Optimal Regulation of Traffic Flows in Networks-on-Chip

Fahimeh Jafari*†, Zhonghai Lu†, Axel Jantsch† and Mohammad H. Yaghmaee*
*Ferdowsi University of Mashhad, Iran
†Royal Institute of Technology (KTH), Sweden
Email: fjafari@wali.um.ac.ir, {zhonghai, axel}@kth.se, hyaghmae@ferdowsi.um.ac.ir

***Abstract*—We have proposed (σ, ρ)-based flow regulation to reduce delay and backlog bounds in SoC architectures, where σ bounds the traffic burstiness and ρ the traffic rate. The regulation is conducted per-flow for its peak rate and traffic burstiness. In this paper, we optimize these regulation parameters in networks on chips where many flows may have conflicting regulation requirements. We formulate an optimization problem for minimizing total buffers under performance constraints. We solve the problem with the interior point method. Our case study results exhibit 48% reduction of total buffers and 16% reduction of total latency for the proposed problem. The optimization solution has low run-time complexity, enabling quick exploration of large design space.**

I. Introduction

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; (3) due to 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.

The admission of traffic flows from source IPs into the SoC infrastructure can be controlled by a regulator rather than injecting them as soon as possible [1]. In this way, we can control Quality-of-Service (QoS) and achieve cost-effective communication. To lay a solid foundation for our approach, flow regulation has been based on network calculus [2]. By importing and extending the analytical methods from network calculus, we can obtain worst-case delay and backlog bounds. In [3], 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 an optimization problem.

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; consequently, the size of buffers in the system should be minimized. On the other hand, buffers should be large enough to obtain predictable performance. It means that, there is a trade-off 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. Finally, we show the benefits of the proposed method and quantify performance improvement and buffer size reduction.

The remainder of this paper is organized as follows. Section II gives account of related works. In Section III, we introduce the flow regulation concept along with the basics of *Network Calculus*. Section IV discusses the underlying system model. Section V formulates the minimizing buffer optimization problem. Our simulation results are described in Section VI. Finally, Section VII gives the conclusions.

II. Related Work

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 [4] show the advantage of the topological mapping of IPs on the NoC architectures. In [5], the network topology customization and its effects on the system are considered. In [6], 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 appropriate for a system with hard real-time requirements.

The presented work in this paper follows a different direction by addressing an optimization problem to find the minimum total buffering requirements while satisfying acceptable communication performance. Also, our method is presented based on tight worst-case bounds derived by network calculus. Therefore, it is suitable for real-time system designs.

III. The Concepts of Flow Regulation

A. Network Calculus Basics

In network calculus [2], 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 $(\sigma_j \geq L_j)$, and ρ_j the average (sustainable) rate that we denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. The burstiness also is a 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.

The abstraction of service curve is used in Network calculus to model a network element processing traffic flows. 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 [2]. Notation $x^+ = x$ if $x > 0; x^+ = 0$, otherwise.

According to [2], the maximum delay and the buffer required for flow j are bounded by Eq. (1) and (2), respectively.

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

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

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

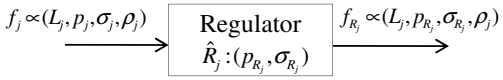

Fig. 1. Flow regulation

TSPEC can also be used to define a traffic regulator. Fig. 1 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 $\sigma_{R_j} \in [L_j, \sigma_j]$ are called the *regulation spectrum*, where the former is for the regulation of peak rate and the latter for the regulation of traffic burstiness. We implemented microarchitecture of the regulator and quantified its hardware speed and cost in [3]. Selecting appropriate p_{R_j} and σ_{R_j} is very effective in performance and cost of communications.

IV. System Model

A. Assumptions and Notations

We consider an 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. NIs provide an interface between IPs and the network. Note that the presence of NIs is the consequence of using a network not regulators. Regulators are inserted between the source IP and NI and their number is the same as the number of flows originating from that node. We presume the number of Virtual Channels (VCs) for each Physical Channel (PC) is the same as the number of flows passing through that channel. Fig. 2 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. Although in this paper we have focused on the output buffers of switches, our method can be easily adapted to input buffers, too. We also assume that the NoC architecture is lossless, and packets traverse the network in a best-effort fashion using a deterministic routing.

We consider NoC as a network with a set of bidirectional channels L, a set of sources S 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 $i \in L_{f_j}$.

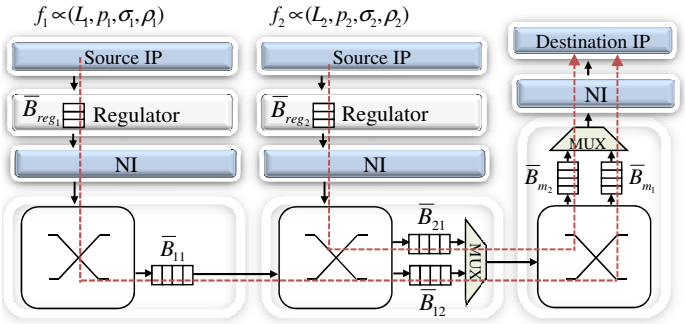

Fig. 2. An example of required buffers for two flows

B. The Analysis of Network Elements

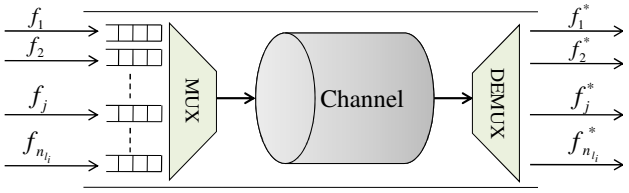

Fig. 3. Shared channel

Fig. 3 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 the channel access is arbitrated with a round robin policy. Assuming a fixed word length of L_w in all of flows, round robin arbitration means that each flow gets at least a c_{l_i}/n_{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. Round robin arbitration has good isolation properties because the minimum bandwidth for each flow does not depend on properties of the other flow. We can model a round robin arbiter of channel l_i as a latency-rate server [7] that its function is as $\beta_{R_{l_i}, T_{l_i}} = R_{l_i}(t - T_{l_i})^+$. R_{l_i} and T_{l_i} are defined as following:

$$R_{l_i} = \frac{c_{l_i}}{n_{l_i}} \quad (3)$$

$$T_{l_i} = \frac{(n_{l_i} - 1)L_w}{c_{l_i}} \quad (4)$$

Fig. 4 shows a traffic flow f_j after regulation which is called f_{R_j} and is passing through adjacent channels. Every channel $l_i \in L_{f_j}$ can be modeled as a latency-rate server with service curve $\beta_{R_{l_i}, T_{l_i}}$.

Assuming node k is destination of flow j, the ejection channel multiplexer of this node also can be modeled as a latency-rate server $\beta_{R_{m_k}, T_{m_k}}$. If processing capacity of the multiplexer is considered as c_{m_k} *flits/cycle*, it offers minimum service rate R_{m_k} *flits/cycle* and the maximum delay T_{m_k} *cycles* for each flow as following:

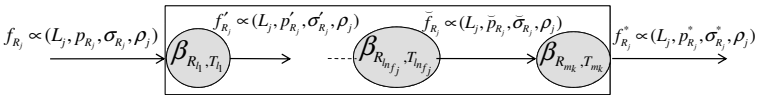

Fig. 4. Modeling each network element as a latency-rate server

$$R_{m_k} = \frac{c_{m_k}}{n_{d_k}} \tag{5}$$

$$T_{m_k} = \frac{(n_{d_k} - 1)L_w}{c_{m_k}} \tag{6}$$

where n_{d_k} is the number of flows with destination node k.

V. Buffer Size Optimization Problem

A. Tight Worst-Case Bounds for Each Flow

Let us assume that flow j passes through the regulator and several network elements offering each a latency-rate service curve. For determining the delay and backlog due to the regulation, the impact of it on the behavior of IPs should be considered. One is that IPs are *stalled* and therefore, there is no queuing buffer at the regulator. In the other case which is considered in this work, IPs are *not stalled* and the regulators use buffers to store transactions. This can reduce back-pressure at the expense of buffering cost. Let D_{reg_j} and B_{reg_j} be the delay and backlog for flow j, 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 $D_{reg_j} = \Delta\sigma_j/\rho_j$ [1].

For calculating tight worst-case bound on backlog along the network, the sum of the individual bounds on every element is computed. Thus, required buffer in network for flow j is bounded as following:

$$\bar{B}_j = \sum_{i \in L_{f_j}} \bar{B}_{ji} + \bar{B}_{m_j} \tag{7}$$

where $\bar{B}_{ji}$ is upper bound on the buffer of flow j for each $i \in L_{f_j}$ and $\bar{B}_{m_j}$ is maximum required buffer for the multiplexer of the destination node of flow j. $\bar{B}_{ji}$ and $\bar{B}_{m_j}$ can easily be obtained by Eq. (2). Finally, the buffer requirements for the flow j is bounded by $B_{reg_j} + \bar{B}_j$.

For obtaining tight worst-case delay bound along the network, we use the theorem of *Concatenation of network elements* [2]. Given are two nodes sequentially connected and each is offering a latency-rate service curve β_{R_i,T_i}, $i = 1\ and\ 2$, can be represented as a single latency-rate server as follows:

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2} \tag{8}$$

We can model all network elements on a given flow as a single latency-rate server $\beta_{R_{e_j},T_{e_j}}$ with following charactericts:

$$R_{e_j} = min(min_{l_i \in Lf_j}(\frac{c_{l_i}}{n_{l_i}}), \frac{c_{m_k}}{n_{d_k}}) \tag{9}$$

$$T_{e_j} = \sum_{l_i \in Lf_j} (\frac{(n_{l_i} - 1)L_w}{c_{l_i}}) + \frac{(n_{d_k} - 1)L_w}{c_{m_k}} \tag{10}$$

Based on a corollary of this theorem which is known *as Pay Bursts Only Once* [2], the equivalent latency-rate server is used for obtaining worst-case delay bound. Therefore, according to (1), (9) and (10), the maximum delay for the flow j in network is bounded by Eq. (11).

$$\bar{D}_j = \frac{L_j + \theta_{R_j}(p_{R_j} - R_{e_j})^+}{R_{e_j}} + T_{e_j} + n_{f_j} d_p \tag{11}$$

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 maximum delay for the flow j is bounded as: $D_{reg_j} + \bar{D}_j$.

B. Problem Definition

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 minimization problem can be formulated as:

$$\min_{p_{R_j}, \sigma_{R_j}} \sum_{\forall f_j \in F} B_{reg_j} + \bar{B}_j \tag{12}$$

subject to:

$$D_{reg_j} + \bar{D}_j \leq d_j; \qquad \forall f_j \in F \tag{13}$$

$$\rho_j \leq p_{R_j} \leq p_j; \qquad \forall f_j \in F \tag{14}$$

$$L_j \leq \sigma_{R_j} \leq \sigma_j; \qquad \forall f_j \in F \tag{15}$$

$$\bar{B}_j > 0; \qquad \forall f_j \in F \tag{16}$$

p_{R_j} and σ_{R_j} are optimization variables and d_j is the maximum delay that flow j can suffer in the network. 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. It is clear that by following the above mentioned equations, we can understand the effect of optimization variables on the objective function and all constrains of the defined problem.

In the literature, problem (12) is called a nonconvex Non-Linear Programming (NLP) problem [8]. There are different methods for solving this kind of optimization problems. In particular, we will use the Interior Point method [8] [9] to solve it.

VI. Experimental Results

A. Experimental Setup

To evaluate the capability of our method, we applied it to a real application provided by Ericsson Radio Systems which are mapped to a 4×4 2D mesh network. Although the experiments are performed on a mesh, our method is topology independent. In this work, the proposed analytical model is implemented in MATLAB and throughout the experiments, we consider an SoC with 500 MHZ frequency, 32 flits packets and 32 bits flits. We also assume that packets traverse the network on a shortest path using a deadlock free XY routing. As mapped onto a 4×4 mesh in Fig. 5, 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.

TABLE I
Comparison of the required buffer between different schemes

	Network Buffer	Regulator Buffer	Total Buffer
Without Reg.	421	0	421
Unoptimized Reg.	400	46	446
Optimized Reg.	196	21	217

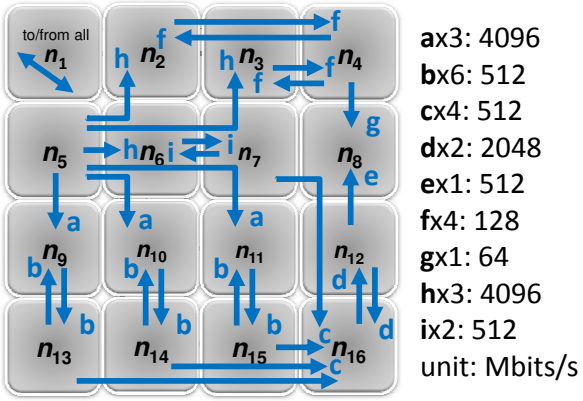

Fig. 5. Ericsson radio systems application

B. Buffer Size Optimization

Tables I and II, respectively, depict the maximum buffer requirements and delay for three schemes: In *without regulation*, there is no regulator; in *unoptimized regulation*, there is a regulator but it works on the worst-case with respect to buffer requirements; *optimized regulation* works based on the proposed minimizing buffer problem (12). From these tables, we can see that the *optimized regulation* scheme leads to a 48% reduction in total maximum required buffer and 16% in total maximum delay when compared with the *without regulation* scheme. Furthermore, these tables show that generally the regulator decreases the maximum buffer and delay in the network because of reducing the contention for shared resources. However, the *unoptimized regulation* scheme does not arrange these parameters appropriately; consequently, buffer area and packet latency in the regulator are increased to the extent that total buffer requirements and delay in this scheme become more than the *without regulation* scheme.

To go into more detail, we depict maximum required buffer and delay of each flow for these schemes in Fig. 6 and 7, respectively. Regarding Fig. 6, it is apparent that in the network with the proposed regulator, most flows require less buffer and also, as mentioned in Table I, total required buffer in this scheme is just a little more than half of it in the network wihout regulator. Also, Fig. 7 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, due to Table II, total and network delay are decreased in the *optimized regulation* scheme because of buffer-aware allocation in the network and contention reduction for shared resources.

The run-time of the proposed method in MATLAB is typically in the order of a few seconds. It is about 0.22 *sec* for the proposed problem of this application. Another interesting point is that the proposed regulator have no negative effect on the network throughput and it is the same in with and without regulation schemes.

TABLE II
COMPARISON OF THE MAXIMUM DELAY BETWEEN DIFFERENT SCHEMES

	Network Delay	Regulator Delay	Total Delay
Without Reg.	1302.9	0	1302.9
Unoptimized Reg.	1219.3	677.6323	1897
Optimized Reg.	907.6691	183.8812	1091.6

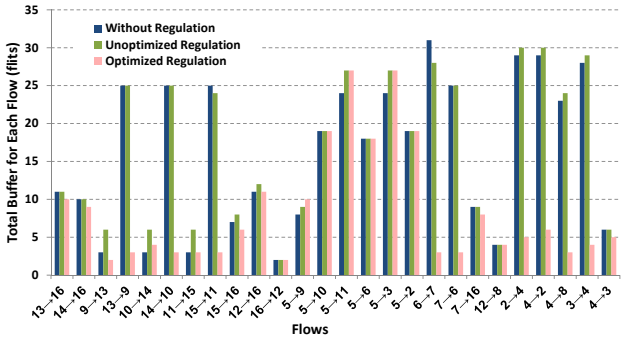

Fig. 6. Maximum buffer requirements for each flow

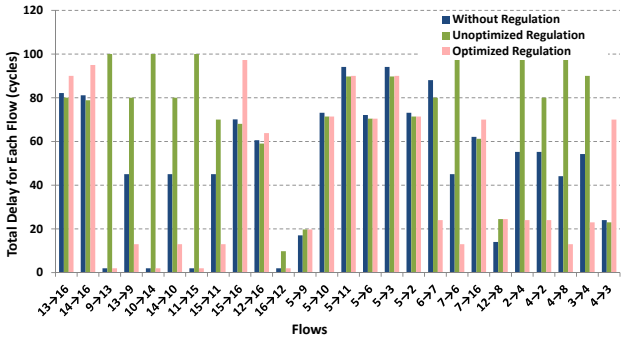

Fig. 7. Maximum delay for each flow

VII. Conclusion

In this paper, based on the concepts of regulation spectrum, we have presented an optimization problem for minimizing total buffers 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. Since reusing similar or identical switches facilitates the design process of NoC-based systems, as future work we intend to model both objectives as a multi-objective problem.

References

[1] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. van der Wolf and T. Henriksson, "Flow Regulation for On-Chip Communication", *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*, Nice, France, April 2009.

[2] J. Y. L. Boudec and P. Thiran, *"Network Calculus: A Theory of Deterministic Queuing Systems for the Internet"*, Number 2050 in LNCS, 2004.

[3] Z. Lu, D. Brachos, and A. Jantsch, "A flow regulator for on-chip communication", *In Proceedings of the System on Chip Conference (SOCC)*, Belfast, Northern Ireland, 2009.

[4] A. E. Kiasari, S. Hessabi, H. Sarbazi-Azad, "PERMAP: A performance-aware mapping for application-specific SoCs", *Proceedings of ASAP*, pp. 73-78, 2008.

[5] A. Jalabert, S. Murali, L. Benini, and G. De Micheli, "xPipesCompiler: A tool for instantiating application-specific NoCs," *Proceedings of Design, Automation and Test in Europe Conference (DATE)*, pp. 884-889, 2004.

[6] L. P. Tedesco, N. Calazans, and F. Moraes, "Buffer Sizing for Multimedia Flows in Packet-Switching NoCs", *Journal Integrated Circuits and Systems*, Vol. 3, No. 1, pp. 46-56, 2008.

[7] F. Gebali and H. Elmiligi, editors, "Networks on Chip: Theory and Practice", Taylor and Francis Group LLC - CRC Press, 2009.

[8] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[9] H.Y. Benson, R.J. Vanderbei, D.F. Shanno. "Interior-Point Methods for Nonconvex Nonlinear Programming: Filter Methods and Merit Functions" *Computational Optimization and Applications*, Vol. 23, No. 2, pp. 257-272(16), 2002.