as shown in Fig. 10. This application consists of 16 IPs. Specifically, n_2, n_3, n_6, n_9, n_{10}, and n_{11} are ASICs; n_1, n_7, n_{12}, n_{13}, n_{14}, and n_{15} are DSPs; n_5, n_8, and n_{16} are FPGAs; n_1 is a device processor which loads all nodes with program and parameters at startup, sets up, and controls resources in normal operation. Traffic to/from n_1 is for system initial configuration and no longer used afterward. There are 26 node-to-node traffic flows that are categorized into nine types of traffic flows $\{a, b, c, d, e, f, g, h, i\}$, as marked in the figure. The traffic flows are associated with a bandwidth requirement.

As stated before, each flow j is characterized by $(L_j, p_j, \sigma_j, \rho_j)$ that are input parameters of the regulator. We assume L_j and p_j for all flows are the same and equal to 1 *flit* and 1 *flit/cycle*, respectively. ρ_j is determined in *flits/cycle* due to Fig. 10 and also, σ_j can be easily calculated for each flow which its value will be shown in Section VII-B3.

1) *Buffer Size Optimization:* As we mentioned before, a regulator limits a flow injection process with two parameters (peak rate and burstiness). Since there are 26 flows in the example, 52 parameters have to be assigned to regulators. To show that how these parameters heavily affect the required buffer and communication delay, we consider two different regulator sets.

1) Optimized regulators, which are optimized based on the proposed minimizing buffer problem (11).

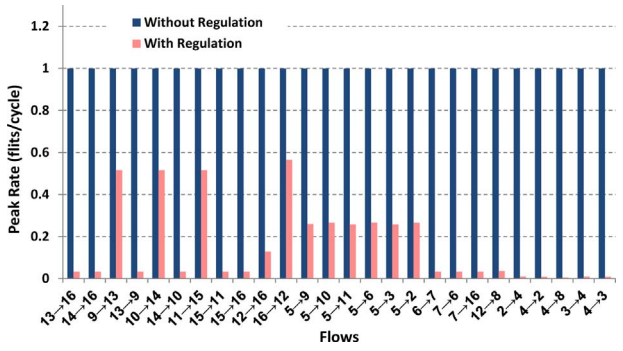

Fig. 11. Peak rate of flows.

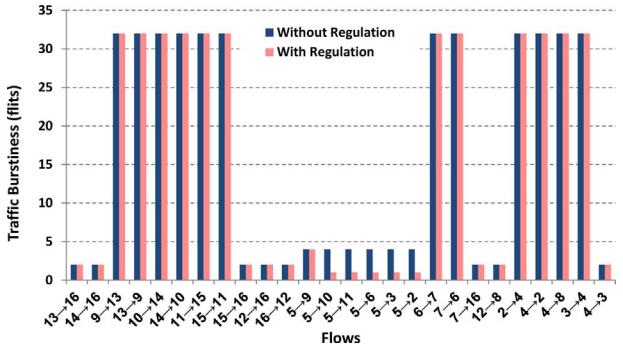

Fig. 12. Traffic burstiness of flows.

TABLE I

COMPARISON OF THE REQUIRED BUFFER BETWEEN DIFFERENT SCHEMES

	Network Buffer	Regulator Buffer	Total Buffer
Without reg.	404	0	404
Optimized reg.	118	28	146
Unoptimized reg.	384	37	421

TABLE II

COMPARISON OF THE MAXIMUM DELAY BETWEEN DIFFERENT SCHEMES

	Network Delay	Regulator Delay	Total Delay
Without reg.	3460	0	3460
Optimized reg.	502	61	563
Unoptimized reg.	3396	163	3559

2) Unoptimized regulators, which are not optimized. Obviously, there is a huge number of unoptimized configurations. We consider a configuration that needs maximum amount of buffers to regulate flows. In fact, we modify the buffer optimization problem (11) to maximize the total number of required buffers instead of minimization.

Then, the total maximum buffer and total maximum delay are calculated and depicted in Tables I and II, respectively, along with values for a system without regulators.

From these tables, we can see that the optimized regulation scheme leads to about 64% reduction in total maximum required buffer and about 84% in total maximum delay when compared with the without regulation scheme. Also these tables show that unoptimized regulators decrease the maximum required buffer and delay in the network because of reducing the contention for shared resources. However, buffer and delay in the regulators are increased to the extent that the total buffer requirements and delay become more than the without regulation scheme because the regulator parameters are not configured appropriately. As a result, we can minimize total buffer cost and improve communications performance by consuming a few buffers in the regulator and assigning the peak and burstiness parameters of regulators in a wise manner.

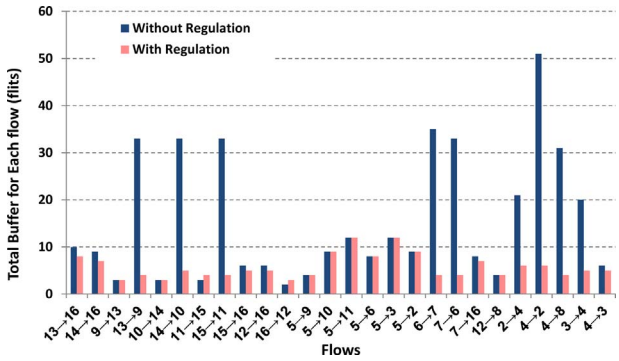

Fig. 13. Maximum required buffers for every flow.

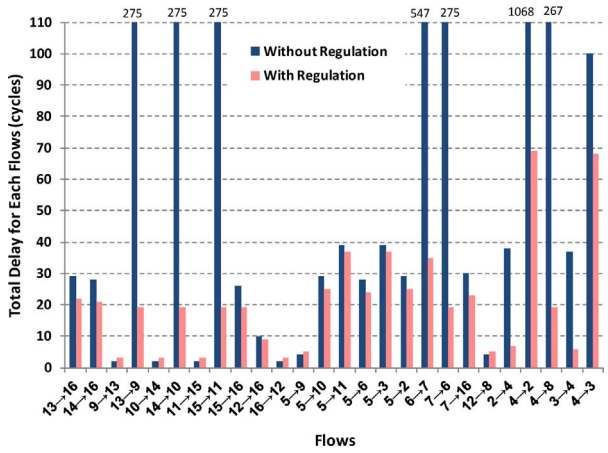

Fig. 14. Maximum worst-case delay for every flow.

TABLE III

COMPARISON BETWEEN DIFFERENT SCENARIOS

	Required Buffer *(flits)*	Variance	
Without regulation	404	436.36	
Minimize-size	146	33.82	
Minimize-variance	192	22.96	
Multiobjective	150	24.29	

2) *Buffer Variance Optimization:* Identical switches throughout the network may be a constraint in NoC-based systems. Therefore, we have formulated the *Minimize-Variance* optimization problem to design similar switches as far as possible. The results show that if there is no regulator in the network, the sum of variances over different channels of switches is about 436.36, while by controlling flows based on obtained output peak rate and traffic burstiness of solving the *Minimize-Variance* problem, it is equal to 22.96. So, we have about 94% reduction on the sum of variances of buffers.

In this respect, the structures of latter switches are more similar than the former one. It is worth mentioning that if the peak and burstiness parameters of regulators are not appropriately assigned with respect to buffer variance minimization, we may have similar or even more buffer variance in comparison to *without regulation* scheme. For instance, in one of the unoptimized schemes, the sum of variances over different channels of switches is about 436.

3) *Multiobjective Optimization:* As both minimizing total required buffer and buffer variance are important for designers,

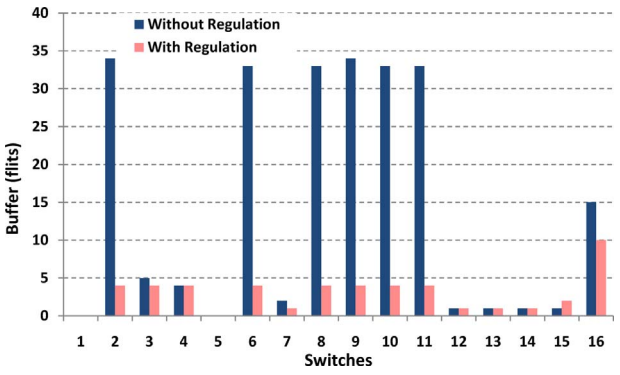

Fig. 15. Maximum required buffers for the ejection channels in switches.

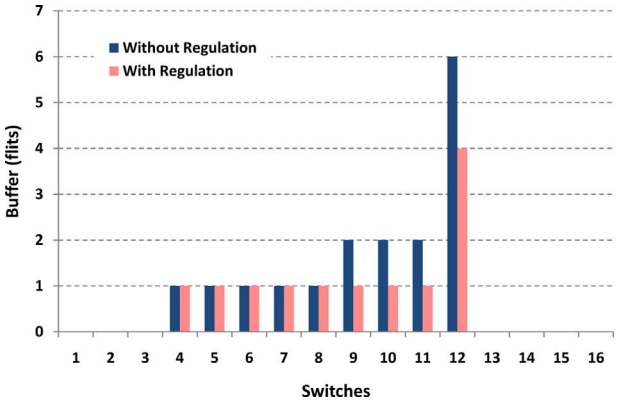

Fig. 16. Maximum required buffers for the southern channels in switches.

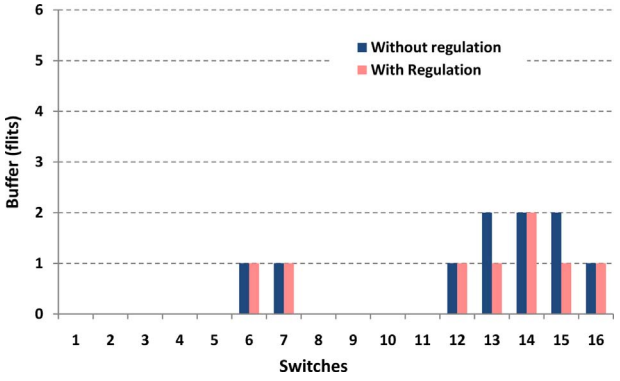

Fig. 17. Maximum required buffers for the northern channels in switches.

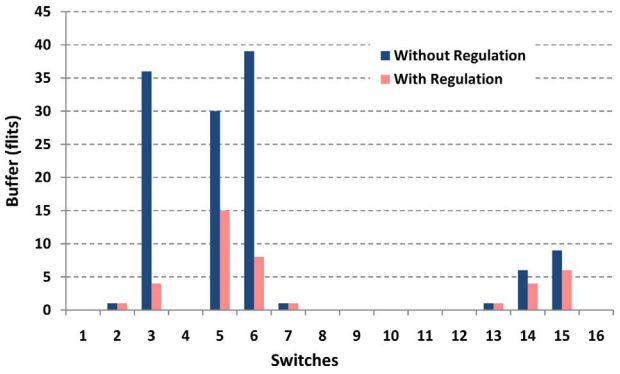

Fig. 18. Maximum required buffers for the eastern channels in switches.

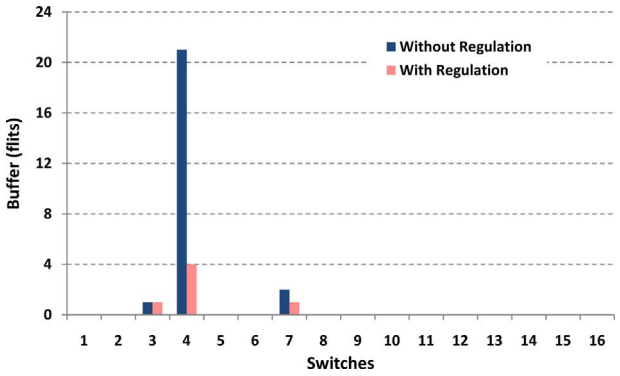

Fig. 19. Maximum required buffers for the western channels in switches.

TABLE IV
COMPARISON OF THE MAXIMUM DELAY BETWEEN DIFFERENT SCENARIOS

	Network Worst-Case Delay	Regulator Worst-Case Delay	Total Worst-Case Delay	Average Worst-Case Delay
Without regulation	3460	0	3460	49.99
With regulation	463	81	544	21.70

we have modeled them as a multiobjective optimization problem. For more detail, we have calculated two parameters *Total Required Buffer* and *Variance* which are listed in Table III.

As can be observed from Table III, *Minimize-Size* problem guarantees that output peak and traffic burstiness selection is carried out in favor of minimizing total required buffer while there is no such guarantee for the sum of variances over various channels. On the contrary, although *Minimize-Variance* yields greater required buffer than *Minimize-Size*, it gives almost the same structure of switches. The results in Table III show that the presented *Multiobjective* problem might be seen as providing a tradeoff between such parameters. Since the *Total Required Buffer* and *Variance* parameters in this problem are very close to their optimal values in *Minimize-Size* and *Minimize-Variance* problems, respectively, they are definitely acceptable for the decision maker. So, in the rest of paper, *with regulation* scheme refers to the regulator which has been optimized for both buffer size and variance.

As can be vividly seen in Figs. 11 and 12, regulators reduce peak rate and traffic burstiness of flows, respectively.

To go into more detail, we depict maximum required buffer and delay of each flow for these schemes in Figs. 13 and 14, respectively. Regarding Fig. 13, it is apparent that in the network with the proposed regulator, most flows require less buffer and also, as mentioned in Table III, total required buffer in this scheme is less than half of it in the network without regulator. Also, Fig. 14 shows that regulated flows can experience longer or shorter delays than other schemes which depends on their requested QoS and also the buffer distribution in the whole network. However, from Table IV, we can see that the total network and average worst-case delay are decreased in the *with regulation* scheme because of buffer-aware allocation in the network and contention reduction for shared resources. We have about 84.3% reduction in total worst-case delay when compared with the *without regulation* scheme.

To better understand the effects of the regulator, maximum required buffers for ejection, southern, northern, eastern, and western channels are revealed in Figs. 15–19, respectively. It is obvious that when regulators control traffic parameters of flows based on the proposed multiobjective problem, the total number of required buffers and their variances are decreased. The *with regulation* scheme leads to about 62.8% reduction in total required buffer and 94.4% reduction on the sum of variances of buffers in comparison to the *without regulation* scheme. So, we have smaller, more similar and more efficient switches. Furthermore, there is desirable QoS in communications through defined constraints in the mentioned multiobjective problem.

C. Synthetic Traffic Patterns

In the case of synthetic traffic patterns, we experimented with hotspot and bit-complement traffic, which represent two extremes of traffic distribution, i.e., unbalanced and balanced workloads.

1) *Hotspot:* in our case, we set a corner node of the 4×4 mesh, node 1, as the hotspot node, and all other nodes send packets to this node.
2) *Bit-complement:* in bit-complement traffic, a node with binary coordinates $b_{n-1}b_{n-2}\ldots b_1b_0$ sends packets only to a node with binary coordinates $\bar{b}_{n-1}\bar{b}_{n-2}\ldots\bar{b}_1\bar{b}_0$. With this workload, all packets must cross the horizontal and vertical network bisections, and the traffic is evenly distributed in the 4×4 network.

For all traffic flows, we set the same values for their maximum packet length L_j and peak rate p_j, which are equal to 1 *flit* and 1 *flit/cycle*, respectively. For different flows, rate ρ_j varies between 0.008 and 1 *flits/cycle*, and burstiness σ_j between 2 and 32 *flits*. We apply the multiobjective optimization here, which is referred to as *with regulation* scheme. Compared with the optimization of single objectives, it is likely more desirable for designers as it can optimize both buffer size and variance,

Table V compares total maximum required buffer, variance, and total maximum delay under the hotspot traffic pattern. This table reveals that by using optimized regulators, the total maximum required buffer, the variance, and the total maximum delay are reduced by 45.4%, 84.3%, and 58.4%, respectively, in comparison with the *without regulation* scheme.

We also compare these results under the bit-complement traffic pattern in Tables VI. As can be seen from this table, the optimized regulation results in about 49.6% reduction in the total maximum required buffer, 95.1% reduction in the variance, and 64.9% reduction in the total maximum delay.

To present more details, we show the maximum required buffer and delay of each flow under the hotspot traffic in Figs. 20 and 21, respectively. Also, these results under the bit-complement are plotted in Figs. 22 and 23.

The run-time of the proposed method in MATLAB is typically in the order of a few seconds. It is about 2.7 s, 5.76 s, and 0.22 s for the multiobjective optimization of the realistic, hotspot, and bit-complement traffic patterns, respectively. Another interesting point is that the proposed regulator has no negative effect on the network throughput and it is the same with and without the regulation schemes. This is because the flow rates are maintained.

TABLE V
COMPARISON BETWEEN DIFFERENT SCENARIOS UNDER HOTSPOT TRAFFIC

	Network Buffer	Regulator Buffer	Total Buffer	Variance
Without regulation	361	0	361	830.4023
With regulation	144	53	197	129.7305
	Network Worst-Case Delay	Regulator Worst-Case Delay	Total Worst-Case Delay	Average Worst-Case Delay
Without regulation	3328	0	3328	89.10
With regulation	789	597	1386	53.68

TABLE VI
COMPARISON BETWEEN DIFFERENT SCENARIOS UNDER BIT-COMPLEMENT TRAFFIC

	Network Buffer	Regulator Buffer	Total Buffer	Variance
Without regulation	254	0	254	178.73
With regulation	112	16	128	8.72
	Network Worst-Case Delay	Regulator Worst-Case Delay	Total Worst-Case Delay	Average Worst-Case Delay
Without regulation	410	0	410	28.10
With regulation	128	16	144	9.23

VIII. SCOPE AND ASSUMPTION

We discuss possible extensions to address the main assumptions of our approach. We have made two main assumptions.

1) The network routing is deterministic. As such, the path of each flow is determined and thus flow contention becomes predictable. Therefore we can use and have used deterministic network calculus to derive deterministic delay and backlog bounds.
   Deterministic routing has advantages in easier analysis, simplicity, and low implementation overhead. However, it may lead to inferior performance due to being unable to adapt workload to the network congestion status. Due to this limitation, adaptive routing may be favored, though complicating implementation. Adaptive routing means that a flow may use multiple possible paths when delivering packets. For each alternative path, one may find a probability for its use. In such a case, stochastic network calculus [24] can be used to calculate delay and backlog bounds. Still, stochastic network calculus keeps the same fundamentals as the deterministic network calculus. However, the derived delay and backlog bounds will accordingly become stochastic.
2) We assume a static set of flows, which are mapped statically on the network nodes.
   The reason to use static flows with static mapping is that the deterministic analysis relies on known traffic characteristics and known source and destination for each flow. Flows' characteristics may be obtained through traffic profiling. Static mapping can usually facilitate the search of mapping design space in order to find an optimal or near-optimal mapping under performance and energy constraints [10]. As a consequence, the static flows and mapping allow us to apply static regulations on the flows.

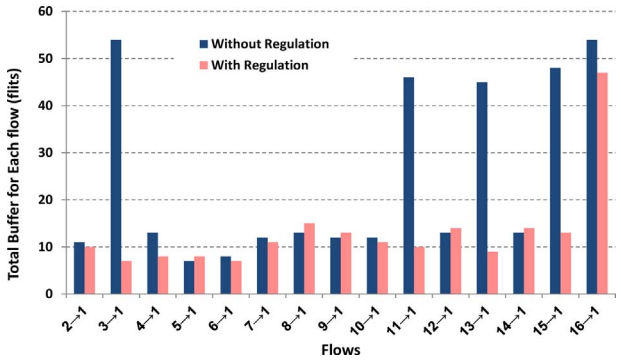

Fig. 20. Maximum required buffers for every flow under hotspot traffic.

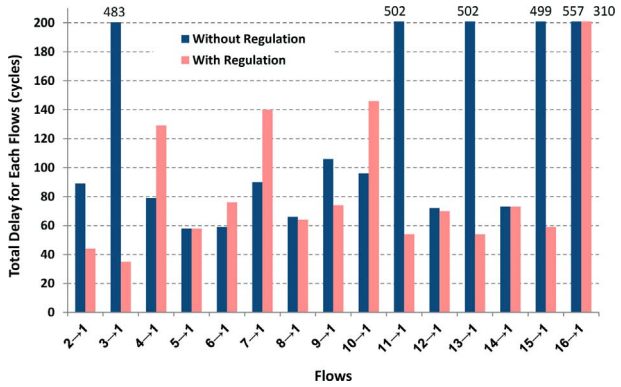

Fig. 21. Maximum worst-case delay for every flow under hotspot traffic.

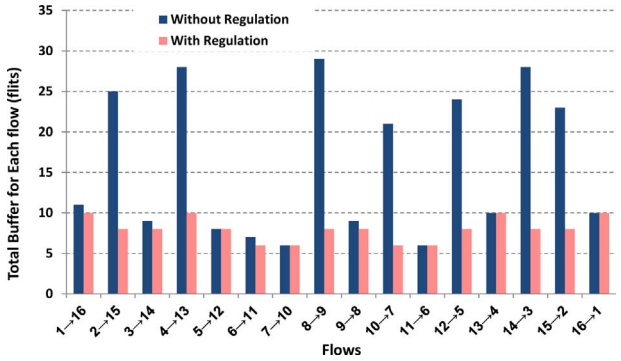

Fig. 22. Maximum required buffers for every flow under Bit-complement.

To alleviate this assumption, there are a few possibilities to enable semi-dynamic and dynamic regulations as we explained as follows.

1) *Semi-dynamic regulation:*
   a) Dynamically changing traffic specifications for each input flow. If a flow's traffic specification may change, we may prepare a set of variants for its parameters. Depending on different traffic specifications, different regulations for the same flow may apply at run-time.
   b) Different use cases and mappings. An application usually contains multiple use cases [25]. For each use case, a set of flows with possible mappings can be pre-compiled. All the use cases must fit into the maximum buffer sizes. These use cases can then be invoked and switched at run-time by reconfiguring the regulators and the network.

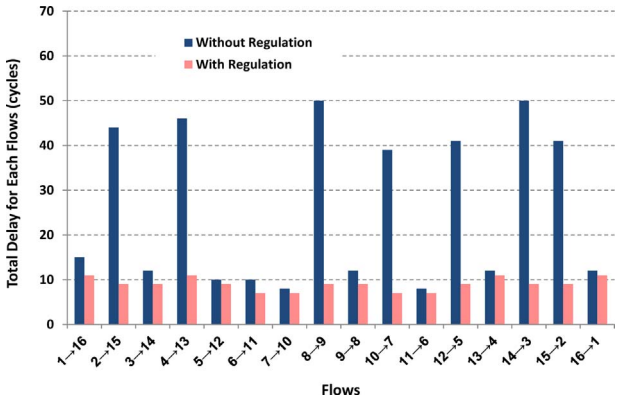

Fig. 23. Maximum worst-case delay for every flow under Bit-complement.

Semi-dynamic configurations can be realized by checking user-defined values of a configurable register in the network interface. Our current regulator implementation in hardware supports re-configuration of regulation parameters at run-time [7].

2) *Dynamic regulation:* we can embed a closed-loop control mechanism in which the network feedback is used as an input to help make regulation decisions. For example, network congestion status could be gathered from the network and then the regulation parameters are adjusted accordingly. This mechanism complicates the regulation mechanisms but has promises in improving performance. In addition, best effort traffic, i.e., traffic without the requirement of delay guarantees, can be better accommodated by allowing them to use the slack bandwidth. The closed-loop control mechanism is currently under our investigation.

IX. Conclusion

IP integration requires the provision of performance guarantees for traffic flows and efficient buffer dimensioning techniques. The regulation changes the burstiness and timing of traffic flows, and thus can be used to control delay and reduce buffer requirements in the SoC. Since a larger fraction of the NoC cost is due to the network buffers, minimizing buffer requirements is an important problem to achieve an efficient NoC implementation. Also, designing similar switches, as far as possible, facilitates the design process of NoC-based systems. In this paper, based on the concepts of formal regulation, we have presented three relevant optimization problems for weighted round robin arbitration, first one for minimizing total required buffers, second one for minimizing the variance of buffers, and last one which is a multiobjective optimization problem for minimizing both of them under QoS requirements. The regulation analysis is performed for best-effort packet switching networks. We have also demonstrated that the proposed model exerts significant impact on communication performance and buffer requirements. The algorithm for solving the proposed minimization problems runs very fast. For the case studies, the optimized solution is found within seconds. Although in this paper we have focused on the output

buffers of switches, our method can be easily adapted to input buffers, too.

REFERENCES

[1] Z. Lu and A. Jantsch, "TDM virtual-circuit configuration for network-on-chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 8, pp. 1021–1034, Aug. 2008.

[2] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. van der Wolf, and T. Henriksson, "Flow regulation for on-chip communication," in *Proc. DATE*, Apr. 2009, pp. 578–581.

[3] J. Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet* (LNCS, vol. 2050). Berlin, Germany: Springer-Verlag, 2004.

[4] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation; part II: Network analysis," *IEEE Trans. Inform. Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.

[5] D. Stiliadis and A. Varma, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 611–624, Oct. 1998.

[6] C. Chang, *Performance Guarantees in Communication Networks*. London, U.K.: Springer-Verlag, 2000, p. 410.

[7] Z. Lu, D. Brachos, and A. Jantsch, "A flow regulator for on-chip communication," in *Proc. SOCC*, 2009, pp. 151–154.

[8] H. Wang, X. Zhu, L. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proc. MICRO*, 2002, pp. 294–305.

[9] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse, "System architecture evaluation using modular performance analysis: A case study," *Int. J. STTT*, vol. 8, no. 6, pp. 649–667, 2006.

[10] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. DATE*, 2004, pp. 896–901.

[11] A. E. Kiasari, S. Hessabi, and H. Sarbazi-Azad, "PERMAP: A performance-aware mapping for application-specific SoCs," in *Proc. ASAP*, 2008, pp. 73–78.

[12] A. Jalabert, S. Murali, L. Benini, and G. De Micheli, "xPipesCompiler: A tool for instantiating application-specific NoCs," in *Proc. DATE*, 2004, pp. 884–889.

[13] L. P. Tedesco, N. Calazans, and F. Moraes, "Buffer sizing for multimedia flows in packet-switching NoCs," *J. Integr. Circuits Syst.*, vol. 3, no. 1, pp. 46–56, 2008.

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

[15] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Optimal regulation of traffic flows in network-on-chip," in *Proc. DATE*, Mar. 2010, pp. 1621–1624.

[16] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[17] H. Y. Benson, R. J. Vanderbei, and D. F. Shanno, "Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions," *Computat. Optimiz. Applicat.*, vol. 23, no. 2, pp. 257–272, 2002.

[18] P. P. Tang and T. Y. C. Tai, "Network traffic characterization using token bucket model," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 51–62.

[19] F. Gebali and H. Elmiligi, Eds., *Networks on Chip: Theory and Practice*. Boca Raton, FL: CRC Press, 2009.

[20] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, Society for Industrial and Applied Mathematics (SIAM), 2000.

[21] L. M. Adams and J. L. Nazareth, *Linear and Nonlinear Conjugate Gradient-Related Methods*, Society for Industrial and Applied Mathematics (SIAM), 1996.

[22] M. H. Wright, "Interior methods for constrained optimization," *Acta Numerica*. vol. 1, pp. 341–407, Jan. 1992.

[23] C. A. Coello Coello, "A comprehensive survey of evolutionary based multiobjective optimization techniques," *Knowl. Inform. Syst.: An Int. J.*, vol. 1, no. 3, pp. 269–308, 1999.

[24] Y. Jiang, "A basic stochastic network calculus," in *Proc. Conf. Applicat., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, 2006, pp. 123–134.

[25] A. Hansson and K. Goossens, "Tradeoffs in the configuration of a network on chip for multiple use-cases," in *Proc. 1st Int. Symp. NoCs*, 2007, pp. 233–242.

Fahimeh Jafari received the B.S. and M.S. degrees in computer engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2002 and 2005, respectively. She is currently pursuing the Ph.D. degree from the Department of Electronic Systems, Royal Institute of Technology, Kista, Stockholm, Sweden.

Her current research interests include design methodologies, interconnection networks, optimization theory, and performance evaluation.

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

From 1989 to 2000, he worked extensively on the areas of electronic and embedded systems. He took research visits to Samsung Electronics, Seoul, Korea, the National Institute of Informatics, Tokyo, Japan, and the Swiss Federal Institute of Technology, Zürich, Switzerland. He is currently a Senior Researcher with the Department of Electronic Systems, School of Information and Communication Technology, KTH. His current research interests include network-on-chip/system-on-chip, multicore computing architectures, cyber-physical systems, performance analysis, and design automation. He has published about 70 papers in these areas.

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

He was with Siemens Austria, Vienna, Austria, as a System Validation Engineer from 1995 to 1997. Since 1997, he has been an Associate Professor with the Royal Institute of Technology (KTH), Kista, Stockholm, Sweden. Since 2000, has been a Docent, and since December 2002, a Full Professor of Electronic System Design with the Department of Electronic Systems. He has published over 200 papers in international conferences and journals, and one book in the areas of very large scale integration design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks on chip.

Dr. Jantsch received the Alfred Schrödinger Scholarship from the Austrian Science Foundation while a Guest Researcher with KTH between 1993 and 1995. He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES+ISSS, SOC, NOCS, and others. He has been the TPC Chair of SSDL/FDL 2000, the TPC Co-Chair of CODES+ISSS 2004, the General Chair of CODES+ISSS 2005, and the TPC Co-Chair of NOCS 2009. From 2002 to 2007, he was a Subject Area Editor for the *Journal of System Architecture*. At KTH, he is heading a number of research projects involving a total number of ten Ph.D. Students, in two main areas: system modeling and networks-on-chip.

Mohammad Hossein Yaghmaee (M'09) was born in Mashhad, Iran, in July 1971. He received the B.S. degree in communication engineering from the Sharif University of Technology, Tehran, Iran, in 1993, and the M.S. and Ph.D. degrees in communication engineering from the Tehran Polytechnic (Amirkabir) University of Technology, Tehran, in 1995 and 2000, respectively.

Since 1992, he has been a Computer Network Engineer with several networking projects at the Iran Telecommunication Research Center, Tehran, Iran. From November 1998 to July 1999, he was a Visiting Research Scholar with the Network Technology Group, C&C Media Research Laboratories, NEC Corporation, Tokyo, Japan. From September 2007 to August 2008, he was a Visiting Associate Professor with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown. He is currently an Associate Professor with the Computer Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad. He is the author of four books, all in Farsi. He has published more than 90 international conference and journal papers. His current research interests include wireless sensor networks, traffic and congestion control, high-speed networks including ATM and MPLS, quality of services, and fuzzy logic control.