Weighted Round Robin Configuration for Worst-Case Delay Optimization in Network-on-Chip

Fahimeh Jafari, Axel Jantsch, *Member, IEEE*, and Zhonghai Lu, *Member, IEEE*

***Abstract*— We propose an approach for computing the end-to-end delay bound of individual variable bit-rate flows in an First Input First Output multiplexer with aggregate scheduling under weighted round robin (WRR) policy. To this end, we use a network calculus to derive per-flow end-to-end equivalent service curves employed for computing least upper delay bounds (LUDBs) of the individual flows. Since the real-time applications are going to meet guaranteed services with lower delay bounds, we optimize the weights in WRR policy to minimize the LUDBs while satisfying the performance constraints. We formulate two constrained delay optimization problems, namely, minimize-delay and multiobjective optimization. Multiobjective optimization has both the total delay bounds and their variance as the minimization objectives. The proposed optimizations are solved using a genetic algorithm. A video object plane decoder case study exhibits a 15.4% reduction of the total worst case delays and a 40.3% reduction on the variance of delays when compared with round robin policy. The optimization algorithm has low run-time complexity, enabling quick exploration of the large design spaces. We conclude that an appropriate weight allocation can be a valuable instrument for the delay optimization in on-chip network designs.**

***Index Terms*— Network calculus, network-on-chip (NoC), performance evaluation, weight configuration, worst case delay optimization.**

I. Introduction

MANY multicore systems on chip (SoC) require different levels of service for different applications. Real-time applications have stringent performance requirements; the correctness relies not only on the communication result but also on the end-to-end delay bound. A data packet received too late could be useless. In other words, the least upper delay bound (LUDB) for each packet must not exceed its deadline. In such systems, it is desirable to minimize the end-to-end delay bound of the traffic streams satisfying their QoS requirements. Therefore, the first important consideration is to derive the LUDB for a given communication flow. To this end, based on the network calculus theory [1], we have presented and proved the required theorems in [2] and [3]. We then presented a methodology [4] to consider resource sharing scenarios, and also derive an end-to-end equivalent service curve (ESC) and LUDB by applying the proposed theorems. We assume that all traffic can be well characterized as the flows and scheduled as the aggregates, which means the multiple flows are scheduled as an aggregate flow. For a given flow, we study the maximum interference of all other flows based on the network calculus. Our proposed models [4] have been defined and validated under round robin (RR) policy. The RR policy uses the same service level for each connection while multiple service levels allow to better adapt to the application requirements by providing a different bandwidth and the latency guarantees. A weighted RR (WRR) scheduling policy assigns the weights to concurrent communications to define the multiple service levels. Higher service levels have greater weights and do not preempt the lower ones. It is important for designers to find the appropriate weights in WRR policy such that the corresponding service levels can support the QoS requirements for each communication connection. It is also desirable to optimize delay and throughput in the network.

In this paper, we extend our earlier proposed methodology for RR [4] to WRR policy. We then address an optimization problem of minimizing the total delay bounds subject to the performance constraints of the applications running on the SoC. To avoid unfair services, we consider another objective for minimizing the variance of the delay bounds in different flows. Minimizing the variance of the delay bounds avoids an intolerable delay of some flows, caused by processing and transmission of other flows. As both the mentioned objective functions are important for the real-time applications, we formulate them as a multiobjective problem under the QoS constraints. Finally, we show the benefits of the proposed method and quantify performance improvement.

Random variables appear in the formulation of the optimization problem, which causes random objective functions. Such optimization problems are usually solved by metaheuristic methods, which do not guarantee an optimal solution. However, they usually find high-quality solutions in a reasonable time [5]. There is a wide variety of metaheuristics like simulated annealing, tabu search, iterated local search, evolutionary computation, and genetic algorithms. We compare the performance of several metaheuristics [pure random search (PRS), Markov monotonous approach (MMA), adaptive search (AS), and genetic algorithm], and conclude that a genetic algorithm based method is the most suitable.

The rest of this paper is organized as follows. Section II discusses the related works. Section III introduces the basics of network calculus. Section IV is devoted to the underlying

Manuscript received September 24, 2015; revised February 16, 2016; accepted March 29, 2016.

F. Jafari is with the Department of Mathematics and Computer Science, Liverpool Hope University, Liverpool L16 9JD, U.K.(e-mail: jafarif@hope.ac.uk).

A. Jantsch is with Technische Universität Wien, Vienna 1040, Austria (e-mail: axel.jantsch@tuwien.ac.at).

Z. Lu is with the School of Information and Communication Technology, KTH Royal Institute of Technology, Kista 16440, Sweden (e-mail: zhonghai@kth.se).

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/TVLSI.2016.2556007

system model and the notations in our analysis. Section V introduces the major features of our formal method for analyzing the contention scenarios and computation of LUDB along with an example. The proposed optimization problems and corresponding solutions are represented in Sections VI and VII. Section VIII implements the algorithms for solving the proposed optimization problems. Experimental results are reported in Section IX. Finally, Section X concludes this paper.

II. Related Work

A. Performance Evaluation of Real-Time Services

There are different mathematical formalisms for performance evaluation in on-chip networks. We have surveyed four popular techniques along with their applications and also reviewed their strengths and weaknesses [6]. Most of the current works use queuing theory-based approaches. For example, Ben-Itzhak *et al.* [7] propose an analytical model for deriving the average end-to-end latency and link utilization of wormhole network-on-chips (NoCs) with heterogeneous link capacities and heterogeneous number of virtual channels (VCs) per unidirectional link. Queuing approaches often use probability distributions like Poisson to model traffic in the network while Poisson distribution is not appropriate for characterizing all the traffic patterns in the NoC applications. Qian *et al.* [8]–[10] suggest a methodology for addressing the limitation of assuming the traffic arrivals as Poisson processes. They present analysis for a general arrival process based on G/G/1 queue model [9]. In support vector regression-NoC [8], [10], the proposed analytical model is embedded into the learning process to form the feature vectors, and in turn, consider a more generalized traffic model. Although, the queuing approaches have been widely used for performance analysis, they cannot properly model some significant features, such as nonstationary, self-similarity, and higher order statistics, for NoC-based multicore platform designs. To overcome this limitation, Bogdan *et al.* [11] suggest an approach that analyzes the traffic dynamics and captures the nonstationary effects of the NoC workload. They propose the QuaLe model [12] based on statistical physics to analyze the information flow and buffer usage in the NoCs, and also investigate the impact of packet injection rate and data packet sizes on the multifractal spectrum of traffic. In the follow-up to this paper, Bogdan [13] proposes mathematical frameworks for exascale computing in data-centers-on-a-chip architectures that exploit the NoC paradigm for interconnecting a large number of heterogeneous processing elements. The frameworks can account and exploit the nonstationary and multifractal characteristics of computation and communication workloads. To address the problem of distant and large volume data exchange, Xue and Bogdan [14] present a user-cooperation network coding strategy for the NoCs.

Network calculus is another mathematical approach specialized for deriving worst case performance analysis. Network calculus is able to model all the traffic patterns with bounds defined by the arrival curves. It provides the facility of capturing dynamic features of the network based on the traffic flows shapes. Network calculus has been applied in a number of papers for the performance analysis of real-time services in networks with aggregate scheduling. For example, Charny and Le Boudec [15] derive a closed-form delay bound in a generic network assuming a fluid model. An extended model is proposed [16] to look into packetization effects. The main limitation of these models is that they work well only for small utilization factors in a generic network configuration. Lenzini *et al.* [17] describe a methodology for obtaining per-flow worst case delay bound in the tandem networks of rate-latency nodes traversed by the leaky-bucket shaped flows. This method yields better bounds than those previously proposed. Qian *et al.* [19] present analytical models for traffic flows under strict priority queuing and WRR scheduling in on-chip networks. They then derive per-flow end-to-end delay bounds using these models.

All the previous works based on network calculus investigate computing the delay bounds only for average behavior of flows and they do not consider peak behavior, which results in less accurate bounds. Since a considerable number of real-time applications are transmitted by variable bit-rate (VBR) traffics, we have proposed a methodology [4] to consider performance analysis for VBR traffic characterized by (L, p, σ, ρ) in on-chip networks employing aggregate resource management. This method achieves more accurate delay bounds. In this paper, we extend this method to WRR, and then regulate weights in each router to minimize the delay bounds while satisfying the performance constraints.

B. Performance Improvement

There exist some previous related works focused on flow control, adaptive routing, and also domain isolation based on noninterfering router microarchitecture. Concer *et al.* [20] propose an end-to-end flow control protocol, called connection-then-credit (CTC), to handle message-dependent deadlocks in the on-chip networks. Although, this protocol adds the latency overhead to the transfer of the message, under some conditions, the performance of the system can be improved. As CTC is an extension to the normal credit-based flow control protocol, Sallam *et al.* [21] implement both the protocols and investigate implementation tradeoffs along with performance analysis. Joshi and Mutyam [22] consider a prevention flow control mechanism that satisfies the cost/performance constraints in torus networks while preventing deadlocks by combining priority arbitration with prevention slot cycling. However, the mechanism can lead to the deadlocks with variable-sized packets.

There are a wide range of adaptive algorithms varying from partially to fully adaptive and have the potential for improving system performance [23]–[26]. For instance, Lin *et al.* [23] develop a bufferless routing algorithm and have shown improvements of up to 10% for average latency when compared to older designs. Najib *et al.* [24] present a look-ahead partial adaptive routing for their prior proposed low-latency NoC router [27]. They show that their algorithm improves the performance of a baseline design of the router under imbalanced traffic. Ma *et al.* [25], [26] propose a flow control technique, called whole packet forwarding, which is a VC reallocation scheme for fully adaptive routing in wormhole on-chip networks. They show that the fully adaptive routing

can offer higher performance than several partial adaptive routing algorithms.

Wassel *et al.* [28] propose SurfNoC that is a low-latency time-division-multiplexed packet-switched k-ary n-cube network. The channels are scheduled in each dimension of a mesh network in a pipelined fashion using the dimension-ordered routing algorithm. Psarras *et al.* [29] present a noninterfering VC-based architecture for on-chip networks, called PhaseNoC, which adopts time-division multiplexing at the VC level. Applications are mapped to disjoint sets of the VCs to isolate them both inside the routers pipeline and at the network level. They also design appropriate scheduling of flows to reduce area/delay cost in the network.

All the above-mentioned works consider reducing the average latency in different system models while, in this paper, we calculate worst case delay bounds under WRR scheduling policy and deterministic routing and minimize worst case delay and variance.

III. Network Calculus Background

Network calculus is a mathematical approach to compute the worst case bounds for guaranteed services in communication networks [1]. It uses min-plus algebra to convert nonlinear queuing systems into linear systems. The algebra structure of min-plus is $(\Re \cup \{+\infty\}, \wedge, +)$ in which $\wedge$ represents the minimum operation $f \wedge g = \min(f, g)$. The min-plus convolution, denoted by $\otimes$, is defined as $(f \otimes g)(t) = \inf_{0 \le s \le t}\{f(t-s) + g(s)\}$ where the two functions f and g are the wide-sense increasing functions.

An arrival curve defines an upper bound on the cumulative arrival process to characterize a traffic flow and a service curve defines a lower bound on the cumulative service process. In this paper, we use Traffic SPECification (TSPEC) [35] for characterizing traffic to look into both the average and peak behaviors of a flow. With TSPEC, the arrival curve of flow f_j is defined as $\alpha_j(t) = \min(L_j + p_j t, \sigma_j + \rho_j t)$, where L_j is the maximum transfer size, p_j is the peak rate ($p_j \ge \rho_j$), σ_j is the burstiness ($\sigma_j \ge L_j$), and ρ_j is the average (sustainable) rate. We denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. A well-formulated service model to reflect the service capability of a node is the rate-latency function defined as $\beta_{R,T} = R(t-T)^+$, where R is the minimum service rate and T is the maximum processing latency of the node. We use x^+ to denote the function $x^+ = x$ if $x > 0$; $x^+ = 0$, otherwise. $\vee$ represents the maximum operation $f \vee g = \max(f, g)$. Burst delay function $\delta_T(t) = +\infty$, if $t > T$; $\delta_T(t) = 0$, otherwise. Affine function $\gamma_{b,r}(t) = b + rt$, if $t > 0$; $\gamma_{b,r}(t) = 0$, otherwise. Therefore, $\delta_T \otimes \gamma_{b,r}(t) = b + r(t-T)$. $\oslash$ represents the min-plus deconvolution as $(f \oslash g)(t) = \sup_{s \ge 0}\{f(t+s) - g(s)\}$.

IV. System Model

We consider a NoC architecture in which every node contains a core equipped with a network interface and a router with the input and output channels. Assumptions are given as follows.

1) The NoC architecture can have arbitrary topologies.
2) A flow consists of packets and each packet is broken into flits. We consider the arbitration granularity of one word with a fixed word length of L_w for all the flows. L_w is assumed to be 1 flit.
3) The packets have fixed length and traverse the network in a best-effort fashion with virtual-cut-through switching technique using deadlock-free deterministic routing.
4) Routers have multiple flit input buffers, but no output buffers.
5) The router can have multiple VCs per in-port. The VC allocation is deterministic and each VC receives an aggregate service.
6) The buffers are bounded by (7) and the network is lossless.
7) All traffic is modeled as the TSPEC flows $f = \text{TSPEC}(L, p, \sigma, \rho)$ at the entry into the network.
8) To characterize the flows based on TSPEC, we assume unbuffered leaky bucket controllers (regulators), which do not buffer the packets, but stall the traffic producers or IPs [18].
9) We assume WRR arbitration and model it by a rate-latency service curve as $\beta = \delta_T \otimes \gamma_{0,R}$, it is assumed that $\rho \le R$ and $p \ge R$, where ρ is the average rate, p is the peak rate, and R is the minimum service rate.
10) Flows are classified into a prespecified number of aggregates and the traffic of each aggregate is buffered and transmitted in a First Input First Output (FIFO) order, denoted as FIFO multiplexing.
11) Different aggregates are buffered separately and each aggregate is guaranteed a rate-latency service curve.
12) The hardware limits the peak rate to 1 flit/cycle.

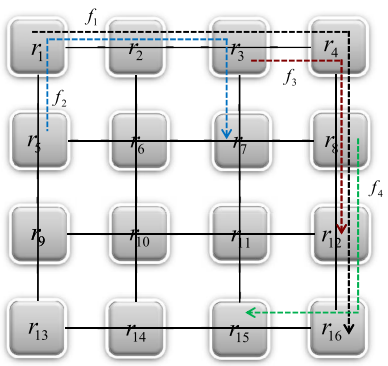


Fig. 1. Example of an NoC with 16 nodes and 4 flows.

Fig. 1 depicts an example with 16 nodes and 4 flows. The multiple flows that share the same buffer and channel in the same router, for example f_1 and f_2 in router 2, are scheduled as an aggregate flow denoted as $f_{\{1,2\}}$. The tagged flow is a flow for which the delay bound is derived and the other flows that compete with the tagged flow for the same resource are called contention flows. In the example, if f_1 is the tagged flow, f_2, f_3, and f_4 would be the contention flows. Table I presents notations in this paper.

Subindices (f_i, r_j) indicate that they are related to the flow f_i in router r_j. For instance, $\alpha_{(f_1,r_2)}$ indicates the arrival curve of f_1 in router r_2. Using f_{s_i} instead of f_i in the subindex means that the notation is related to the f_{s_i}, which can be one flow or an aggregate flow. For example, $\beta_{(\{1,2\},r_2)}$ refers to the aggregate flow $f_{\{1,2\}}$ in router r_2.

TABLE I
LIST OF NOTATIONS

f_i	Flow i
$F^{RPV}_{(j,i,k)}$	The set of flows passing through VC k in physical channel i of router j
$F_{(j,l,s,k)}$	The set of flows passing from VC s of input channel l to output channel k in router j
Input PC#	The number assigned to an input physical channel
Output PC#	The number assigned to an output physical channel
VC#	The number assigned to an input virtual channel
$InPC$	The set of input physical channels in each router
$OutPC$	The set of output physical channels in each router
$InVC$	The set of input virtual channels in each input physical channel
L_i	The maximum transfer size of f_i *(flits)*
p_i	The peak rate of f_i *(flits/cycle)*
σ_i	The burstiness of f_i *(flits)*
ρ_i	The average rate of f_i *(flits/cycle)*
$Src(i)$	The source node of f_i
r_j	Router j
β_j	The service curve of r_j
R	The minimum service rate in a rate-latency service curve
T^l	The maximum processing latency of the arbiter in the router *(cycles)*
T^{HoL}	The maximum waiting time in the FIFO queue of the router *(cycles)*
T^{Total}	The total processing delay which comes from contention flows the router and equals to the sum of T^l and T^{HoL}
D_{router}	Time spent for packet routing decision *(cycles)*
L_w	The word length in the flow *(flits)*
C	The channel capacity *(flits/cycle)*
CF_t	The set of contention flows of tagged flow f_t
s_i	The set of joint flows in an aggregate flow (when the number of elements of s_i is equal to 1, there is only a single flow)
f_{s_i}	An aggregate flow of s_i
$\|s_i\|$	The cardinality of set s_i, which is a measure of the "number of elements of the set"
$F^B_{(s_i,r_j)}$	The set of flows which share the same buffer in router r_j with flow f_{s_i}
$w_{(j,l,s,k)}$	The weight assigned to node r_j, input Physical Channel (PC) l, input VC s, and output PC k
L_{WR}	The length of a round in WRR policy

V. LUDB DERIVATION FOR WRR POLICY

To derive a delay bound per-flow passing a series of nodes, one simple way is to sum up the delay bounds at each node, which results in a loose delay bound. A theorem called Pay Bursts Only Once is known to give a tighter upper estimate on the delay bounds, when an end-to-end service curve is obtained prior to the delay computations. This accounts for bursts of the tagged flow only once instead of at each link independently. This principle also holds in aggregate scheduling networks. To this end, we propose the two following steps to derive the end-to-end service curve for a tagged flow.

1) *Step 1 (Intrarouter ESC):* This step derives intrarouter ESCs for each router through which the tagged flow is passing. Different resource sharing scenarios in each router are distinguished and intrarouter analysis models are built.
2) *Step 2 (Interrouter ESC):* In this step, according to the intrarouter analysis models, we present a set-theoretic approach that recognizes and investigates different contention scenarios that a flow may experience along its routing path and in turn derive an end-to-end ESC for the tagged flow.

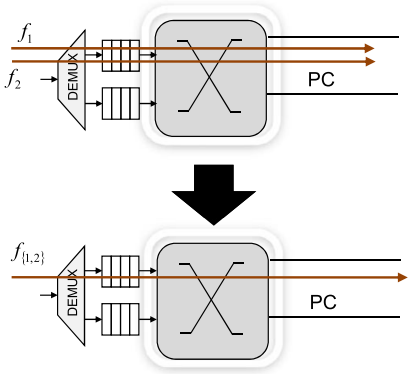


Fig. 2. Example of channel and buffer sharing.

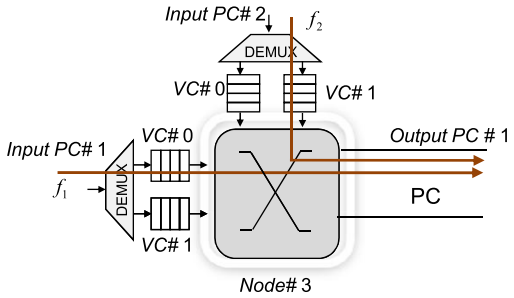


Fig. 3. Example of channel sharing.

For extending our proposed analytical method to WRR policy, we expand the first step while the second step is unchanged. Similarly, to support some other arbitration policies, only the first step must be modified.

A. Intrarouter ESC

In this step, we consider three types of resource sharing, including channel and buffer sharing, channel sharing, and buffer sharing.

1) Channel and Buffer Sharing: As shown in Fig. 2, the multiple flows share both the same buffer and channel in the router, and are scheduled as a flow called aggregate flow. An aggregate flow including the tagged flow is named as tagged aggregate flow. In this case, intra-ESC is derived for the tagged aggregate flow instead of the tagged flow. In Section V-B, due to the contention scenarios, we will remove the contention flows from the ESC of a tagged aggregate flow in order to extract the ESC of the tagged flow.

2) Channel Sharing: Fig. 3 depicts an example of a channel shared between the two flows f_1 and f_2. The WRR arbiter associates a weight $w_{(j,l,s,k)}$ in cycles on each aggregate/single flow f_{s_i} passing from the input VC s of input physical channel (PC) l in router r_j to the output PC k. The value of the weight assigned to a channel depends on the flows passing through that channel. Then, the router will try to give the flow a period of $w_{(j,l,s,k)}$ cycles before moving to the next node. In each round, for a nonempty VC buffer encountered, the router serves up to corresponding configured weight in cycles. The maximum length of a round consequently equals to $\sum_{l,s} w_{(j,l,s,k)}$ cycles, denoted as L_{WR}. The least service offered to one flow in a VC is completely dependent on the weight of that VC and the sum of all the other weights. With the WRR scheduling, the worst case appears for a flow when

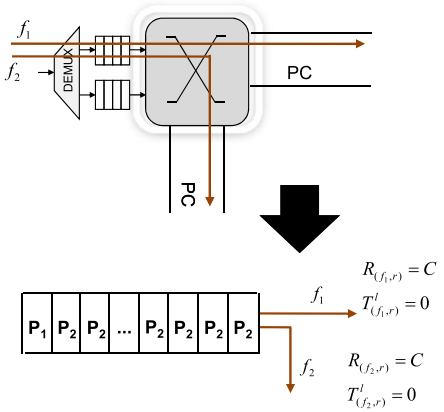


Fig. 4. Example of buffer sharing.

it just misses its slot in the current round. Consequently, it will have to wait for its slot assigned at the next round. In the worst case, each flow f_{s_i} passing from the input VC s of the input PC l in router r_j to the output PC k will have to wait up to $(\sum_{p,q} w_{(j,p,q,k)} - w_{(j,l,s,k)}) \times ((L_w/C) + D_{\text{router}})$ cycles before to be served, and get at least a $(w_{(j,l,s,k)}/\sum_{p,q} w_{(j,p,q,k)}) \times C$ of the channel bandwidth, where C is the channel capacity, L_w is the word length, and D_{router} is the delay for packet routing decision in a router. A flow may get more service rate if the other flows use less, but we now know a worst case lower bound on the bandwidth. Based on the network calculus theory, we can use the abstraction of service curve to model a WRR arbiter in router r_j for the flow f_{s_i} as a rate-latency server $\beta_{(s_i,r_j)} = R_{(s_i,r_j)}(t - T^l_{(s_i,r_j)})^+$, where $R_{(s_i,r_j)}$ is the minimum service rate and $T^l_{(s_i,r_j)}$ is the maximum processing latency of the arbiter in router r_j for flow the f_{s_i}. $R_{(s_i,r_j)}$ and $T^l_{(s_i,r_j)}$ are defined as follows:

$$R_{(s_i,r_j)} = \frac{w_{(j,l,s,k)}}{\sum_{p,q} w_{(j,p,q,k)}} \times C \tag{1}$$

$$T^l_{(s_i,r_j)} = \left(\sum_{p,q} w_{(j,p,q,k)} - w_{(j,l,s,k)}\right) \times \left(\frac{L_w}{C} + D_{\text{router}}\right). \tag{2}$$

In the example of Fig. 3

$$R_{(f_1,r_3)} = \frac{w_{(3,1,0,1)}}{w_{(3,1,0,1)} + w_{(3,2,1,1)}} \times C$$

$$T^l_{(f_1,r_3)} = w_{(3,2,1,1)} \times \left(\frac{L_w}{C} + D_{\text{router}}\right).$$

As aforementioned, the sum of weights of the flows sharing a channel is equal to the round time. Thus, as the value of round time (the sum of weights) is increased or decreased, individual flows proportionally get more or less time slots, respectively, which means the weights proportionally increased or decreased. Therefore, the model is able to adjust the weights based on the round time.

3) Buffer Sharing: Fig. 4 depicts a buffer shared between the two flows f_1 and f_2. In this type of sharing, we introduce two kinds of delay for a tagged flow.

1) Head-of-line delay (HoL) is the maximum waiting time of the packet in the FIFO queue, which is denoted by T^{HoL}.
2) Processing delay is the maximum processing latency of the router's arbiter for the flow, which is denoted by T^l.

Therefore, the total delay for a tagged flow f_i in router r_j is calculated as $T^{\text{Total}}_{(f_i,r_j)} = T^{\text{HoL}}_{(f_i,r_j)} + T^l_{(f_i,r_j)}$.

$T^l_{(f_i,r_j)}$ and $R_{(f_i,r_j)}$ can be calculated according to (2) and (1), respectively. To show how $T^{\text{HoL}}_{(f_i,r_j)}$ is calculated, we consider the example in Fig. 4 and assume that f_1 is the tagged flow. As depicted in Fig. 4, $T^{\text{HoL}}_{(f_1,r)}$ is equal to the maximum delay for passing packets of flow f_2 in the buffer. According to [1], the maximum delay for the flow f_j is bounded by

$$\bar{D}_{(f_j,r)} = T^l_{(f_j,r)} + \frac{L_j + \theta_j (p_j - R_{(f_j,r)})^+}{R(f_j,r)} \tag{3}$$

where $\theta = (\sigma - L)/(p - \rho)$.

Therefore, $T^{\text{HoL}}_{(f_1,r)}$ is given as follows:

$$T^{\text{HoL}}_{(f_1,r)} = T^l_{(f_2,r)} - \theta_2 + \frac{L_2 + \theta_2 p_2}{R_{(f_2,r)}}. \tag{4}$$

In the case of more than one flow sharing the same buffer with the tagged flow, the HoL delay for a tagged flow f_{s_i} in router r_j is calculated as

$$T^{\text{HoL}}_{(s_i,r_j)} = \sum_{\forall f_c \in F^B_{(s_i,r_j)}} T^{\text{HoL}(f_c)}_{(s_i,r_j)} \tag{5}$$

where $F^B_{(s_i,r_j)}$ is the set of flows that share the same buffer in router r_j with a tagged flow f_{s_i}. Also $T^{\text{HoL}(f_c)}_{(s_i,r_j)}$ is given by

$$T^{\text{HoL}(f_c)}_{(s_i,r_j)} = T^l_{(f_c,r)} - \theta_c + \frac{L_c + \theta_c p_c}{R_{(f_c,r)}}. \tag{6}$$

Therefore, the router r_j can give flow f_{s_i} service bounded by a curve $\beta_{(s_i,r_j)} = \delta_{T^{\text{Total}}_{(s_i,r_j)}} \otimes \gamma_{0,R_{(s_i,r_j)}}$, where $T^{\text{Total}}_{(s_i,r_j)}$ is equal to $T^{\text{HoL}}_{(s_i,r_j)} + T^l_{(s_i,r_j)}$ and $R_{(s_i,r_j)}$ is calculated by (1).

We analyze the buffer space threshold for each VC based on the TSPECs of the flows passing through that VC, and also interference between them. The buffer space threshold for VC k in PC i of router j is given as

$$B_{(j,i,k)} = \sum_{\forall f_c \in F^{\text{RPV}}_{(j,i,k)}} \left(\sigma_c + \rho_c T^p_{(f_c,r_j)} + \left(\theta - T^p_{(f_c,r_j)}\right)^+ \times \left[(p_c - R_{(f_c,r_j)})^+ - p_c + \rho_c\right]\right) \tag{7}$$

where $F^{\text{RPV}}_{(j,i,k)}$ is the set of flows passing through VC k in PC i of router j.

B. Interrouter ESC

In this step, we aim to extract the ESC of the tagged flow by removing the contention flows from the ESC of the tagged aggregate flows. We have described this stage in elaborate detail through our previous paper [4]. Algorithm 1 presents the main steps of deriving the end-to-end ESC for a given tagged flow. The only difference between this algorithm and the one presented for the RR [4] results from the different methods proposed for calculating intrarouter ESCs (Line 9). The algorithm with all stages, including the details of inter-router ESC step, is presented in [4].

Algorithm 1 End-to-End ESC Algorithm

```
Find the set of contention flows of tagged flow f_t, denoted by CF_t
for ∀j ∈ CF_t do
    if Src(j) ∉ Path(t) then
        Find joiningnode = JoiningPoint(f_j)
        Calculate X = ESC(f_j, Src(j), joiningnode)
        α_j = α_j ⊘ X
    end if
end for
Calculate intra-router ESC for WRR based on Section V-A.
Calculate inter-router ESC (See [4])
return end-to-end ESC for tagged flow f_t
```

Now, we can obtain LUDB from the end-to-end ESC according to the proposed theorem for calculating the delay bounds [3]. We have automated our proposed analytical approach as a tool for the worst case performance analysis. The WRR gives flow isolation. Each flow is served at its own weight in the worst case. It is notable that the proposed approach is independent of the routing algorithm, but the routing algorithm must be predefined (deterministic).

In our proposed model, σ and ρ represent the congestion level. The effect of these parameters on the delay bounds can be analyzed by the following theorems and formula used in the proposed approach.

VI. Optimization Problem Formulation

Latency is one of the most critical challenges for on-chip interconnection network architectures [30]. However, there exists a huge search space to explore for minimizing latency. Thus, to design a low-latency on-chip network, designers need to investigate the optimization problems and make appropriate decisions. The general problem is defined as follows:

General problem definition

Given architecture specifications, application parameters, and traffic characteristics (e.g., TSPEC in this paper);

Find a set of decision variables;

Such that the network delay is minimized and performance constraints are satisfied.

The decision variables capture application mapping to the processing cores, the traffic regulation parameters (e.g., peak rate, burstiness, and packet injection rates to the network), a switch architecture, a resource allocation strategy (e.g., bandwidth of channels and so on), the weight configuration in WRR policy, and a routing algorithm.

In networks with the WRR policy, the weight configuration for the flows can be in conflict because of the contention for shared resources. This makes the weight parameters nontrivial, and thus, given single or multiple objectives, a parameter selection becomes necessary. In this paper, we find a weight configuration in WRR policy to minimize the total worst case delay in the network. Weight allocation is actually a resource allocation strategy in which a flow with larger weight gets more bandwidth or a higher service level. The weight of each nonempty VC is selected based on the TSPECs of the flows passing through that VC, and to minimize the interference between them. In Section IX, we describe how the weights affect the delays of flows. For example, results for a real-time application show that an optimized weight allocation leads to ∼48.8% reduction in a total worst case delay compared to a random configuration. Optimized WRR weight assignment leads to a 81.1% decrease of delay over a poor weight configuration and a 15.4% decrease over an RR-based allocation.

On the other hand, faster transmission is not necessarily better in a shared communication channel, since faster delivery requires higher link bandwidth reservation and may incur a larger delay for another contention flow in a shared channel, leading to an intolerable delay. To avoid throttling some communications, we investigate another objective function that is minimizing the variance of the delay bounds in different flows. As both the mentioned goals are worthwhile for the real-time applications, we formulate them as a multiobjective problem in Section VI-B.

A. Delay Optimization

As stated before, our objective is to choose appropriate weights in a WRR policy, assigned to the channels on the path of flows, so as to minimize the sum of LUDBs while satisfying acceptable performance in the network. Note that $w_{(j,l,s,k)} = 0$ when no flow is passing from VC s of input channel l to output channel k in router j. Thus, the delay bound minimization problem, minimize-delay, can be formulated as follows.

Given a set of flows $F = \{f_i \propto (L_i, p_i, \sigma_i, \rho_i)\}$, routing matrix R, and the number of weight cycles L_{WR}, find the weights in WRR policy as $w_{(j,l,s,k)}$ for $\forall i \in N$, $\forall j \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$, such that

$$\min_{w_{(j,l,s,k)}} \sum_{\forall f_i \in F} D_i \tag{8}$$

subject to

$$\sum_{l,s} w_{(j,l,s,k)} = L_{\mathrm{WR}} \quad \forall j \in N \quad \forall k \in OutPC \tag{9}$$

$$\frac{L_{\mathrm{WR}} \times \sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)} \leq L_{\mathrm{WR}}$$
$$\forall j \in N \quad \forall l \in InPC \quad \forall s \in InVC \quad \forall k \in OutPC \tag{10}$$

where $w_{(j,l,s,k)}$ for $\forall j \in N$, $\forall l \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$ are the optimization variables.

Equation (8) is the objective function of this optimization problem, which minimizes the total LUDBs. Constraint (9) says that the sum of weights assigned to the flows that pass through the same output channel k in the router j, the same WRR scheduler, is equal to L_{WR}. Although, we have assumed the same value of L_{WR} for all the arbiters, the optimization problem can be easily adapted to different values of the sum of weights. To reach acceptable performance in the network, the share of $w_{(j,l,s,k)}$ from L_{WR} should be proportionate to $(\sum_{m \in F_{(j,l,s,k)}} \rho_m / C)$, where $F_{(j,l,s,k)}$ is the set of flows that pass through VR s of the input channel l to the output channel k in router j. Therefore, we can consider $(\sum_{m \in F_{(j,l,s,k)}} \rho_m / C)$ as a criterion of minimum guaranteed performance for the flows in $F_{(j,l,s,k)}$. In this respect, we

have $(\sum_{m\in F_{(j,l,s,k)}} \rho_m/C) \leq (w_{(j,l,s,k)}/L_{\mathrm{WR}})$, which means $(L_{\mathrm{WR}} \times \sum_{m\in F_{(j,l,s,k)}} \rho_m/C) \leq w_{(j,l,s,k)}$ as stated in constraint (10). It is also clear that the value of each weight should be less than the number of weight cycles, which means $w_{(j,l,s,k)} \leq L_{\mathrm{WR}}$.

By following the equations described in Section V, the effect of optimization variables on the objective function of the defined problem is obvious.

In the literature, problem (8) is called a stochastic and nonlinear optimization problem [31]. We solve it using genetic algorithms because of their well-known robustness and ability to solve the large and complex discrete optimization problems.

B. Multiobjective Optimization Problem

In order to avoid an intolerable delay of some flows due to processing and transmission of other flows, we would like to find the appropriate weights in WRR policy, so that the variance of the delay bounds in the network is minimized. Using a general variance formula, we can calculate the variance of the delay bounds as $(1/|F|) \times \sum_{\forall f_i \in F} (E(D) - D_i)^2$. Hence, another optimization problem can be formulated to minimize both the total delay bounds and their variance while satisfying constraints (9) and (10), as follows.

Given a set of flows $F = \{f_i \propto (L_i, p_i, \sigma_i, \rho_i)\}$, routing matrix R, and the number of weight cycles L_{WR}, find the weights in WRR policy as $w_{(j,l,s,k)}$ for $\forall j \in N$, $\forall l \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$, such that

$$\min_{w_{(j,l,s,k)}} \sum_{\forall f_i \in F} D_i \tag{11}$$

$$\min_{w_{(j,l,s,k)}} \frac{1}{|F|} \times \sum_{\forall f_i \in F} (E(D) - D_i)^2 \tag{12}$$

subject to

$$\sum_{l,s} w_{(j,l,s,k)} = L_{\mathrm{WR}} \quad \forall j \in N \quad \forall k \in OutPC \tag{13}$$

$$\frac{L_{\mathrm{WR}} \times \sum_{m\in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)} \leq L_{\mathrm{WR}}$$
$$\forall j \in N \quad \forall l \in InPC \quad \forall s \in InVC \quad \forall k \in OutPC. \tag{14}$$

Although, the solution of multiobjective optimization problems consists of a set of solutions, the user needs only one solution. The decision about which solution is the best depends on the decision maker, and there is no universally accepted definition of optimum as in single-objective optimizations [36]. A multiobjective problem is often solved by composing the objective function as the weighted sum of the objectives, which is in general known as the weighted-sum or scalarization method. In this approach, a relative preference factor of the objectives should be known in advance. In more detail, the weighted-sum method minimizes a positively weighted sum of the objectives, that is

$$\min(\gamma_1 f_1 + \gamma_2 f_2) \tag{15}$$

where γ_1 and γ_2 are the weighting coefficients representing the relative importance of the objectives.

The simplicity and efficiency of this method makes it an appropriate option for solving the multiobjective optimizations with complex and nonsmooth objective functions. Therefore, we convert our proposed multiobjective problem into a scalar optimization problem with equal weighting coefficients. Since the problem is still a nonsmooth and stochastic optimization, we use the genetic algorithm to solve it.

Algorithm 2 General Scheme of GA in Pseudo-Code

```
P1 ← Generate random population of n chromosomes
Evaluate the fitness f(x) for each x ∈ P1
repeat                                   ▷ Create a new population
    Selection: Select two parents from a population.
    Crossover: With a crossover probability cross over the
      parents to form a new offspring (children).
    Mutation: With a mutation probability mutate new
      offspring at each locus (position in chromosome).
    Accepting: Place new offspring in a new population
until the new population is not complete
Use new generated population for a further run.
if the end condition is satisfied then
    return The best solution in current population
else
    Go to step 2
end if
```

VII. Solution Method

The proposed optimization problems have complex and highly nonlinear objective functions. Moreover, due to (1), the minimization functions of the decision variables appear in the formulation of per-flow LUDBs and consequently in the objective formulation, which cause the random objective functions.

Such optimization problems are usually solved by the metaheuristic methods, which make few assumptions about the problem being solved and do not guarantee an optimal solution. However, they can usually find a good solution [5].

Among different types of metaheuristics, we choose genetic algorithms to solve the proposed optimization problems, because they are the most appropriate for the large and complex nonlinear models specially where the objective function is discontinuous, stochastic, very rugged and complex, noisy, or has many local optima [37], [38]. Moreover, they have been proven to be effective in avoiding the local optima and discovering the global optimum in even a problem with very complex objective functions [37]. Genetic Algorithms (GAs) tend to be computationally expensive for the solutions of optimization problems with nonlinear equality and inequality constraints [38], which do not occur in our proposed problems. Although, a GA does not always find a global optimum to a problem, it almost always finds high-quality solutions [37].

GA generates solutions to the optimization problems mimicking the process of natural evolution such as inheritance, mutation, selection, and crossover. Algorithm 2 presents a general scheme of GA in pseudo-code. The algorithm is started with an initial population of solutions represented by chromosomes. A chromosome contains the solution as a set of parameters in a form of genes. A gene is a position or set of positions in a chromosome, represented as a simple string or other data structures. The algorithm selects the solutions, called parents, from the population and produces a new

solution, called offspring, to form a new population. Although, parents can be selected in many different ways, the main idea is that better parents according to their fitness hopefully will produce better offspring. Crossover and mutation are the two basic operators of GA, which produce a new offspring. This process is repeated until some condition, such as the number of populations or improvement of the best solution, is satisfied.

A method for encoding potential solutions of the problem is needed. There are different approaches to encode the solutions like binary encoding, value encoding, permutation encoding, and tree encoding.

VIII. Implementation

In Section VII, we have discussed why the GA is selected for solving the optimization problems and presented a general description of the GA. This section shows how we have mapped our proposed optimization problems into the problems that can be solved by the GA. We present Algorithm 3 to detail the procedure of deriving the optimal weights for the proposed optimization problems. Objective and constraint functions in the GA are the same as what we have defined for the proposed optimization problems. The objective functions are implemented as fitness function, which is called whenever the population is created or a selection is made from the population. The weights, which are the decision variables, are considered as a vector and uniquely mapped onto a chromosome. As the proposed optimizations have only boundary constraints, these constraints in GA can be reflected as intervals of chromosomes domain. Parent, which is a chromosome, presents the current solution for this round and offspring is a new vector generated from the parent, which may be the next solution.

The algorithm uses a binary representation of the chromosomes as fixed-length strings over the alphabet $\{0, 1\}$, such that they are well suited to handle the optimization problems. It uses a function $Encoding()$ to map solutions $\vec{w} \in W$ to a binary string $\{0, 1\}^l$ and defines a function $Decoding()$ to do the reverse. To this end, real-valued vector $\vec{w} \in \Re^n$ is presented by a chromosome in the form of a binary string $\vec{x} \in \{0, 1\}^l$. The chromosome is logically divided into n segments (gene) of equal length S_{gene} as $(w_1 \dots w_n)$, where S_{gene} is gene size and $l = n \times S_{\text{gene}}$. Each gene w_i is decoded to yield the corresponding integer value, and the integer value is in turn linearly mapped to its interval of real values, denoted as $[Lb_i, Ub_i] \subset \Re$, where Lb_i and Ub_i indicate lower and upper bound constraints on w_i, respectively. In this paper, we use a gray code interpretation of the binary string. The main advantage of the gray codes is that they are different by only one bit.

Fig. 5 shows an example of the decoding process for the string segments of length $S_{\text{gene}} = 8$, which allows the representations of integers $\{0, 1, \dots, 255\}$. As shown in Fig. 5, the function $Decoding()$ first converts a given gray code to an integer value $p_i \in \{0, \dots, 2^{S_{\text{gene}}} - 1\}$ and then maps p_i linearly to its corresponding interval $[Lb_i, Ub_i]$ as $Lb_i + (Ub_i - Lb_i/2^{S_{\text{gene}}} - 1) \times p_i$.

Algorithm 3 Genetic Algorithm-Based Weight Optimization

1: $Pop1 \leftarrow Initilization\_FirstPopulation()$
2: $Encoded\_Pop1 \leftarrow Encoding(Pop1)$
3: $Temp\_Pop \leftarrow Encoded\_Pop1$
4: **for** i=1 to Iteration# **do**
5: $\quad New\_Pop[0] \leftarrow Elitism(Lb, Ub)$
6: $\quad$ **for** j=1 to Pop_Size **do**
7: $\quad\quad Cross\_Rate \leftarrow MersenneTwister()$
8: $\quad\quad$ **if** $(Cross\_Rate \leq Cross\_Prob)$ **then**
9: $\quad\quad\quad Chromosome1 \leftarrow Selection(Lb, Ub)$
10: $\quad\quad\quad Chromosome2 \leftarrow Selection(Lb, Ub)$
11: $\quad\quad\quad Offspring \leftarrow Crossover(Chromosome1, Chromosome2)$
12: $\quad\quad$ **else**
13: $\quad\quad\quad Offspring \leftarrow Selection(Lb, Ub)$
14: $\quad\quad$ **end if**
15: $\quad\quad Mut\_Rate \leftarrow MersenneTwister()$
16: $\quad\quad$ **if** $(Mut\_Rate \leq Mut\_Prob)$ **then**
17: $\quad\quad\quad Offspring \leftarrow Mutation(Offspring)$
18: $\quad\quad$ **end if**
19: $\quad\quad New\_Pop[j] \leftarrow Offspring$
20: $\quad$ **end for**
21: $\quad Temp\_Pop \leftarrow New\_Pop$
22: **end for**
23: $Decoded\_Pop \leftarrow Encoding(Temp\_Pop)$
24: $Optimal\_Weight \leftarrow Minimum(Decoded\_Pop)$
25: **return** $Optimal\_Weight$

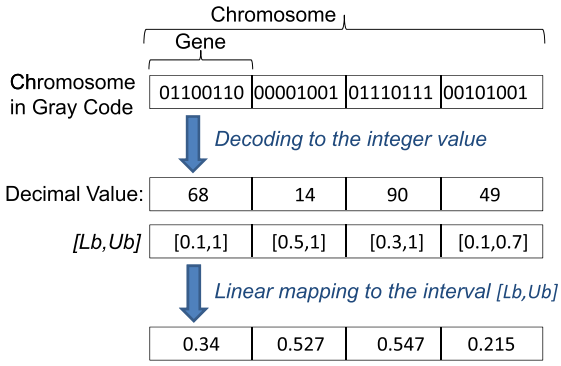


Fig. 5. Example of decoding and linear mapping.

After encoding, the algorithm starts producing a new population in line 5–20. The function $Elitism()$ in line 5 copies the best chromosome of the current population to the new population, so the best chromosome found can survive. Elitism can very rapidly increase the performance of GA, because it prevents losing the best found solution. To create other new offsprings, three basic operators including selection, crossover, and mutation are applied as follows.

Selection in GA means how to select the parents for crossover or mutation. The main idea is to select the better parents in hope that the better parents will produce the better offspring. Thus, the function $Selection()$ in the algorithm selects randomly two chromosomes from the current population, evaluates their fitness values, and finally returns the one, which has the smaller fitness value as one of the parents. Another parent is selected in the same way.

$Cross\_Prob$ in line 8 is the crossover probability, which states how often a crossover is performed. If there is a crossover, two parents' chromosomes are selected and the offspring is made from their crossover. If there is no crossover,

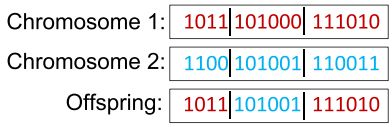


Fig. 6. Example of crossover.

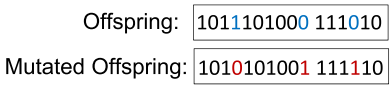


Fig. 7. Example of mutation.

the offspring is the exact copy of a chromosome from the old population. Due to $Cross\_Prob$, the new generation is a mix of offsprings made by the crossovers and the chromosomes from the old population. Although the crossovers have the tendency to improve chromosomes, it has been shown to be beneficial to keep part of the old population.

Crossover selects the genes from the parents' chromosomes and creates a new offspring. There are different ways to make a crossover. This algorithm chooses randomly two crossover points, and everything before the first point and after the second point is copied from the first parent, and the section between the two crossover points is copied from the second parent. Fig. 6 shows an example of the crossover applied in this algorithm (| denotes the crossover point).

After crossover, mutation is performed. Mut_Prob in line 16 is the mutation probability, which states how often a chromosome is mutated. If mutation is performed, the parts of the chromosome are changed. If there is no mutation, the offspring is copied after crossover without any change. Mutation is made to prevent an entire population being trapped in a local optimum. The mutation in Algorithm 3 changes the new offspring by randomly switching a few bits. It is worth mentioning that the mutation should not occur very often, because then GA will convert into a random search. Fig. 7 shows an example of mutation used in the algorithm.

This process repeats for a specified number of iterations.

As shown in Fig. 8, we have developed a tool in C++, divided into two main subtools, including End-to-End Delay Program and Optimization Program. The former derives per-flow worst case bounds by applying the proposed formal approach in Section V. The bounds are represented as the functions of weights in the WRR policy. The latter optimizes the weights in the WRR policy based on the optimization problem formulated in Section VI. Input for the first subtool includes an application communication graph, specification of flows, topology graph, routing matrix, and characteristics of routers. The outputs from the first subtool along with the set of system constraints will be the inputs for the second part. If the flows or traffic pattern are changed, the per-flow end-to-end delay bounds and the optimization problem need to be resolved. Since we aim for a design phase tool, it is executed once for static flows.

IX. Experimental Results

To evaluate the potential of our method, we applied it to two real applications and a synthetic traffic pattern on a larger network.

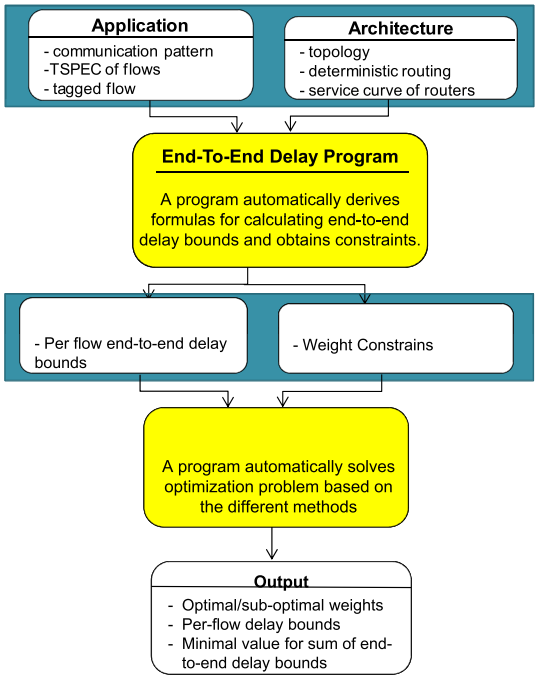


Fig. 8. Flowchart of the developed tool.

A. VOPD Application

We have applied our model to a real-time multimedia application with a random mapping to the tiles of a 4×4 on-chip network. Fig. 9 shows the task graph and flow mapping of a video object plane decoder (VOPD) [39] in which each block corresponds to an IP, and the numbers near the edges represent the bandwidth (in MB/s) of the data transfer, for a 30 frames/sec MPEG-4 movie with 1920×1088 resolution [40]. There are 21 communication flows characterized by the TSPEC.

Hence, each flow i is characterized by $(L_i, p_i, \sigma_i, \rho_i)$. The maximum transfer size and peak rate refer to the real traffic flow over the flit channel between the routers. They are constrained by the flit channel capacity. Packets may have different burst sizes. They are sent flit by flit over the flit channel. This means the maximum transfer size of 1 flit and peak rate 1 flit/cycle. Therefore, we assume L_i and p_i for all the flows are the same and equal to 1 flit and 1 flit/cycle, respectively. ρ_i is determined in flits/cycle due to associated bandwidth with flow f_i in Fig. 9 and σ_i varies between 8 and 128 flits for different flows. The length of a round in WRR scheduling L_{WR} is assumed to be ten cycles.

1) Delay Optimization: As mentioned before, the decision variables in the proposed optimization problems are the weights on shared channels. Due to the shared channels in VOPD application, 20 weights are formulated in the optimizations as a weight vector W defined as follows:

$$\begin{aligned} W = (&w_{(6,3,0,4)}, w_{(10,2,0,0)}, w_{(14,0,0,2)}, w_{(13,3,0,2)}, w_{(12,0,0,2)} \\ &w_{(9,4,0,0)}, w_{(4,3,0,4)}, w_{(4,0,0,2)}, w_{(8,2,0,0)}, w_{(8,4,0,2)} \\ &w_{(6,2,0,4)}, w_{(10,4,0,0)}, w_{(14,3,0,2)}, w_{(13,1,0,2)}, w_{(12,3,0,2)} \\ &w_{(9,3,0,0)}, w_{(4,0,0,4)}, w_{(4,4,0,2)}, w_{(8,4,0,0)}, w_{(8,0,0,2)}). \end{aligned} \tag{16}$$

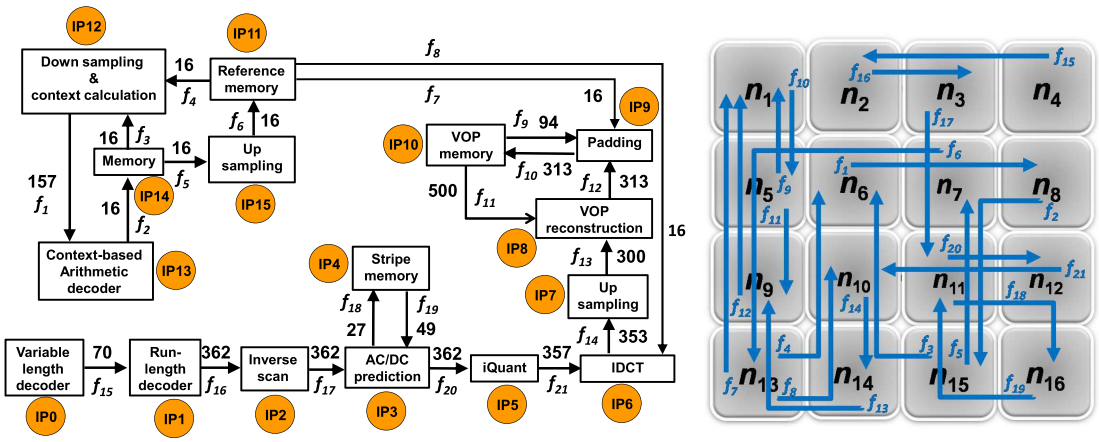


Fig. 9. VOPD application.

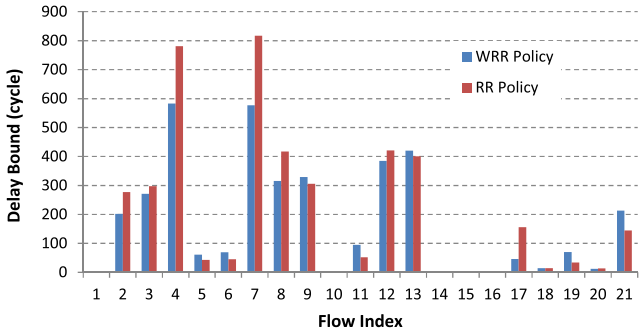


Fig. 10. Maximum worst case delay for every flow in VOPD application.

The end-to-end delay program calculates per-flow worst case bounds as the functions of weights for each flow in VOPD application and derives the corresponding constraints. The optimization program formulates the minimize-delay problem and derives the weights for the VOPD application.

To show how these weights affect the communication delay, we consider four different schemes.

1) *Random Scheme:* The weights are selected randomly.
2) *RR Scheme:* The weights have the same values to represent RR policy.
3) *Optimized Scheme:* The weights are optimized based on the optimization problem (8).
4) *Unoptimized Scheme:* The weights are not optimized and there are many unoptimized configurations. In this scheme, we allocate the weights so as to maximize the optimization problem (8) instead of minimization.

Then, the total maximum delay are calculated for different schemes and depicted in Table II. From this table, we can see that the optimized scheme leads to ∼15.4%, 48.8%, and 81.1% reduction in total maximum delay when compared with the RR, random, and unoptimized schemes, respectively. The results show that although the WRR is able to make a better performance in terms of latency than RR scheduling, if the weights are not allocated properly, it may be worse. Therefore, an appropriate weight configuration makes the WRR able to reduce total and average maximum delays by balancing the allocation of shared network bandwidth to different traffic flows with respect to their specifications and contentions for shared resources.

TABLE II
HOW GOOD ARE OPTIMIZED WEIGHTS

Scheme Type	Weight Vector	Total Worst-case Delay (*cycles*)	Average Worst-case Delay (*cycles*)
Optimized	(2, 8, 8, 2, 6, 6, 4, 2, 3, 6, 8, 2, 2, 8, 4, 4, 6, 8, 7, 4)	3671	174
Round Robin	(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	4237	202
Random	(1, 4, 2, 7, 2, 3, 9, 5, 8, 5, 9, 6, 8, 3, 8, 7, 1, 5, 2, 5)	7177	343
Unoptimized	(1, 1, 1, 9, 1, 1, 9, 9, 9, 5, 9, 9, 9, 1, 9, 9, 1, 1, 1, 5)	19432	926

To better understand the effects of the weights, the per-flow delay bounds for RR and WRR with the optimized scheme are shown in Fig. 10. This figure shows that the flows in WRR can experience the longer or shorter delays than the RR scheme, which depends on the amount of network bandwidth allocated to each flow (due to the assigned weights). However, from Table II, we can see that the total and average worst case delays are decreased in WRR with the optimized scheme, because the weights are assigned in a way to minimize the total delay, satisfy the performance constraints, and reduce the contentions for the shared resources leaving room for other contention interfering flows. Therefore, the WRR can be used to control the per-flow delay bound by controlling its assigned weight.

It is worth mentioning that the RR is a special case of the WRR (all weights equal), and will most likely be found by the optimization algorithm when it is preferable according to the defined optimization objectives.

We have investigated the accuracy of the proposed analytical model with the BookSim simulator in our previous work [4]. However, as we have extended the model to the WRR policy, we compare the per-flow delay bounds obtained from the analytical model and the BookSim simulator [32] for the optimized scheme. The simulation uses the same assumptions as explained in [4]. As shown in Fig. 11, all the delays observed in simulations are below the LUDB but not too far, suggesting that the analytical bound is fairly tight, since the simulation typically does not exercise the worst case.

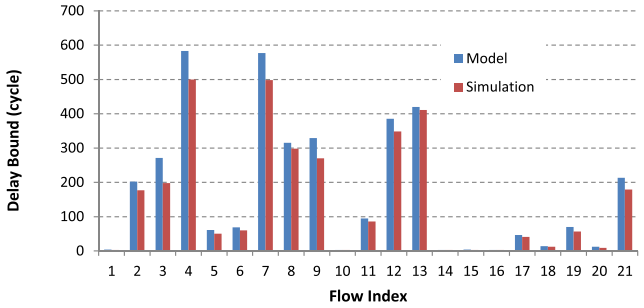


Fig. 11. Comparison of delay bounds from the proposed model and the simulator for VOPD application.

TABLE III

HOW GOOD IS MULTIOBJECTIVE OPTIMIZATION

	Weight Vector	Total Worst-case Delay (*cycles*)	Variance
Round Robin	$(5,5,5,5,5,5,5,5,5,5,$ $5,5,5,5,5,5,5,5,5,5)$	4237	59324.49
Minimize-Delay	$(2,8,8,2,6,6,4,2,3,6,$ $8,2,2,8,4,4,6,8,7,4)$	3671	35416.67
Multi-objective	$(6,9,9,1,6,7,2,1,1,6$ $4,1,1,9,4,3,8,1,1,6)$	4045	29320.71

TABLE IV

COMPARISON OF THE RUN TIME FOR DIFFERENT METHODS

Optimal point obtained by the methods	
Optimal Weight Vector	Total Delay
$(2,8,8,2,6,6,4,2,3,6,8,2,2,8,4,4,6,8,7,4)$	3671 *cycles*
Performance in different methods	
	Iteration# / Time *(sec)*
PRS	100,000 / 2.71
MMA	100,000 / 2.8
AS	100,000 / 2.85
PRS $(10+10)$	5,000 / 13
MMA $(10+10)$	5,000 / 13.37
AS $(10+10)$	5,000 / 12.83
GA	250 / 1.05

We have also computed the relative errors with respect to simulation results to consider the accuracy of the analytical model. The calculations show that the maximum and average relative errors are ∼33.33% and 16.3%, respectively.

Monitoring the delay of packets shows that the worst case delay is much larger than average-case delay, which is reasonable, because the worst case bounds cover corner cases. We have also noticed that traffic burstiness has the most influence on the delay distribution. The larger the burstiness, the larger the delay variance.

2) Multiobjective Optimization: In the multiobjective optimization minimizing the delay and the variance, we have calculated two parameters: total worst case delay and variance listed in Table III. As can be observed from Table III, the minimize-delay problem guarantees that the weight allocation is carried out in favor of minimizing the total worst case delay while there is no such guarantee for the variance over various flows. In contrast, the multiobjective optimization provides a tradeoff between such parameters.

Under the multiobjective optimization, the standard deviation is less than 0.89 of the average delay, because the variance is an explicit target for minimization. It is also fairly small

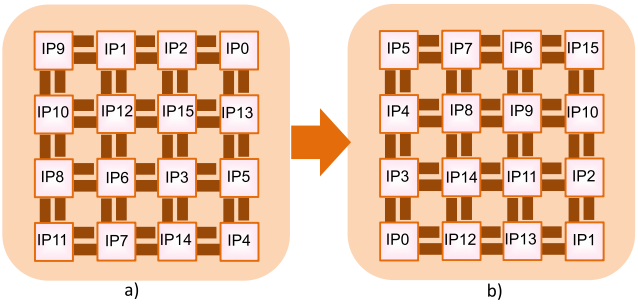


Fig. 12. Two different mappings for VOPD application. (a) Random mapping. (b) Optimized mapping.

TABLE V

COMPARISON OF TOTAL WORST CASE DELAY WITH THE RANDOM AND OPTIMIZED MAPPINGS

	Random Mapping	PERMAP Mapping
Round Robin	4237	2385
Minimize-Delay	3671	2159
Multi-objective	4045	2544

under the minimize-delay objective (standard deviation <1.1 of the average delay), because the greater imbalances of the flows (bigger variance) tend to lead to worse contention between the flows, and thus to higher average delays. Hence, the minimize-delay algorithm implicitly tends to reduce the variance as well.

Although, we have assumed the same importance for the total delay and variance in the multiobjective problem by considering the same weighting coefficients in (15), it is possible for designers to change the value of the weighting coefficients γ_1 and γ_2 to specify another relative importance of the objective functions.

3) Comparing With Other Solution Methods: As a comparative study, we implement three other metaheuristics, namely, PRS [33], MMA [34], and AS [34] to compare them with the genetic algorithm in terms of run time and efficiency. These algorithms belong to a category of metaheuristics called trajectory-based methods. A trajectory-based algorithm works on single solutions at any time, namely, it starts from an initial state (initial solution) and follows a trajectory to reach a successor solution, which may or may not belong to the neighborhood of the current solution. Population-based metaheuristics, on the contrary, deal with a set (a population) of solutions in each iteration, and in turn provide an intrinsic method for exploring the search space. The way of manipulating the population has a significant impact on the performance of these methods. Genetic algorithms belong to this category. We also extend PRS, MMA, and AS to support a population of the solutions instead of a single solution. Hereby, they produce m solutions in every iteration and select n solutions for the next iteration. The extended versions of PRS, MMA, and AS are called PRS $(m+n)$, MMA $(m+n)$, and AS $(m+n)$. Table IV presents the iteration number and run time required for solving the optimization problem (8).

The results show that all the metaheuristics presented in Table IV obtain the same solution for the problem. Therefore, we can say with some confidence that the solution is of high quality.

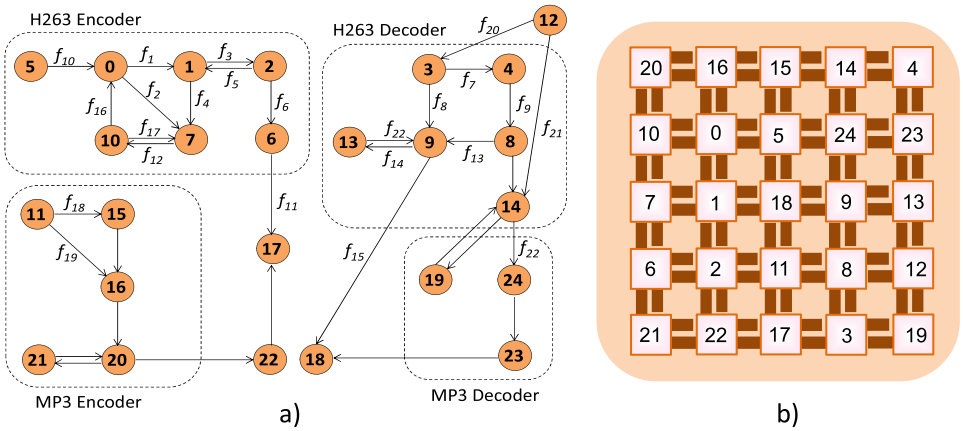


Fig. 13. MMS application. (a) Communication task graph. (b) Optimized mapping.

TABLE VI
COMPARISON AMONG DIFFERENT SCENARIOS FOR MMS APPLICATION

	Weight Vector	Total Worst-case Delay (*cycles*)	Variance
Random Scheme	(5, 4, 4, 7, 9, 3, 1, 1, 9, 9, 7, 4, 5, 2, 4, 2, 9, 9, 2, 5, 5, 2, 3, 1, 7 8, 1, 1, 3, 6, 3, 6, 8, 1, 1, 8, 5)	13509	671014
Round Robin	(5, 3, 3, 5, 5, 5, 3, 3, 5, 5, 5, 5, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 5 4, 5, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5)	4783	26324.9
Minimize-Delay	(5, 3, 4, 7, 2, 8, 1, 4, 5, 4, 7, 5, 5, 2, 4, 2, 1, 4, 2, 5, 5, 3, 3, 8, 2 5, 5, 6, 3, 5, 3, 6, 8, 9, 6, 8, 5)	4034	14832.5
Multi-objective	(9, 1, 1, 8, 2, 8, 1, 4, 4, 4, 2, 9, 1, 1, 1, 5, 1, 1, 6, 1, 1, 8, 2, 8, 2 5, 6, 6, 8, 1, 8, 9, 5, 9, 9, 4, 9)	4855	10628.4

Table IV shows that the genetic algorithm has a shorter execution time with fewer iterations. GA is not an exhaustive optimization method. However, as it is well known that the GAs provide an efficient and robust method for solving problems in which the objective function is discontinuous, nondifferentiable, or highly nonlinear and due to the results from Table IV, we believe that GA is a well suited solution method for our problem.

4) Comparing With an Optimized Mapping: By this point, we have considered a random mapping for VOPD application as shown in Fig. 12(a). To show how a good mapping affects the results from our approach, we take the optimized mapping shown in Fig. 12(b) from PERMAP algorithm [42]. Table V presents the total worst case delay parameter derived from our approach for different scenarios on these two mappings. As it can be seen, applying our technique along with a good mapping can give much better results in terms of delay minimization.

B. Multimedia Application

We have evaluated our method for a generic MultiMedia System (MMS). MMS is an integrated video/audio system that includes an H263 video encoder, an H263 video decoder, an MP3 audio encoder, and an MP3 audio decoder. This application can be partitioned into 40 concurrent tasks, and then these tasks are assigned onto 25 selected IPs [43]. These IPs range from DSPs, generic processors, and embedded DRAMs to customized application specific integrated circuits. Real video and audio clips are used as inputs to derive the communication patterns among these IPs. Fig. 13(a) reveals the communication task graph of this system [43]. We have applied the PERMAP algorithm [42] to get an optimized mapping for this system as shown in Fig. 13(b).

Due to the shared channels in the MMS application, weight vector W consisting of 37 weights is defined as

$$W = (w_{(16,1,0,0)}, w_{(15,3,0,2)}, w_{(15,0,0,2)}, w_{(5,3,0,0)}, w_{(14,4,0,2)} \; w_{(19,1,0,2)}, w_{(18,4,0,2)}, w_{(18,3,0,2)}, w_{(13,2,0,4)}, w_{(23,3,0,0)} \; w_{(14,1,0,4)}, w_{(10,2,0,4)}, w_{(1,0,0,4)}, w_{(1,3,0,4)}, w_{(11,4,0,0)} \; w_{(6,0,0,3)}, w_{(13,1,0,0)}, w_{(8,2,0,0)}, w_{(2,4,0,0)}, w_{(21,3,0,0)} \; w_{(16,4,0,0)}, w_{(15,4,0,2)}, w_{(5,4,0,0)}, w_{(14,0,0,2)}, w_{(19,4,0,2)} \; w_{(18,0,0,2)}, w_{(13,3,0,4)}, w_{(23,2,0,0)}, w_{(14,2,0,4)}, w_{(10,0,0,4)} \; w_{(1,1,0,4)}, w_{(11,2,0,0)}, w_{(6,1,0,3)}, w_{(13,4,0,0)}, w_{(8,4,0,0)} \; w_{(2,1,0,0)}, w_{(21,2,0,0)}). \quad (17)$$

Fig. 14 depicts the delay bounds for each flow under the RR policy and the optimized WRR scheme with the minimize-delay. Although the flows in WRR can experience the longer or shorter delays than under the RR scheme, the optimized WRR scheme guarantees an appropriate weight allocation in terms of minimizing the total worst case delay.

For more detail, we have presented two parameters total worst case delay and variance for different defined scenarios in Table VI. As explained before, the minimize-delay problem allocates the weights, such that guarantees total worst case delay minimization and multiobjective problem, provide a tradeoff between these two parameters.

C. Transpose Traffic Pattern

To investigate a larger network, we experiment an 8×8 mesh network under the transpose traffic pattern with 56 communication flows. The values of L and p are assumed 1 flit and 1 flit/cycle, respectively. For different flows, ρ varies between 0.001 and 0.03 flits/cycle, and σ between 2 and 128 flits.

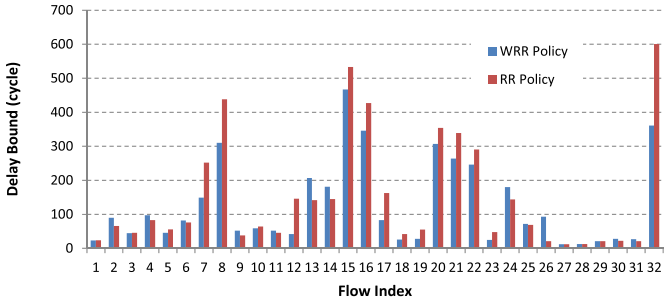


Fig. 14. Maximum worst case delay for every flow in MMS application.

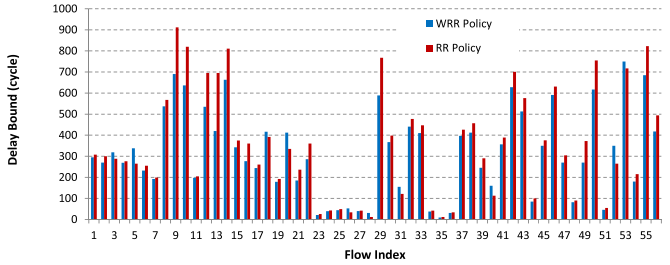


Fig. 15. Maximum worst case delay for every flow under transpose traffic pattern.

TABLE VII

LIST OF FLOWS UNDER TRANSPOSE TRAFFIC PATTERN

$f_1 : 0 \longrightarrow 63$	$f_{15} : 19 \longrightarrow 37$	$f_{29} : 63 \longrightarrow 0$	$f_{43} : 37 \longrightarrow 19$
$f_2 : 1 \longrightarrow 55$	$f_{16} : 18 \longrightarrow 45$	$f_{30} : 55 \longrightarrow 1$	$f_{44} : 45 \longrightarrow 18$
$f_3 : 2 \longrightarrow 47$	$f_{17} : 17 \longrightarrow 53$	$f_{31} : 47 \longrightarrow 2$	$f_{45} : 53 \longrightarrow 17$
$f_4 : 3 \longrightarrow 39$	$f_{18} : 16 \longrightarrow 61$	$f_{32} : 39 \longrightarrow 3$	$f_{46} : 61 \longrightarrow 16$
$f_5 : 4 \longrightarrow 31$	$f_{19} : 27 \longrightarrow 36$	$f_{33} : 31 \longrightarrow 4$	$f_{47} : 36 \longrightarrow 27$
$f_6 : 5 \longrightarrow 23$	$f_{20} : 26 \longrightarrow 44$	$f_{34} : 23 \longrightarrow 5$	$f_{48} : 44 \longrightarrow 26$
$f_7 : 6 \longrightarrow 15$	$f_{21} : 25 \longrightarrow 52$	$f_{35} : 15 \longrightarrow 6$	$f_{49} : 52 \longrightarrow 25$
$f_8 : 13 \longrightarrow 22$	$f_{22} : 24 \longrightarrow 60$	$f_{36} : 22 \longrightarrow 13$	$f_{50} : 60 \longrightarrow 24$
$f_9 : 12 \longrightarrow 30$	$f_{23} : 34 \longrightarrow 43$	$f_{37} : 30 \longrightarrow 12$	$f_{51} : 43 \longrightarrow 34$
$f_{10} : 11 \longrightarrow 38$	$f_{24} : 33 \longrightarrow 51$	$f_{38} : 38 \longrightarrow 11$	$f_{52} : 51 \longrightarrow 33$
$f_{11} : 20 \longrightarrow 29$	$f_{25} : 32 \longrightarrow 59$	$f_{39} : 29 \longrightarrow 20$	$f_{53} : 59 \longrightarrow 32$
$f_{12} : 10 \longrightarrow 46$	$f_{26} : 41 \longrightarrow 50$	$f_{40} : 46 \longrightarrow 10$	$f_{54} : 50 \longrightarrow 41$
$f_{13} : 9 \longrightarrow 54$	$f_{27} : 40 \longrightarrow 58$	$f_{41} : 54 \longrightarrow 9$	$f_{55} : 58 \longrightarrow 40$
$f_{14} : 8 \longrightarrow 62$	$f_{28} : 48 \longrightarrow 57$	$f_{42} : 62 \longrightarrow 8$	$f_{56} : 57 \longrightarrow 48$

Table VII presents the source and destination of the flows along with the index assigned to them.

Similar to previous case studies, we depict the per-flow worst case delay bounds for the RR policy and the optimized WRR scheme in Fig. 15. Regarding this figure, it is apparent that some flows in the optimized WRR policy may suffer longer delays than the RR scheme. However, the total delay bound in the optimized WRR scheme is equal to 17 610 *cycles* while it is 19 761 *cycles* in the RR scheme. These values confirm that an appropriate weight allocation can guarantee the total delay bound minimization in the network.

X. CONCLUSION

We have extended our earlier proposed methodology [4] for deriving the per-flow delay bounds under RR policy to WRR scheduling and then compared them. We have developed algorithms to automate the analysis steps. It is notable that the proposed methodologies for both the RR and the WRR do not deal with the back pressure, but we have calculated the buffer size thresholds to make sure the back-pressure does not occur in the network. Based on these analytical models, we have presented two optimization problems for weight allocation in the WRR scheduling, one for minimizing the total worst case delays and one for minimizing both the total worst case delays and their variance under performance requirements to control the per-flow delay bounds. We have also demonstrated that the proposed model exerts significant impact on communication performance. The algorithm for solving the proposed minimization problems runs very fast. For the case study, the optimized solution is found within $\sim$1 s. The contribution of this paper is providing a performance evaluation tool for designers to make a good decision at the design phase. The algorithm can be performed at run time as it is quite fast, but it needs a control infrastructure to get feedback from the network and distribute the results. On the other hand modifying the weights at run time is not easy. The way of applying the algorithm at run time can be considered as a future work. In the future, we intend to investigate the other scheduling policies. We also plan to extend the proposed analytical method in case of back pressure in the network. Zhao and Lu [41] propose analytical models to derive the worst case bounds for constant bit-rate flows due to back-pressure in the network. This paper does not consider speculative VC-allocation/switch allocation techniques. Extending the model to adjust these techniques can be considered as another future work. In this paper, we have assumed a virtual-cut-through switching as the model is suitable for the NoCs with small packets only. Small packet NoCs are relevant and important, even in practice. Extension of the model to support wormhole routing is under our investigation. There are currently some papers in our group on wormhole routing, but they consider only the average behavior of flows [19], [44].

REFERENCES

[1] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet* (Lecture Notes in Computer Science), vol. 2050. Springer Verlag, 2004.

[2] F. Jafari, A. Jantsch, and Z. Lu, "Output process of variable bit-rate flows in on-chip networks based on aggregate scheduling," in *Proc. 29th Int. Conf. Comput. Design (ICCD)*, 2011, pp. 445–446.

[3] F. Jafari, A. Jantsch, and Z. Lu, "Worst-case delay analysis of variable bit-rate flows in network-on-chip with aggregate scheduling," in *Proc. Design, Autom. Test Eur. Conf. (DATE)*, 2012, pp. 538–541.

[4] F. Jafari, Z. Lu, and A. Jantsch, "Least upper delay bound for VBR flows in networks-on-chip with virtual channels," *ACM Trans. Design Autom. Electron. Syst.*, vol. 20, no. 3, Jun. 2015, Art. no. 35.

[5] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[6] A. E. Kiasari, A. Jantsch, and Z. Lu, "Mathematical formalisms for performance evaluation of networks-on-chip," *ACM Comput. Surv.*, vol. 45, no. 3, 2013, Art. no. 38.

[7] Y. Ben-Itzhak, I. Cidon, and A. Kolodny, "Average latency and link utilization analysis of heterogeneous wormhole NoCs," *Integr., VLSI J.*, vol. 51, pp. 92–106, Sep. 2015.

[8] Z.-L. Qian, D.-C. Juan, P. Bogdan, C.-Y. Tsui, D. Marculescu, and R. Marculescu, "A support vector regression (SVR)-based latency model for network-on-chip (NoC) architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 3, pp. 471–484, Mar. 2016.

[9] Z. Qian, D.-C. Juan, P. Bogdan, C.-Y. Tsui, D. Marculescu, and R. Marculescu, "A comprehensive and accurate latency model for network-on-chip performance analysis," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2014, pp. 323–328.

[10] Z. Qian, D.-C. Juan, P. Bogdan, C.-Y. Tsui, D. Marculescu, and R. Marculescu, "SVR-NoC: A performance analysis tool for network-on-chips using learning-based support vector regression model," in *Proc. Design, Autom. Test Eur. (DATE)*, 2013, pp. 354–357.

[11] P. Bogdan, M. Kas, R. Marculescu, and O. Mutlu, "QuaLe: A quantum-leap inspired model for non-stationary analysis of NoC traffic in chip multi-processors," in *Proc. 4th Int. Symp. Netw.-Chip (NOCS)*, 2010, pp. 241–248.

[12] P. Bogdan and R. Marculescu, "Workload characterization and its impact on multicore platform design," in *Proc. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2010, pp. 231–240.
[13] P. Bogdan, "Mathematical modeling and control of multifractal workloads for data-center-on-a-chip optimization," in *Proc. 9th Int. Symp. Netw.-Chip (NOCS)*, 2015, Art. no. 21.
[14] Y. Xue and P. Bogdan, "User cooperation network coding approach for NoC performance improvement," in *Proc. 9th Int. Symp. Netw.-Chip (NOCS)*, 2015, Art. no. 17.
[15] A. Charny and J.-Y. Le Boudec, "Delay bounds in a network with aggregate scheduling," in *Proc. QofIS*, 2000, pp. 1–13.
[16] Y. Jiang, "Delay bounds for a network of guaranteed rate servers with FIFO aggregation," *Comput. Netw., Int. J. Comput. Telecommun. Netw.*, vol. 40, no. 6, pp. 683–694, 2002.
[17] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks," *Perform. Eval.*, vol. 63, no. 9, pp. 956–987, 2006.
[18] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Buffer optimization in network-on-chip through flow regulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1973–1986, Dec. 2010.
[19] Y. Qian, Z. Lu, and Q. Dou, "QoS scheduling for NoCs: Strict priority queueing versus weighted round robin," in *Proc. 28th Int. Conf. Comput. Design (ICCD)*, 2010, pp. 52–59.
[20] N. Concer, L. Bononi, M. Soulie, R. Locatelli, and L. P. Carloni, "The connection-then-credit flow control protocol for heterogeneous multicore systems-on-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 6, pp. 869–882, Jun. 2010.
[21] M. Sallam, M. W. El-Kharashi, and M. Dessouky, "The connection-then-credit flow control protocol for networks-on-chips: Implementation trade-offs," in *Proc. Int. Workshop Netw. Chip Archit. (NoCArc)*, 2014, pp. 25–30.
[22] A. Joshi and M. Mutyam, "Prevention flow-control for low latency torus networks-on-chip," in *Proc. 5th Symp. Netw. Chip (NoCS)*, 2011, pp. 41–48.
[23] J. Lin, X. Lin, and L. Tang, "Making-a-stop: A new bufferless routing algorithm for on-chip network," *J. Parallel Distrib. Comput.*, vol. 72, no. 4, pp. 515–524, Apr. 2012.
[24] N. Najib, A. Monemi, and M. N. Marsono, "Partially adaptive look-ahead routing for low latency network-on-chip," in *Proc. Student Conf. Res. Develop. (SCOReD)*, 2014, pp. 1–5.
[25] S. Ma, N. E. Jerger, and Z. Wang, "Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip," in *Proc. High-Perform. Comput. Archit. (HPCA)*, 2012, pp. 1–12.
[26] S. Ma, Z. Wang, N. E. Jerger, L. Shen, and N. Xiao, "Novel flow control for fully adaptive routing in cache-coherent NoCs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 2397–2407, Sep. 2014, doi: 10.1109/TPDS.2013.166.
[27] A. Monemi, C. Y. Ooi, and M. N. Marsono, "Low latency network-on-chip router microarchitecture using request masking technique," *Int. J. Reconfigurable Comput.*, vol. 2015, Jan. 2015, Art. no. 2.
[28] H. M. G. Wassel *et al.*, "SurfNoC: A low latency and provably non-interfering approach to secure networks-on-chip," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, 2013, pp. 583–594.
[29] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "PhaseNoC: TDM scheduling at the virtual-channel level for efficient network traffic isolation," in *Proc. Design, Autom. Test Eur. Conf. (DATE)*, 2015, pp. 1090–1095.
[30] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L. S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sep. 2007.
[31] D. P. Bertsekas, "Stochastic optimization problems with nondifferentiable cost functionals," *J. Optim. Theory Appl.*, vol. 12, no. 2, pp. 218–231, 1973.
[32] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 2013, pp. 86–96.
[33] S. H. Brooks, "A discussion of random methods for seeking maxima," *Comput. J.*, vol. 6, no. 2, pp. 244–251, 1958.
[34] R. C. White, Jr., "A survey of random methods for parameter optimization," *Simulation*, vol. 17, no. 5, pp. 197–205, 1971.
[35] J. Wroclawski, *The Use of RSVP With IETF Integrated Services*, document RFC 2210, IETF, Sep. 1997.
[36] C. A. C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowl. Inf. Syst.*, vol. 1, no. 3, pp. 269–308, 1999.
[37] P. Bajpai and M. Kumar, "Genetic algorithm—An approach to solve global optimization problems," *Indian J. Comput. Sci. Eng.*, vol. 1, no. 3, pp. 199–206, 2010.
[38] J. Guan and M. M. Aral, "Progressive genetic algorithm for solution of optimization problems with nonlinear equality and inequality constraints," *Appl. Math. Model.*, vol. 23, no. 4, pp. 329–343, 1999.
[39] D. Bertozzi *et al.*, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, Feb. 2005.
[40] E. B. van der Tol and E. G. Jaspers, "Mapping of MPEG-4 decoding on a flexible architecture platform," *Proc. SPIE*, vol. 4674, pp. 1–13, Dec. 2001.
[41] X. Zhao and Z. Lu, "Per-flow delay bound analysis based on a formalized microarchitectural model," in *Proc. 7th IEEE/ACM Int. Symp. Netw.-Chip (NoCS)*, Apr. 2013, pp. 1–8.
[42] A. E. Kiasari, S. Hessabi, and H. Sarbazi-Azad, "PERMAP: A performance-aware mapping for application-specific SoCs," in *Proc. Appl.-Specific Syst., Archit. Process. (ASAP)*, 2008, pp. 73–78.
[43] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2003, pp. 233–239.
[44] Y. Qian, Z. Lu, and W. Dou, "Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip," in *Proc. Symp. Netw.-Chip (NoCS)*, 2009, pp. 44–53.

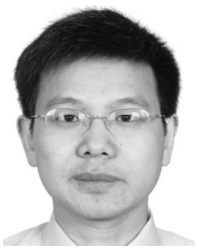
Fahimeh Jafari received the B.Sc. and M.Sc. degrees in computer engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2002 and 2005, respectively, and the Ph.D. degree in electronic and computer systems from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2015.

She joined the Department of Mathematics and Computer Science, Liverpool Hope University, Liverpool, U.K., as a Post-Doctoral Teaching Fellow, where she is currently a Lecturer (Assistant Professor). Her current research interests include design methodologies, interconnection networks, optimization theory, and performance evaluation.

Axel Jantsch (M'97) received the Dipl.-Ing. and Dr.Tech. degrees from the Technische Universität Wien, Vienna, Austria, in 1988 and 1992, respectively.

He was with Siemens Austria, Vienna, from 1995 to 1997, as a System Validation Engineer. From 1997 to 2002, he was an Associate Professor with the KTH Royal Institute of Technology, Stockholm, Sweden, where he was also a Full Professor of Electronic Systems Design from 2002 to 2014. Since 2014, he has been a Professor of Systems-on-Chip with the Institute of Computer Technology, Technische Universität Wien. He has authored over 300 articles and one book in the areas of VLSI design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks-on-chip, and has more recently turned his attention to self-awareness in cyber-physical systems.

Mr. Jantsch received the Alfred Schrödinger Scholarship from the Austrian Science Foundation for a post-doctoral research with the KTH Royal Institute of Technology from 1993 to 1995.



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

He was an Engineer in Electronic and Embedded Systems from 1989 to 2000. He was a Post-Doctoral Researcher with the KTH Royal Institute of Technology for two years, where he is currently an Associate Professor with the Department of Electronics and Embedded Systems, School for Information and Communication Technology. He has authored over 110 peer-reviewed papers in his research areas. His current research interests include on-chip networks, many-core architectures, performance analysis, and design automation.