

Delay Analysis and Buffer Optimization for Priority-Aware Networks-on-Chip (NoC)

Baoliang Li, Zeljko Zilic, Wenhua Dou,

Abstract—Networks-on-Chip (NoC) is a key component for modern Chip-MultiProcessors(CMPs) and System-on-Chip (SoC). Among all the implementation alternatives, priority-aware wormhole-switched NoC is promising to meet the rigorous requirements of on-chip latency-aware communication, e.g. cache coherent protocol and multimedia applications. The end-to-end latency and buffer requirement analysis are very important for the development of real-time applications on this platform. In this paper, we first propose a Real-Time Calculus (RTC) based performance model for the priority-aware NoC. Then, we propose an end-to-end latency analysis algorithm and a buffer optimization algorithm. Our algorithms are topology-insensitive, it takes as input the scheduling network model and the application traffic characterization, and gives the end-to-end latency and buffer requirement for each traffic flow. Our model enables the fast performance evaluation and buffer optimization of priority-aware NoC, which can be used for application mapping, routing selection and power reduction. Experiment results demonstrate the effectiveness and tightness of our model. In addition, further comparisons with other theoretical models also indicates that, our method outperforms the existing methods while the tightness of the derived delay and backlog bound are considered.

Keywords—Networks-on-Chip (NoC), priority-aware, wormhole switch, real-time calculus, delay and backlog bound

I. INTRODUCTION

Conventional interconnection paradigms, e.g. bus, ring and point-to-point links, are not able to meet the strict and complex communication requirements of modern large scale Chip-MultiProcessors (CMPs) and System-on-Chip (SoC). As an alternative, Networks-on-Chip (NoC) is proposed to provide better scalability, higher power efficiency, and low-latency global communication, etc. As a key component of CMPs and SoC, NoC must be well designed to meet the rigorous requirements on end-to-end latency, buffer constraint, throughput and power, etc. Although various proposals have been emerged, with each focusing on improving different performance metrics of on-chip communication, most of the existing research on NoC are focusing on the improvement of average performance of on-chip wormhole-switched network, and simulation is the widely used performance evaluation method. Whereas, there also exists lots of on-chip applications, which are sensitive to the worst-case or real-time communication performance of NoC, e.g. cache coherent protocol [1] and multimedia application [2]. How to design the on-chip communication

Baoliang Li and Wenhua Dou are with the College of Computer Science, National University of Defense Technology, Changsha 410073, P.R. China
 Zeljko Zilic are with Department of Electrical & Computer Engineering, McGill University, Montreal H3A-2A7, Quebec, Canada

Manuscript received XX XX, 2014; revised XX XX, 2014.

infrastructure for these applications and analyze its feasibility are of big challenges for the researchers.

To meet the rigorous Quality-of-Service (QoS) requirement, various special hardware implementations have been proposed, e.g. Time-Division Multiplexing-Access (TDMA) [3] and time-triggered switch [4], etc. Although provide strict real-time communication guarantee, the average performance and resource utilization of these proposals are very poor. In contrast, wormhole-switched NoC is widely used in on-chip network due to its simplicity and high-efficiency. Thus, providing real-time communication support on the conventional wormhole-switched NoC to meet both real-time and non-real-time communication requirements is the most promising solution. To achieve this goal, a special scheduling policy (e.g. DifServ [5] or priority-aware implementation [6][7][8]) or flow control mechanism (e.g. [9][10]) should be implemented. For all these wormhole-switch based real-time communication proposals, a key step before their adoption as a platform of real-time applications is the analysis of the worst-case communication latency for all the real-time flows, to guarantee that the deadline of each flow can be met. In addition, an effective buffer estimation approach is also needed to optimize the buffer allocation under real-time constraint.

A creditable and accurate worst-case analysis is crucial for the application of wormhole-switched NoC in real-time communication, since an over optimistic estimation will lead to the violation of deadline, while an over pessimistic estimation will make the utilization of on-chip resource very low. The conventional simulation based method might not be competent for the worst-case analysis, this is because the worst-case scenarios are hard to be captured by simulation. As an alternative, the mathematical approach can establish the relationship between performance metrics and design parameters in a very short time, and giving the worst-case performance immediately. For the worst-case analysis of fixed-priority wormhole-switched on-chip networks, Flow-Level Analysis (FLA) [6], Link-Level Analysis (LLA) [11][12] and Deterministic Network Calculus (DNC) [13] are widely used. Both FLA and LLA have their roots in the classic scheduling theory, which assume that, the traffic flows are strictly periodic, and the packet transmission delay is less than the packet inter-arrival time. In addition, they both assume that, the input buffer of wormhole-switched NoC is sufficient large, so that the back-pressure caused by flow control can be ignored. Deterministic network calculus overcomes these limitations by allowing the packet arrival at arbitrary pattern and arbitrary inter-arrival time. By applying the advanced sub-additive closure operator in dioid algebra, the large buffer assumption can also be eliminated.

However, we found that, the DNC based latency bound can be further improved, this is because, the DNC based method [13] ignored the maximal service capacity of on-chip routers, which has significant impact on the output arrival curve of traffic flow served by a specific router. The over estimated output arrival curve in [13] will further leads to a looser leftover service curve and looser performance bounds for the low-priority flows, which will be further demonstrated and explained in Section V. To overcome this shortcoming of DNC, we adopt the real-time calculus (RTC) theory [14] originally used for the real-time task scheduling analysis to build an end-to-end performance model for the wormhole-switched, credit-based flow controlled on-chip networks with RTC. Compared with DNC model [13], the RTC model in this paper use the maximal service curve and minimal arrival curve to limit the output arrival curve and left service curve, which leads to the significant improvement of performance bound. The main contribution of this paper is two folded: (1) We propose an end-to-end latency analysis algorithm for the priority-aware NoC, compared with the existing methods, e.g. LLA [11][12] and DNC [13], it can gives tighter latency bound. The output of this algorithm can be used as the reference of IP core mapping, task mapping, routing selection, or NoC parameters configuration; (2) We also propose an RTC based buffer optimization algorithm for the application-specific NoC to reduce the buffer size under deadline constraint, which can also be used to minimize the power consumption and chip area, since the buffer usually consumes about 46% power [15] and occupies 30% area [16] of entire router.

The rest of this paper is organized as follows: we present the existing real-time communication proposals and its related performance analysis methods in Section II. In Section III, the basic assumptions on wormhole switched NoC and a brief introduction to RTC theory is presented. The detailed modeling process is presented in Section IV, where we also propose an end-to-end latency analysis algorithm and a buffer optimization algorithm. We present the experiment results and comparison with other mathematical methods in Section V. Finally, we summarize our paper in Section VI.

II. RELATED WORK

Since first introduced in 2001 [17], various NoC proposals have been emerged to meet different on-chip communication requirements. The main requirements posed to NoC by on-chip applications are latency and bandwidth. To meet these demand, NoC are designed to be either either best-effort or guaranteed-service, depending on the hardware cost and application fields. The performance evaluation of these two categories include the average analysis and worst-case analysis. For the average analysis, simulation and probability-based approach hold the dominant position for both best-effort and guaranteed-service NoC. However, for the worst-case analysis, simulation is not competent due to the difficulty in covering all the corner cases. The analytical worst case analysis for these two categories is also slightly different. Synchronous Data Flow (SDF) graph [18] and DNC [19] have been presented to model the worst case performance bound of best-effort NoC. The former

method assumes the traffic flow to be periodical, and the latter one eliminates this constraint to allow the traffic to be arbitrary patterns, which extends applicable of the method. In [19], the authors build an analytical performance model with DNC taking the various contention and flow control into consideration. This result is extended in [20], where the traffic splitter was proposed to support the multi-path routing polices. Another method is presented in [21] to compute the worst-case latency for conventional wormhole switched network, and a real-time Wormhole Channel Feasibility Checking (WCFC) algorithm is proposed. This research is further extended to calculate the bandwidth and latency bound in [22], and used for topology synthesis of best-effort NoC in [23]. For more details about the mathematic modeling of best-effort NoC please refer to [24].

For the implementation of service-guaranteed NoC, a simple and effective solution to provide differential service for different applications is classify these applications into several service classes, each with different priorities, and the network provides services according to the priority of each class. Representative implementations of this idea include QNoC [25], fixed-priority NoC [6] and \mathcal{A} Ethereal [3] etc. Although all the analytical methods mentioned above aim for the worst-case analysis of conventional NoC, when they are adopted to the priority-aware NoC, the obtained performance bound is very conservative, especially for the high priority flows. This is because, these methods do not take the priority-aware scheduling into consideration. Thus, new methods should be developed for the real-time communication analysis. In [26], contention tree was proposed to analyze the feasibility of real-time traffic delivered by wormhole-switched NoC. It improves the previous results , e.g. lumped link model [10] and dependency graph model [7], by allowing the concurrent link usage. This is similar as the the LLA [11], which improves the FLA proposed in [6] by treating each link segment separately. Two buffer sizing method based on these two approach, i.e. Flow-Level Buffer-space Analysis (FLBA) and Link-Level Buffer-space Analysis (LLBA), are proposed in [12] to reduce the buffer requirement of priority-aware NoC. The main drawback of FLA and LLA is that, these method require the traffic arrival periodically, and the packet inter-arrival time must greater than the end-to-end transmission latency, and the previous research based on these two method assumes that, the buffer at each routers is sufficient large, so that the feedback caused by flow-control can be ignored.

To overcome these shortcomings, DNC was used to analyze the worst case latency of priority-aware NoC [13]. But we found that, the DNC results can be further improved if we take the maximum service curve of each router and minimum arrival curve of each flow into consideration, this is because the maximal service curve and minimum arrival curve can limit the output arrival curve and improve the leftover service curve for low-priority flows. (please refer to Theorem 1.6.2 in [27] for the reason). Motivated by this observation, we adopt the RTC [14] originally used for real-time task scheduling analysis to compute the worst-case end-to-end performance of priority-aware wormhole-switched NoC. Real-time calculus is the extension of DNC theory by integrating the maximum

service curve and lower arrival curve, which has been widely used in the modeling and analysis of network processor [28], CAN [29], FlexRay [30] and DSP systems [31], etc. To ease the application of RTC, a toolbox [32] has also been implemented to support the numerical calculation.

III. PRELIMINARIES

A. Basic Assumptions

For the detailed description of generic NoC architecture, please refer to [33]. In this paper, we consider the same network model as in [8][6][10][11]: Each router has the same number of input and output port, and each input port has sufficient FIFO buffer, i.e. Virtual Channel (VC), to accommodate all the incoming traffic of different priority level. The allocation of VC is determined by the VC allocator. To ensure the predictable transmission of message, we assume that, a deterministic routine computation module is used to determine the output port of each communication message. Crossbar is used to switch traffic from input ports to the output ports, and the admission control is performed by a switch allocator. Thus, the router micro-architecture considered here has 5 pipeline stages, i.e. Buffer-Write (BW), Route Computation (RC), VC Allocation (VA), Switch Allocation (SA), Switch Traversal (ST) and Link Traversal (LT). Communication message is broken input packets, and each packet is composed of a head flit and several non-head flits optionally. Each head flit should traverse all the five stages to find a path and reserve buffer space for the follow non-head flits, and non-head flits skip the RC and VA stage since the routine and VC have been determined by head flit.

We assume that, the switch allocator is priority-aware: if multiple flits from different input ports or different VCs of the same input port contend for the same output port, it will only grant the flit with highest priority. Flits from a lower priority can transmit a flit if and only if there are no flits from higher priority in the input buffer or the flits with higher priority are self-blocked due to the insufficiency of VC buffer at downstream router. The packet of high priority can also preempt the transmission of packets with low priority. In addition, we also assume that, the buffer depth of each VC is finite, and credit-based flow control is adopted between each router. Although we focus on the 5-stage router, our method can be easily adopted to the speculation-based router, e.g. routers with only 2 or 3 stages, the only difference is the initial delay of our model. To simplify of our analysis, we also assume that, the entire chip use synchronous design, the frequency and period of clock are f and T , respectively. Our method can also be applied to analyze Global Asynchronous Local Synchronous (GALS) NoC without any modification, because the routers located in different voltage-frequency islands can be synchronized with a half cycle fixed-latency synchronizer [34].

Our model is topology independent, but to demonstrate the basic idea of our method, we take the mesh topology as an example, as shown in Fig. 1. The router in mesh topology has five ports, corresponding to the four cardinal directions (West, East, North and South) and the Network Interface (NI), which

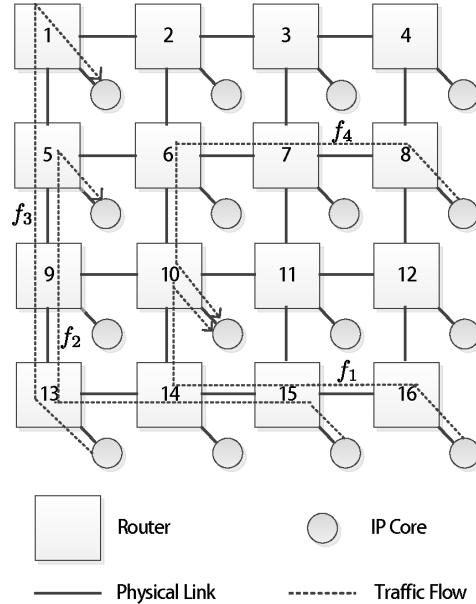


Fig. 1: Mesh topology with 4 real-time traffic flows

connects to the local Intellectual Property (IP) core. There are 4 traffic flows in the network, i.e. f_1 , f_2 , f_3 and f_4 . A flow is a sequence of packets following the same transmission path with the same source and destination address. The path of a traffic flow is defined as a sequence of routers from source NI to destination NI. We must emphasize that, although there is only four flows in the network, it is sufficient to demonstrate our idea, and our method can handle more traffic flows efficiently. To ensure the low latency transmission for real-time traffic flows, we have to assign higher priorities to these flows. Denote by P_i the priority of flow f_i , in the example shown in Fig. 1, and a higher value indicates a higher priority. As the minimum transmission unit in NoC is flit and a higher priority packet can preempt the transmission of a lower priority packet, the NoC architecture considered in this paper is flit-level preemptive [21]. Our method extends the existing methods [11][13] which assumes per-flow priority, to allows multiple flows share a same priority, e.g. both f_1 and f_4 in Fig. 1 are assigned the highest priority. Flits from different flows with the same priority are served in round-robin fashion, and the flits of the same flow were served in FIFO order.

B. Introduction to Real-Time Calculus

Real-time calculus is an extension of DNC [14], by adding the upper service curve and lower arrival curve to describe the maximal service capacity and lower arrival rate. It is the mathematical basis of Modular Performance Analysis (MPA) [35] technique used for real-time task scheduling. Here we only present the definition of RTC arrival curve and service curve, for more details about this theory, please refer to [14].

Definition 1 (Real-Time Arrival Curve): Let $R[s, t]$ denote the number of events that arrived in the time interval $[s, t]$. The lower bound and upper bound on $R[s, t]$ is called the lower

arrival curve α^l and upper arrival curve α^u which satisfy

$$\alpha^l(t-s) \leq R[s, t] \leq \alpha^u(t-s), \forall s < t$$

where $\alpha^l(0) = \alpha^u(0) = 0$. The RTC arrival curve for R is denoted as $\langle \alpha^l, \alpha^u \rangle$ for short.

From the definition, we find that the upper arrival curve (corresponding to the arrival curve of DNC [27]) and lower arrival curve are used to characterize the upper and lower bound of arrived events within any interval Δ .

Definition 2 (Real-Time Service Curve): Let $S[s, t]$ denote the total number of events that can be processed by the system in the time interval $[s, t]$. The lower bound and upper bound on $S[s, t]$ is called the lower service curve β^l and upper service curve β^u which satisfy

$$\beta^l(t-s) \leq S[s, t] \leq \beta^u(t-s), \forall s < t$$

where $\beta^l(0) = \beta^u(0) = 0$. The RTC service curve for S is denoted as $\langle \beta^l, \beta^u \rangle$ for short.

From the definition of RTC service curve, we find that the upper and lower service curve are corresponding to the maximal service curve and service curve in DNC theory [27], respectively. Thus, the concatenation theorem (see Theorem 1.46 and Theorem 1.6.1 in [27]) for service curve and maximal service curve can also be applied to RTC service curve. In this paper, we will use the discrete time RTC arrival curve and service curve to characterize the system, since the minimal time unit in the wormhole-switched NoC is clock period T . Events in arrival curve and service curve refer to the arrival and service of flits, respectively. If we obtain the arrival curve $\langle \alpha^l, \alpha^u \rangle$ of a specific traffic flow and the service curve $\langle \beta^l, \beta^u \rangle$ provided to this flow, we can get the output arrival curve $\langle \alpha'^l, \alpha'^u \rangle$ of this flow and leftover service curve $\langle \beta'^l, \beta'^u \rangle$ for the other flows with the following equations:

$$\alpha'^l = \lfloor \min\{(\alpha^l \otimes \beta^u) \otimes \beta^l, \beta^l\} \rfloor \quad (1)$$

$$\alpha'^u = \lceil \min\{(\alpha^u \otimes \beta^u) \otimes \beta^l, \beta^u\} \rceil \quad (2)$$

$$\beta'^l = (\beta^l - \alpha'^u) \bar{\otimes} 0 \quad (3)$$

$$\beta'^u = \max\{(\beta^u - \alpha'^l) \bar{\otimes} 0, 0\} \quad (4)$$

where $\lceil \cdot \rceil$, $\lfloor \cdot \rfloor$ are the ceiling operator and flooring operator, and \otimes , \oslash , $\bar{\otimes}$, $\bar{\oslash}$ are corresponding to the min-plus convolution, de-convolution, and max-plus convolution and de-convolution [27], respectively.

After we obtained the arrival curve $\langle \alpha_{f_i}^l, \alpha_{f_i}^u \rangle$ of traffic flow f_i and the service curve $\langle \beta_{R_j, f_i}^l, \beta_{R_j, f_i}^u \rangle$ provided by each router R_j to flow f_i on its path, we can obtain the end-to-end latency and backlog bound by the following equations,

$$\text{Delay}(f_i) = H(\alpha_{f_i}^u, \beta_{R_1, f_i}^l \otimes \beta_{R_2, f_i}^l \otimes \cdots \otimes \beta_{R_N, f_i}^l), \quad (5)$$

$$\text{Delay}(f_i) = V(\alpha_{f_i}^u, \beta_{R_1, f_i}^l \otimes \beta_{R_2, f_i}^l \otimes \cdots \otimes \beta_{R_N, f_i}^l) \quad (6)$$

where $H(\cdot, \cdot)$ and $V(\cdot, \cdot)$ mean the maximal vertical and horizontal deviation, respectively.

IV. DELAY ANALYSIS AND BUFFER OPTIMIZATION

In this section, we propose a delay analysis algorithm and a buffer optimization algorithm based the RTC based analytical model. While applying the RTC theory to build the analytical model of priority-aware wormhole-switched NoC, the following four aspects should be considered: (1) Only head flit need to be processed by RC and VA stage, the subsequent flits of a packet just follow the decision of head flit. We need specific mechanism to characterize the service provided to head and non-head flits in a unified way, so that we can simplify our RTC model. (2) Due to the on-chip buffer limitation, credit-based flow control is used as a back-pressure mechanism to prevent buffer overflow. Before analysis the end-to-end performance with Eq.(5) and Eq.(6), we should first break the cyclic-dependence caused by flow control. (3) Our model extends the existing approach [11][13] by allowing priority sharing between flows. Thus, the leftover service curve provided to lower priority flows can only be derived after all the flows with higher priority have been serviced. (4) To guarantee the tightness of our theoretical bound and reduce the computation complexity, we should collapse the collapsible sub-paths as far as possible. These four issues are discussed in the follow subsections.

A. Traffic Model

In a CMP or SoC system interconnected with NoC, the communication between each pair of cores was realized by transmitting packets. Each packet is further divided into flits, which is the minimum transmission unit in wormhole-switched NoC. Denote the arrival curve $\langle \alpha^l(\Delta), \alpha^u(\Delta) \rangle$ of flow as the minimum and maximum amount of flits within Δ time unit. Here, we provide two ways to obtain this arrival curve: (1) We can extract the arrival curve from the communication trace generated by a real CMP or SoC system with the sliding window method [14]. This method analyze the time series of flits by the follow way: for each window size Δ , it try to find the maximal and minimal number of arrived flits in the trace, denoted by $\alpha^l(\Delta)$ and $\alpha^u(\Delta)$, respectively. We need to mention that, the obtained arrival curve obtained by this method might not be periodic, but, it does not prevent the utilization of RTC theory. (2) If we know the inter-arrival time of packets in a flow, we can also get the arrival curve directly. For example, suppose all the packets have the same length L and arrived periodically with period I . Then, we can obtain the service curve of this flow by simply amplifying the arrival curve for periodic event stream provided in [14] at y -axis with a factor L , as shown in Fig. 2. The alert readers would notice that, the flit arrival curve obtained by this way is the same as the packet arrival curve $\mathcal{P}^L(\alpha)$ [27]. Thus, this arrival curve can also be used to compute the end-to-end packet latency, please refer to Theorem 1.7.1 in [27] for more details.

B. Router Model

While modeling the routers with RTC, we can analyze the data path stage-by-stage. After found the service curve for each stages, the service curve provided by the entire router

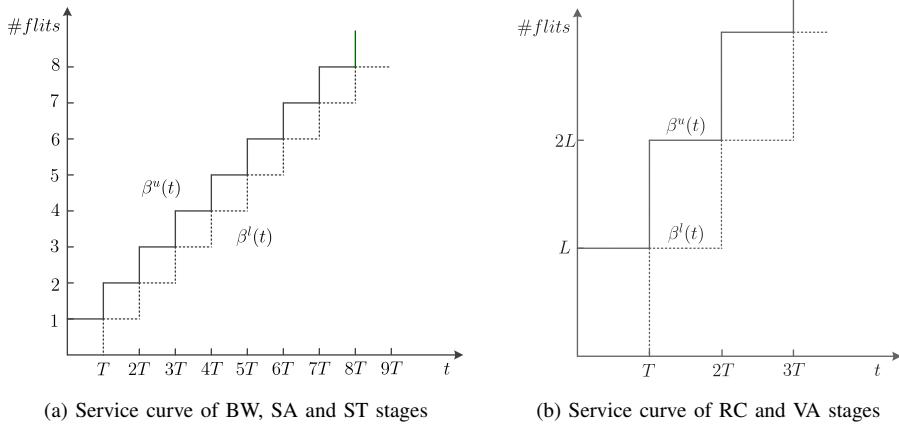


Fig. 3: Service model for each pipeline stages. The solid lines and dotted lines represent the upper service curves and lower service curves, respectively.

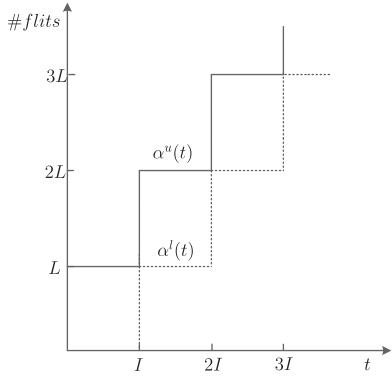


Fig. 2: Real-time calculus arrival curve for periodically arrived traffic with period I and packet length L . The solid line and dotted line represent the upper arrival curve and lower arrival curve, respectively.

to each flow can be obtained by concatenating all the service curves of these five stages. This is significantly different with the existing DNC based model [19], [13], where they treat the entire router as a whole and designate a Latency-Rate (LR) service curve for the router for simplicity. The advantage of our method is that, it can be easily modified to characterize the non-standard routers, by simply letting the service curve of non-existing stages to be infinity. Next, we try to obtain the service curve of all these 5 stages:

(1) BW stage and ST stage: all the flits within a traffic flow will traverse these two stages, and experience a fixed delay T . Take the BW stage as an example, because this stage outputs one flit at each cycle T . For any time interval of length less than T , the maximum and minimal number of flits that can be seen are one and zero, respectively. Similarly, for any time interval of length greater than T , the maximum and minimal number of flits that can be seen are two and one, respectively. The resulted service curves, i.e. $\langle \beta_{BW}^l, \beta_{BW}^u \rangle$ and $\langle \beta_{ST}^l, \beta_{ST}^u \rangle$, are shown in Fig. 3a.

(2) RC stage and VA stage: only the head flit should traverse these two stages. Suppose the packet length is L , a

sophisticated solution to construct service curve for these two stages is that, since the traverse latency for the non-head flit is 0 cycle, we can view these two stages can provide service for all the L flits within T period. Then, we can get the equivalent service curve for these two stages with the same method as BW and ST stages, i.e. $\langle \beta_{RC}^l, \beta_{RC}^u \rangle$ and $\langle \beta_{VA}^l, \beta_{VA}^u \rangle$, as shown in Fig. 3b.

(3) SA stage: each output port in the wormhole-switched NoC has a SA scheduler to schedule the switch traversal at each cycle. Thus, following the same approach as BW and ST stage, we can get the service curve $\langle \beta_{SA}^l, \beta_{SA}^u \rangle$ provided by SA stage to all the contention flows, as shown in Fig. 4a. For the fixed-priority based scheduling policy, switch allocators provide service for high priority flows first, and flows with the same priority will be served with Round-Robin order. The unserved flows will be imposed an additional latency T due to the failure of switch arbitration.

Denote by $\langle \beta_{SA,R_i,f_j}^l, \beta_{SA,R_i,f_j}^u \rangle$ the service curve provided to flow f_j by SA stage of router R_i , the equivalent service curve of router R_i provided to f_j , i.e. $\langle \beta_{R_i,f_j}^l, \beta_{R_i,f_j}^u \rangle$, can be obtained by concatenating the service curves of all the 5 stages together:

$$\beta_{R_i,f_j}^l = \beta_{BW}^l \otimes \beta_{RC}^l \otimes \beta_{VA}^l \otimes \beta_{SA,R_i,f_j}^l \otimes \beta_{ST}^l,$$

$$\beta_{R_i,f_j}^u = \beta_{BW}^u \otimes \beta_{RC}^u \otimes \beta_{VA}^u \otimes \beta_{SA,R_i,f_j}^u \otimes \beta_{ST}^u.$$

The alert readers would notice that, the contention of different flows only occurs at SA stage. Thus, if we obtained the service curve provided by SA stage to flow f_j , we can obtain the service curve of router directly. To obtain the service curve $\langle \beta_{SA,R_i,f_j}^l, \beta_{SA,R_i,f_j}^u \rangle$, we should consider the following two cases:

(a) all the flows contending with f_j at R_i have lower priorities. For the synchronize router architecture, leftover service curve after serving higher priority flows is provided to f_j totally. But for an asynchronous architecture, we should take the inference from low-priority flows into consideration. For example, a flit from f_j arriving at SA stage just after a flit from lower priority gotten granted has to stall for a cycle. At this circumstance, the actual service curve obtained by flow

f_j is

$$\beta_{SA,R_i,f_j}^l = \max\{0, \beta_{SA,R_i,f_j}^{l'} - 1\} \quad (7)$$

where $\beta_{SA,R_i,f_j}^{l'}$ is the leftover service curve calculated with Eq.(3). Similarly, a flow with higher priority than f_j can also be blocked by f_j for a cycle. Thus, denote by $\beta_{SA,R_i,f_j}^{u'}$ the service curve calculated with Eq.(4), the actual upper service curve obtained by flow f_j is

$$\beta_{SA,R_i,f_j}^u = \min\{\beta_{SA,R_i,f_j}^u, \beta_{SA,R_i,f_j}^{u'} + 1\}. \quad (8)$$

(b) there exists some contention flows with the same priority as f_j , and the leftover service curve after serving flows with high priority than f_j is $< \beta_{SA,R_i}^l, \beta_{SA,R_i}^{u'} >$. Denote by the set of contention flows at router R_i with the same priority as f_j as Θ_{R_i,f_j} , and let N_{R_i,f_j} be the number of flows in Θ_{R_i,f_j} . Then, the service curve provided to f_j is $< \lfloor \beta_{SA,R_i}^l / N_{R_i,f_j} \rfloor, \lceil \beta_{SA,R_i}^u / N_{R_i,f_j} \rceil >$, since all the flows in Θ_{R_i,f_j} got service in Round-Robin order. This is slightly different from the conventional continuous time model which is $< \beta_{SA,R_i}^l / N_{R_i,f_j}, \beta_{SA,R_i}^u / N_{R_i,f_j} >$, this is because our model is flit-level preemptive. An example with two flows with the same priority is shown in Fig. 4. After serving by all the flows in Θ_{R_i,f_j} , the leftover service capacity for low priority flows can be obtained by applying Eq.(3) and Eq.(4).

C. Upper Service Curve for Flow Controller

Credit-based flow control introduces cyclic-dependence between the adjacent routers, and leads to the self-blocking within a flow due to the insufficiency of buffer space at the downstream router. Historically, the effect of self-blocking is computed numerically by fixed-point iteration [36] or symbolically analyzed by the transformation from marked dataflow graph [37]. In this paper, we try to tackle the same problem with another solution. This is motivated by [19], where the authors abstract the flow control as a component providing a service curve, and the equivalent service curve (corresponding to the lower service curve of RTC) of this flow controller was obtained by applying some basic dioid algebra. In this section, we follow the same procedure in [19] to derive the upper service curve of flow controller. The obtained upper service curve together with the lower service curve derived in [19] enables us to break the dependence loop caused by credit-based flow control and build a comprehensive performance model with RTC. To make the question clear, we take flow f_2 as an example to demonstrate this method. The scheduling network of flow f_2 is plotted in Fig. 5, we ignore f_4 and the flow control of other flows for simplicity. We assume that, the destination NI has sufficient large input buffer, thus there is no flow controller between the router and destination NI. But, to prevent the buffer overflow, there must be a flow controller between input NI and router. Denote by the output arrival process (i.e. the amount of arrived flits by time t) of router R_9 , injection process (actually injected flits by time t) and departure process of router R_5 as $A(t)$, $I(t)$ and $D(t)$, respectively. We try to derive the service curve of flow controller at router R_9 , as shown in the following Theorem.

Theorem 1: Suppose each router provide an upper service curve β^u , the virtual channel depth of each router is B , and the

credit feedback delay is σ , then the flow controller provides an equivalent upper service curve β_τ^u , where

$$\beta_\tau^u(t) = \overline{\beta^u \otimes \delta_\sigma(t) + B}$$

and \bar{f} is the sub-additive closure of f , i.e. $\bar{f} = \inf_{n \geq 0} \{f^{(n)}\}$ and $f^{(0)} = \delta_0(t)$.¹

Proof: The feedback link can be represented as a network element providing upper service curve $\delta_\sigma(t)$. Next, we applying the same idea as [19] to derive the upper service curve this flow controller. We know that, $I(t) = \min\{A(t), D'(t) + B\}$, where $D' = D \otimes \delta_\sigma$. Denote by the service curve of router is β^u , thus $D(t) \leq I \otimes \beta^u(t)$. Bring $I(t)$ and $D'(t)$ into this equation, we get

$$\begin{aligned} D(t) &\leq I \otimes \beta^u(t) \\ &\leq \min\{A \otimes \beta^u(t), D \otimes \delta_\sigma \otimes \beta^u(t) + B\}. \end{aligned}$$

By applying Theorem 4.31 in [27], we have

$$D \leq A \otimes \beta^u \otimes \overline{\beta^u \otimes \delta_\sigma + B}.$$

Thus,

$$\begin{aligned} I &= \min\{A, D' + B\} \\ &\leq \min\{A, D \otimes \delta_\sigma + B\} \\ &\leq \min\{A, A \otimes \beta^u \otimes \overline{\beta^u \otimes \delta_\sigma + B} \otimes \delta_\sigma + B\} \\ &= \min\{A \otimes \delta_0, A \otimes (\beta^u \otimes \delta_\sigma + B) \otimes \overline{\delta_\sigma \otimes \beta^u + B}\} \\ &= A \otimes \overline{\beta^u \otimes \delta_\sigma + B}. \end{aligned}$$

which implies that, the flow controller has an equivalent upper service curve $\beta^u \otimes \delta_\sigma + B$. ■

Theorem 1 derives the upper service curve of a single flow controller, and we can get the service curve of a router chain by applying Theorem 1 iteratively. As shown in Fig. 5, the downstream routers' service curve can affect the service curve of flow controller at upstream. Hence, we should first derive the service curve of f_2 obtained at each router with the method mentioned in Subsection IV-B, and then compute the service curve of flow controller from destination to source. Service curve obtained at Router R_{15} to flow f_2 can be derived by applying Eq.(3) and Eq.(4), since f_2 is a lower priority flow at R_{15} . Router R_{13} , R_9 and R_5 provide all their service curves to f_2 because f_2 is the highest priority flow at these routers. Denote by the service curve of flow controller obtained by Theorem 1 at router R_9 is $< \beta_{\tau_9}^l, \beta_{\tau_9}^u >$, by applying the concatenation theorem, we can obtain the equivalent service curve of router R_9 are $< \beta_{R_9}^l \otimes \beta_{\tau_9}^l, \beta_{R_9}^u \otimes \beta_{\tau_9}^u >$. Then, by applying Theorem 1, we can get the service curve of flow controller at router R_{13} , R_{14} , R_{15} and NI that connected to R_{15} iteratively.

D. Collapsible Sub-Path

We have build the traffic model, router model and flow control model in the previous subsection. Before we give the performance evaluation algorithm, we first consider the follow scenario. Take f_2 and f_3 in Fig. 1 as an example, they content the output link at both router R_{13} and R_9 . Suppose $P_2 > P_3$

¹ $\delta_\sigma(t) = +\infty$ for $\forall t > \sigma$, and 0 otherwise.



Fig. 4: Service curve of SA stage for two flows with the same priority under Round-Robin scheduling policy. (a) Service capacity of SA stage. (b) Upper and lower service curve provided to each contending flow, corresponding to the solid line and dotted line; the solid green line and dotted red line represent the unrounded upper and lower service curves.

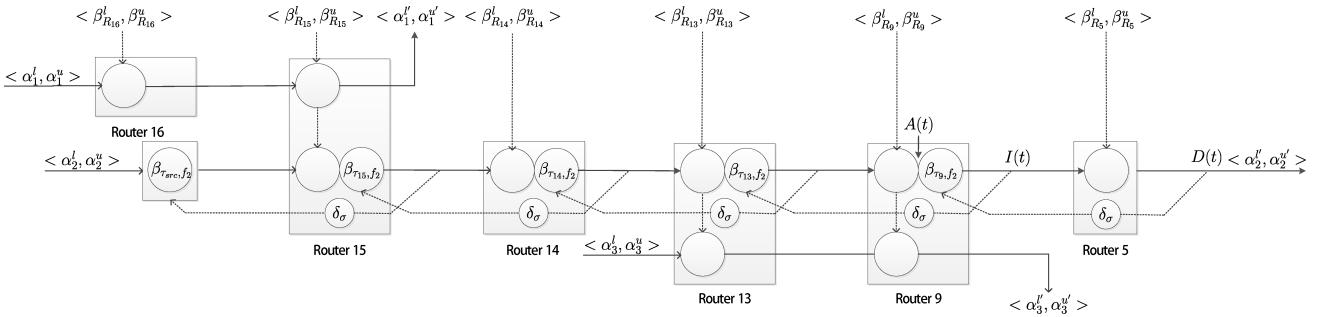


Fig. 5: Scheduling network model for flow f_2

and denote by B_{R_9, f_2} the buffer depth of R_9 allocated to flow f_2 , $\langle \beta_{R_{13}, f_2}^l, \beta_{R_{13}, f_2}^u \rangle$ and $\langle \beta_{R_9, f_2}^l, \beta_{R_9, f_2}^u \rangle$ the service curve of R_{13} and R_9 provided to flow f_2 , $\langle \beta_{\tau_{13}, f_2}^l, \beta_{\tau_{13}, f_2}^u \rangle$ the service curve for flow controller of router R_{13} , $\langle \alpha_{R_{13}, f_2}^l, \alpha_{R_{13}, f_2}^u \rangle$ and $\langle \alpha_{R_{13}, f_3}^l, \alpha_{R_{13}, f_3}^u \rangle$ the arrival curve of f_2 and f_3 at router R_{13} . The question is: when B_{R_9, f_2} is sufficiently large so that $\beta_{R_{13}, f_2}^l \otimes \beta_{\tau_{13}, f_2}^l = \beta_{R_{13}, f_2}^l$ and $\beta_{R_{13}, f_2}^u \otimes \beta_{\tau_{13}, f_2}^u = \beta_{R_{13}, f_2}^u$, how to derive the leftover service curve for f_3 at R_{13} and R_9 , and obtain a tight delay and backlog bound of f_3 efficiently?

An intuitive solution is that calculating the leftover service curve router by router: Firstly, obtaining the leftover service curve of R_{13} by applying Eq.(3) and Eq.(4); Secondly, deriving the output arrival curve $\langle \alpha_{R_9, f_2}^l, \alpha_{R_9, f_2}^u \rangle$ of f_2 by applying Eq.(1) and Eq.(2); Thirdly, the leftover service curve of R_9 can be easily obtained by applying Eq.(3) and Eq.(4); And finally, the leftover service curve for f_3 at R_{13} and R_9 can be easily obtained by concatenation theorem of service curve. Another solution is: we can substitute R_{13} and R_9 by a virtual router $R_{13,9}$ providing service curve $\langle \beta_{R_{13}, f_2}^l \otimes \beta_{R_{13}, f_2}^u, \beta_{R_{13}, f_2}^u \otimes \beta_{R_{13}, f_2}^u \rangle$, since B_{R_9, f_2} is sufficiently large and the flow control between R_{13} and R_9 can be ignored, as shown in Fig. 6. Then, the leftover service curve f_3 obtained at R_{13} and R_9 can be directly obtained by applying Eq.(3) and Eq.(4). Compared with previous method,

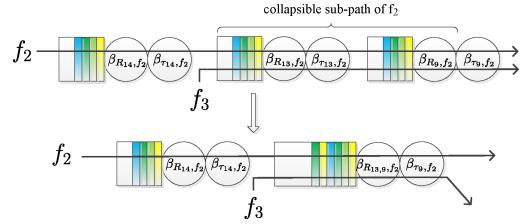


Fig. 6: Collapsible sub-path of f_2 . Since the flow control between R_{13} and R_9 can be ignored, they are replaced by a single virtual router $R_{13,9}$.

it eliminates the calculation of intermediate arrival curve, and compute the equivalent service curve by just invoke Eq.(3) and Eq.(4) once, which is calculation efficient. We will also demonstrate in Section V that, this method can achieve tighter performance bound compared with the former method.

Next, we formalize this observation and propose the concept of collapsible Sub-Path (CSP). After identifying all the CSPs on the path of a flow, the optimized performance calculation method can be applied.

Definition 3 (Collapsible Sub-Path): The collapsible sub-path of flow f , denoted by $CSP(f)$, is a sub-path of f satisfying the follow conditions:

- 1) all the routers R_i on this sub-path, except the last one,

- satisfy $\beta_{R_i,f}^l \otimes \beta_{\tau_i,f}^l = \beta_{R_i,f}^l$ and $\beta_{R_i,f}^u \otimes \beta_{\tau_i,f}^u = \beta_{R_i,f}^u$.
2) $CSP(f)$ is also a sub-path of all the flows in Ω_f , where Ω_f is the set of contention flows with lower priorities on this sub-path.

For the high priority flows, the process of calculating the end-to-end equivalent service curve is also a collapsing process. But, to compute the leftover service curve at some routers, we have to know the arrival curve at these routers. As implied previously, instead of the router-by-router calculation, we can leverage the concept of CSP and replace all the routers on $CSP(f)$ with a single virtual router providing service curve $<\otimes_{R_i \in CSP(f)} \beta_{R_i,f}^l, \otimes_{R_i \in CSP(f)} \beta_{R_i,f}^u>$. For a CSP with N routers, this method reduces $N - 1$ times calculation of arrival curve and leftover service curve, the leftover service curve calculation is greatly simplified, and the accuracy of performance bound is also improved due to the well-known “Pay-Burst-Only-Once” phenomenon in DNC theory [27].

E. End-to-End Latency

After obtained the traffic model, router model and flow control model, compute the end-to-end latency is still a non-trivial task. The follow four aspects should be considered carefully: (1) We should always compute the end-to-end latency by collapsing all the CSP to a single virtual node, as the hop-by-hop computation will lead to a looser bound; (2) In the fixed-priority flit-level preemptive NoC, only the leftover service curve can be used by the low priority flows, thus, our computation must start from higher priority flows to lower priority; (3) Before computing the leftover service curve for lower priority flows, we must ensure that all the higher priority flows has been served; (4) The computed service curve for flow controller can only be applicable for specific flow, and we should compute these curves for each flows.

Keeping these four aspects in mind, we propose the performance evaluation algorithm, as shown in Algorithm 1. Here are some explanation about our algorithm: suppose the entire NoC is represented as a directional topology graph $G : V \times E$, where V and E represent the set of routers and links, respectively. For each link $e_{i,j} \in E$ represent there is a physical channel between router R_i and router R_j , $e_{i,j} \neq e_{j,i}$ since the topology graph is a directional graph. The set of all the flows in the network is denoted as \mathcal{F} , and each flow $f_i \in \mathcal{F}$ has a fixed-priority P_i . The set of routers a flow f_i traversed is denoted as \mathcal{R}_i , and the set of links a flow f_i traversed is denoted as Γ_i . If there exists interference between flow f_i and f_j , then $\Gamma_i \wedge \Gamma_j \neq \emptyset$ and the set of contention routers is $\mathcal{R}_i \wedge \mathcal{R}_j$. The arrival curve of f_i at router R_j is $<\alpha_{R_j,f_i}^l, \alpha_{R_j,f_i}^u>$, and the leftover service curve of SA stage at router R_j is denoted as $<\beta_{SA,R_j}^l, \beta_{SA,R_j}^u>$ (initially, $\beta_{SA,R_j}^l = \beta_{SA,R_j}^l$ and $\beta_{SA,R_j}^u = \beta_{SA,R_j}^u$). For all the router R_j along the path of flow f_i , denote by the contention flows with the same priority as f_i as Θ_{R_j,f_i} , the set with lower priority flow as Ω_{R_j,f_i} , and the buffer size as B_{R_j,f_i} . In addition, let N_{R_j,f_i} and N_{f_i} be the number of flows in Θ_{R_j,f_i} and the number of routers that f_i traversed.

In Algorithm 1, the leftover service curve for lower priority flows should compute after all the service curve of higher

Algorithm 1 End-to-End Latency Analysis Algorithm

```

1: for each flow  $f_i \in \mathcal{F}$  do
2:   for each router  $R_j \in \mathcal{R}_i$  do
3:      $\beta_{SA,R_j,f_i}^l = \delta_0(t); \beta_{SA,R_j,f_i}^u = \delta_0(t);$ 
4:   end for
5: end for
6: for each flow  $f_i \in \mathcal{F}$  with priority order do
7:   for each router  $R_j \in \mathcal{R}_i$  do
8:     if  $\Theta_{R_j,f_i} \neq \emptyset$  then
9:       for each flow  $f_k \in \Theta_{R_j,f_i}$  do
10:         $\beta_{SA,R_j,f_k}^{l'} = \lfloor \beta_{SA,R_j}^l / N_{R_j,f_i} \rfloor;$ 
11:         $\beta_{SA,R_j,f_k}^{u'} = \lceil \beta_{SA,R_j}^u / N_{R_j,f_i} \rceil;$ 
12:      end for
13:    end if
14:    if  $\beta_{SA,R_j,f_i}^l = \delta_0(t)$  and  $\beta_{SA,R_j,f_i}^u = \delta_0(t)$  then
15:       $\beta_{SA,R_j,f_i}^l = \beta_{SA,R_j}^{l'}; \beta_{SA,R_j,f_i}^u = \beta_{SA,R_j}^{u'};$ 
16:    end if
17:     $\beta_{R_j,f_i}^l = \beta_{BW}^l \otimes \beta_{RC}^l \otimes \beta_{VA}^l \otimes \beta_{SA,R_j,f_i}^l \otimes \beta_{ST}^l;$ 
18:     $\beta_{R_j,f_i}^u = \beta_{BW}^u \otimes \beta_{RC}^u \otimes \beta_{VA}^u \otimes \beta_{SA,R_j,f_i}^u \otimes \beta_{ST}^u;$ 
19:  end for
20:   $\beta_\tau^l = \delta_0(t); \beta_\tau^u = \delta_0(t);$ 
21:  for each router  $R_j \in \mathcal{R}_i$  from destination to source do
22:     $\beta_{\tau_j,f_i}^l(t) = \beta_\tau^l; \beta_\tau^l = \beta_{R_j,f_i}^l \otimes \beta_\tau^l \otimes \delta_\sigma(t) + B_{R_j,f_i};$ 
23:     $\beta_{\tau_j,f_i}^u(t) = \beta_\tau^u; \beta_\tau^u = \beta_{R_j,f_i}^u \otimes \beta_\tau^u \otimes \delta_\sigma(t) + B_{R_j,f_i};$ 
24:  end for
25:  Collapse  $CSP(f_i)$  and update  $\Theta_{R_j,f_i}$ ,  $\Omega_{R_j,f_i}$  and  $\mathcal{R}_i$ ;
26:   $Delay(f_i) = H(\alpha_{f_i}^u, \beta_{R_1,f_i}^l \otimes \beta_{R_2,f_i}^l \otimes \dots \otimes \beta_{R_{N_{f_i}},f_i}^l);$ 
27:   $\beta_{f_i}^l = \delta_0(t); \beta_{f_i}^u = \delta_0(t);$ 
28:  for For all the router  $R_j$  in  $\mathcal{R}_{f_i}$  do
29:     $\beta^l = \beta_{f_i}^l \otimes \beta_{BW}^l \otimes \beta_{RC}^l \otimes \beta_{VA}^l;$ 
30:     $\beta^u = \beta_{f_i}^u \otimes \beta_{BW}^u \otimes \beta_{RC}^u \otimes \beta_{VA}^u;$ 
31:     $\alpha_{R_j,f_i}^l = [\min\{(\alpha^l \otimes \beta^u) \otimes \beta^l, \beta^l\}];$ 
32:     $\alpha_{R_j,f_i}^u = [\min\{(\alpha^u \otimes \beta^u) \otimes \beta^l, \beta^u\}];$ 
33:     $\beta_{f_i}^l = \beta_{f_i}^l \otimes \beta_{R_j,f_i}^l; \beta_{f_i}^u = \beta_{f_i}^u \otimes \beta_{R_j,f_i}^u;$ 
34:    if  $\Omega_{R_j,f_i} \neq \emptyset$  then
35:      if delay of  $\forall f_k \in \Theta_{R_j,f_i}$  has been calculated then
36:         $\alpha_{R_j,f_i}^l = \alpha_{R_j,f_i}^l + \sum_{f_k \in \Theta_{R_j,f_i}} \alpha_{R_j,f_k}^l;$ 
37:         $\alpha_{R_j,f_i}^u = \alpha_{R_j,f_i}^u + \sum_{f_k \in \Theta_{R_j,f_i}} \alpha_{R_j,f_k}^u;$ 
38:      end if
39:       $\beta_{SA,R_j}^l = (\beta_{SA,R_j}^l - \alpha_{R_j,f_i}^u) \bar{\otimes} 0;$ 
40:       $\beta_{SA,R_j}^u = \max\{(\beta_{SA,R_j}^u - \alpha_{R_j,f_i}^l), 0\};$ 
41:    end if
42:  end for
43: end for

```

priority flows have been calculated, we add the label “Calculated” to distinguish this scenario. The overall algorithm has two embedded loops, and the computation complexity for this algorithm is $O(np)$, where n and p is the number of flows and the hop count of each flow. This algorithm is of pseudo-polynomial complexity due to the computation complexity of algorithmic min-plus convolution and sub-additive closure [38]. This algorithm can be easily integrated into the RTC toolbox [32] to compute the end-to-end latency automatically.

F. Buffer Optimization

The priority-aware NoC requires the same amount of VC as the priorities, to reduce the design cost, we need to reduce either the number of VCs with priority sharing techniques [39] or the buffer depth of each VC. In [12], two backlog bound are derived with FLBA and LLBA, but these bounds are obtained with the assumption that, the buffer size is sufficient large so that the blocking caused by flow control can be ignored. In this section, we propose a buffer optimization algorithm taking the flow control into consideration to further improve the existing backlog bounds. We assume the application has been mapped onto the NoC, and each flow f_i has been assigned to their corresponding priority P_i and deadline D_i . Our aim is the reduce the buffer size under the constraint of deadline not be violated. Our algorithm can be used to optimize the buffer size of priority aware NoC in conjunction with the priority sharing techniques.

The path of flow f_i is started from source NI (denoted by $start_i$) following a sequence of routers, and ended at destination NI (denoted by end_i). For any router/NI $curnode$ on the path, denote by $Pre(curnode)$ the predecessor of $curnode$, and let $Pre(start_i) = null$. The detailed algorithm is shown in Algorithm 2. The entire algorithm optimizes the buffer size for each flow from high priority to low priority gradually. For each flow f_i , we first set the buffer size reserved for flow f_i at router R_j be B_{R_j,f_i} , which is the smallest buffer size that does not trigger the flow control. After assigned the initial buffer size for each router along the path, we reduce the buffer size gradually from end_i to $start_i$. For each iteration, checking whether the deadline is satisfied, and move $curnode$ to its predecessor $Pre(curnode)$ if the former reduction violates the deadline constraint. Iteration for flow f_i stops when $Pre(curnode) = null$. This algorithm can be implemented in RTC toolbox [32] to optimize the buffer size automatically. The entire computation complexity is $O(np)$, where n and p are the number of flows and the number of routers along the path. This complexity is pseudo-polynomial due to the end-to-end latency calculation. We can also halt the optimization when the buffer budget are met to reduce the computation procedure.

We also need to mention that, FLBA and LLBA proposed in [12] assume the packet injection period is less than the end-to-end latency, and the backlog bound obtained in [12] is the minimum buffer size that does not trigger the flow control, i.e. B_{R_j,f_i} computed in Line 12 of Algorithm 2. Our buffer optimization algorithm can also be used to reduce the router area and power consumption. But, there is significant difference between [40] and our method. In [40], a DNC based latency model was proposed and energy optimization based on this method was carried out with Dynamic Voltage and Frequency Scaling (DVFS), they minimize the energy consumption by adjusting the voltage and frequency for the fixed configuration and deadline. In contrast, our method try to optimize the buffer size under the deadline constraint, and the buffer reduction directly leads to the area and power reduction.

Algorithm 2 Buffer Optimization Algorithm

```

1: for each flow  $f_i \in \mathcal{F}$  with priority order do
2:   for each router  $R_j \in \mathcal{R}_i$  do
3:     if  $\Theta_{R_j,f_i} \neq \emptyset$  then
4:        $\beta_{R_j,f_i}^l = \beta_{BW}^l \otimes \beta_{RC}^l \otimes \beta_{VA}^l \otimes \lfloor \frac{\beta_{SA,R_j}^{l'}}{N_{R_j,f_i}} \rfloor \otimes \beta_{ST}^l;$ 
5:        $\beta_{R_j,f_i}^u = \beta_{BW}^u \otimes \beta_{RC}^u \otimes \beta_{VA}^u \otimes \lceil \frac{\beta_{SA,R_j}^{u'}}{N_{R_j,f_i}} \rceil \otimes \beta_{ST}^u;$ 
6:     else
7:        $\beta_{R_j,f_i}^l = \beta_{BW}^l \otimes \beta_{RC}^l \otimes \beta_{VA}^l \otimes \beta_{SA,R_j}^{l'} \otimes \beta_{ST}^l;$ 
8:        $\beta_{R_j,f_i}^u = \beta_{BW}^u \otimes \beta_{RC}^u \otimes \beta_{VA}^u \otimes \beta_{SA,R_j}^{u'} \otimes \beta_{ST}^u;$ 
9:     end if
10:     $B^l = \inf\{B | \beta_{R_j,f_i}^l \otimes \overline{\beta_{R_j,f_i}^l \otimes \delta_\sigma(t) + B} \geq \beta_{R_j,f_i}^l\};$ 
11:     $B^u = \inf\{B | \beta_{R_j,f_i}^u \otimes \overline{\beta_{R_j,f_i}^u \otimes \delta_\sigma(t) + B} \geq \beta_{R_j,f_i}^u\};$ 
12:     $B_{R_j,f_i} = \max\{B^l, B^u\};$ 
13:  end for
14:   $curnode = Pre(end_i);$ 
15:  while  $Pre(curnode) \neq null$  do
16:     $\beta_{f_i}^l = \bigotimes_{R_j \in \mathcal{R}_i} (\beta_{R_j,f_i}^l \otimes \beta_{R_j,f_i}^l \otimes \delta_\sigma + B_{R_j,f_i});$ 
17:    while  $H(\alpha_{f_i}^u, \beta_{f_i}^l) < D_i$  do
18:       $B_{curnode,f_i} = B_{curnode,f_i} - 1;$ 
19:       $\beta_{f_i}^l = \bigotimes_{R_j \in \mathcal{R}_i} (\beta_{R_j,f_i}^l \otimes \beta_{R_j,f_i}^l \otimes \delta_\sigma + B_{R_j,f_i});$ 
20:    end while
21:     $curnode = Pre(curnode);$ 
22:  end while
23:   $\beta_{f_i}^l = \delta_0(t); \beta_{f_i}^u = \delta_0(t);$ 
24:  for For all the router  $R_j$  in  $\mathcal{R}_{f_i}$  do
25:     $\beta^l = \beta_{f_i}^l \otimes \beta_{BW}^l \otimes \beta_{RC}^l \otimes \beta_{VA}^l;$ 
26:     $\beta^u = \beta_{f_i}^u \otimes \beta_{BW}^u \otimes \beta_{RC}^u \otimes \beta_{VA}^u;$ 
27:     $\alpha_{R_j,f_i}^l = [\min\{(\alpha^l \otimes \beta^u) \otimes \beta^l, \beta^l\}];$ 
28:     $\alpha_{R_j,f_i}^u = [\min\{(\alpha^u \otimes \beta^u) \otimes \beta^l, \beta^u\}];$ 
29:     $\beta_{f_i}^l = \beta_{f_i}^l \otimes \beta_{R_j,f_i}^l; \beta_{f_i}^u = \beta_{f_i}^u \otimes \beta_{R_j,f_i}^u;$ 
30:    if  $\Omega_{R_j,f_i} \neq \emptyset$  then
31:      if delay of  $\forall f_k \in \Theta_{R_j,f_i}$  has been calculated then
32:         $\alpha_{R_j,f_i}^l = \alpha_{R_j,f_i}^l + \sum_{f_k \in \Theta_{R_j,f_i}} \alpha_{R_j,f_k}^l;$ 
33:         $\alpha_{R_j,f_i}^u = \alpha_{R_j,f_i}^u + \sum_{f_k \in \Theta_{R_j,f_i}} \alpha_{R_j,f_k}^u;$ 
34:      end if
35:       $\beta_{SA,R_j}^{l'} = (\beta_{SA,R_j}^l - \alpha_{R_j,f_i}^u) \bar{\otimes} 0;$ 
36:       $\beta_{SA,R_j}^{u'} = \max\{(\beta_{R_j}^u - \alpha_{R_j,f_i}^l) \bar{\otimes} 0, 0\};$ 
37:    end if
38:  end for
39: end for

```

V. EXPERIMENTS

In this section, we first evaluate the tightness of our latency model compared with other analytical method. There are lots of latency analysis models for wormhole-switched real-time communication, e.g. contention tree model [26], FLA model [6], LLA model [11], DNC model [13]. In this section, we only compare our model with LLA and DNC, as they outperform all the other existing approach. Then, we consider the buffer optimization algorithm by comparing with other analytical method. There are also lots of backlog analysis models, e.g. scheduling analysis model [41] and LLBA model [12]. In

addition, the DNC model proposed in [13] can also be used to analyze the backlog bound.

The correctness of our latency model and buffer optimization algorithm is verified by simulation. We will utilize the Heterogenous-Networks-on-Chip (HNoCs) [42], a modular open source NoC simulator, to verify our analytical model. To achieve this goal, we modify the switch allocator to support fixed-priority scheduling, and collect the maximal end-to-end packet delay at each destination IP core. The network considered in this section is a 4×4 mesh, with four periodical traffic flows f_i ($i = 1, 2, 3, 4$) in the network, as shown in Fig. 1. Denote by the packet length of flow f_i is L_i (in flits) and the injection period is I_i . Packets from the same flow has the same length, but packets of different flows can be of different lengths. We can obtain the arrival curve according to the method introduced in subsection IV-A.

A. Comparison with Link Level Analysis

To ease the analysis, LLA supposes the number of bits in a flit is the same as the physical channel width, and the latency of a flit traverses a router is 1 cycle. Under these assumptions, the traversing time of a packet with F_i flits without contention, i.e. L_i , is equal to F_i . To compare with our method, we should specialize the standard 5 stages wormhole router to a single cycle router, this is achieved by letting the latency of BW, RC, VA and ST stage be zero. Thus, the service curve of entire router is the same as the service curve provided by the SA stage, which is $\langle \beta_{SA,R_i}^l, \beta_{SA,R_i}^u \rangle$. Next, we compare the end-to-end latency and buffer requirement computed with LLA and RTC.

1) *Latency Comparison*: For this scenario, we suppose the VC buffer is large enough and the credit feedback delay $\sigma = 0$, so that we can ignore the back-pressure caused by flow control and directly apply LLA for the latency analysis. There are two flows sharing the same priority in Fig. 1, i.e. f_1 and f_4 . While analyzing the latency of f_1 , we can treat f_4 as a flow with higher priority than f_1 , and vice versa. Let the injection period be packet length be 1 flit, we calculate end-to-end latency for each packet with our method and LLA under different injection period. the calculation result is shown in Table I. The RTC result is obtained by considering the $CSP(f_2)$ (i.e. R_{13} and R_9). As shown in Table I, for small injection period, the RTC result is looser than the LLA results, this is because the RTC is stateless algebra approach, and ignored some detailed information about the system.

TABLE I: Delay comparison with link level analysis

I_i	$L_i = 1$		$L_i = 2$		$L_i = 4$	
	RTC	LLA	RTC	LLA	RTC	LLA
4	6,6,6,4	6,6,6,4		∞		∞
6	5,6,6,4	5,6,5,4	9,8,9,5	7,10,9,5		∞
8	5,6,5,4	5,6,5,4	7,8,9,5	7,8,7,5		∞
10	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	15,12,15,7	11,16,19,7
12	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,15,7	11,16,15,7
14	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,11,7	11,16,15,7
16	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,11,7	11,12,11,7
18	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,11,7	11,12,11,7
20	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,11,7	11,12,11,7

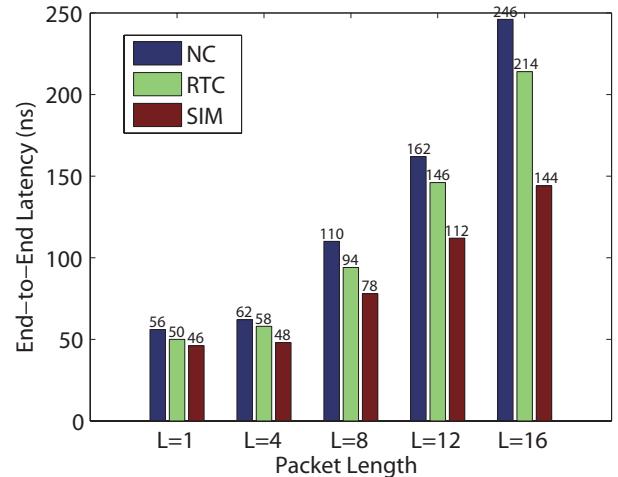


Fig. 7: Comparison with network calculus and simulation

2) *Buffer Optimization Comparison*: The LLBA [12] given the required buffer size to prevent flow control, in this section, we try to demonstrate that, the RTC method can optimize the buffer size further. Suppose the injection period $I_i = 20$ and packet size $F_i = 4$ ($i = 1, 2, 3, 4$), the end-to-end latency for the four flows are 11, 12, 11, 7, and the buffer size are (1, 1, 1, 4), (1, 1, 1, 1, 1), (4, 4, 1, 1), (1, 1, 1, 1). If we change the deadline constraint, we can optimize the buffer requirement with RTC method, as shown in Table II.

B. Comparison with Network Calculus and Simulation

In this Subsection, we present the numerical results and simulation results to demonstrate the tightness and improvement of our method. In [13], the authors derived the service curve for each contention flow at the same input router. But, these service curves can only be used to derive the latency of injection router, for the service curve of each flow at subsequent router, the service curve should be computed with the same procedure as [19]. We set the flit injection rate as 0.1 flit/cycle, buffer depth as $B = 4$ flits, credit feedback delay $\sigma = 0$, and change the packet length from 1 flit to 16 flits. The simulation results and numerical results is plotted in Fig. 7. By comparison, we find that, for flow f_1 and f_2 , the DNC method and our method get the same latency result. But, for flow f_3 , we find that, RTC can derive a much tighter latency bound compared with DNC, as shown in Fig. 7. and the difference becomes larger when we increase the packet length. One important reason is that, the simulation might not cover the corner case during the simulation process.

Next, we explain the reason why our method outperform the DNC based method proposed in [13]. As implied by Theorem 1.72 in [27], if we can take the maximal service curve into consideration, we can get a much tighter output arrival curve, then, the leftover service curve of the next router will be much tighter than the one calculated with [19]. To demonstrate this, let $B = 4$, $L = 4$ flits and $T = 2ns$, we obtained the calculated service curve for flow f_2 both with DNC and RTC, as shown in Fig. 8. The calculation was carried out with RTC toolbox.

TABLE II: buffer requirement comparison

(D_1, D_2, D_3, D_4)	f_1 $(B_{16}, B_{15}, B_{14}, B_{10})$	f_2 $(B_{15}, B_{14}, B_{13}, B_9, B_5)$	f_3 (B_{13}, B_9, B_5, B_1)	f_4 (B_8, B_7, B_6, B_{10})	Total Buffer Size
(15,15,15,15)	(1,1,1,2)	(2,1,1,1,1)	(4,4,1,1)	(1,1,1,1)	25
(17,17,17,17)	(1,1,1,2)	(2,1,1,1,1)	(3,3,1,1)	(1,1,1,1)	23
(19,19,19,19)	(1,1,1,2)	(2,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	22
(21,21,21,21)	(1,1,1,2)	(1,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	20
(23,23,23,23)	(1,1,1,1)	(1,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	19
(25,25,25,25)	(1,1,1,1)	(1,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	19
(27,27,27,27)	(1,1,1,1)	(1,1,1,1,1)	(1,1,1,1)	(1,1,1,1)	17

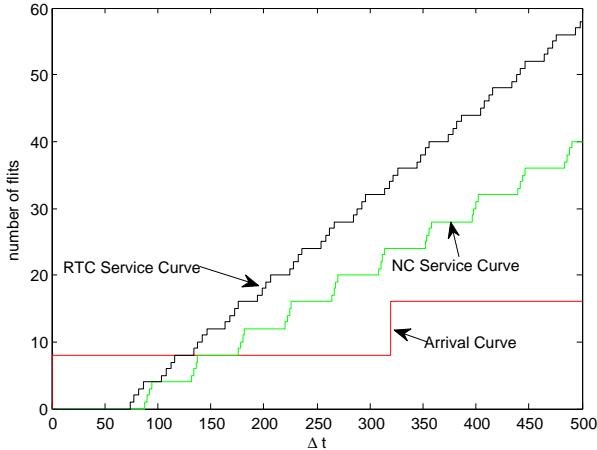


Fig. 8: Comparison of service curve computed with real-time calculus and network calculus

From Fig. 8, we find that, the calculated service curve with RTC is much ‘better’ than DNC, which explains why the RTC can produce better results.

As has shown in previous subsections, the DNC based method [13] can handle some circumstances that FLA and LLA can not be applied. This motivate us to optimize the buffer size of NoC with our method. Let the flit injection rate be 0.1 flit/cycle, packet length $L = 16$ flits and change the VC buffer depth from 4 flits to 10 flits, we compare the theoretical bound computed with DNC and our method, as shown in Fig. 9. It clear that, our method outperform the DNC based method, because it gives much tight bound for both f_2 and f_3 . Further, if we set the deadline of f_2 and f_3 to 150ns, with our method, we can found that $B = 8$ flits is sufficient enough to guarantee the delay bound. While the DNC based estimation is larger than 10 flits. We also observed from Fig. 9 that, the high priority flows (i.e. f_1 and f_4) are very insensitive to the buffer depths, which motivate us to allocate small buffers for these flows to reduce the area and power cost. We also observed that, for the priority-aware NoC, the lower priorities, the higher blocking ratios, thus leading to a larger buffer requirement.

VI. CONCLUSION

The priority-aware wormhole-switched NoC is a promising platform for the on-chip real-time communication if the worst-case performance can be accurately analyzed and guaranteed. Simulation is not competent for this purpose because it is difficult to cover all the corner cases. In this paper, we propose an

RTC based analytical model to achieve this goal. We first build the traffic model and service model for this NoC, and propose a novel method to derive the upper service curve of window flow control. Then, for the given task mapping and routing of each flow, we apply our RTC model to the follow two cases: (1) To verify whether all these flows meet their deadline under this configuration. We propose a latency analyzing algorithm to compute the end-to-end latency for each flow automatically. To improve the calculated delay bound, we proposed the concept of collapsible sub-path. (2) To optimize the buffer size of each router under the constraint of deadline. To achieve this goal, we propose an iterative algorithm to optimize the buffer from high priority flows to low priority flow gradually. These two algorithms can be implemented with RTC toolbox to perform the latency analysis and buffer optimization automatically. Experiment results demonstrate that, our method outperforms the conventional analytical methods, e.g. LLA, FLA and DNC, when the tightness of performance bound are considered. We also explain the rooted cause of this improvement. In addition, our results can also be applied to the mapping, routing and power optimization of NoC.

ACKNOWLEDGEMENT

The authors would thank the reviewers for their suggestions and comments, and all the experiments are carried out at Integrated Microsystem Lab (IML) of McGill University. This research is supported by High Technology Research and Development Program of China (Grant No. 2012AA012201, 2012AA011902).

REFERENCES

- [1] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny. The power of priority: Noc based distributed cache coherency. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 117–126, May 2007.
- [2] Jörn Ostermann, Jan Bormans, Peter List, Detlev Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer, and Thomas Wedi. Video coding with h. 264/avc: tools, performance, and complexity. *Circuits and Systems magazine, IEEE*, 4(1):7–28, 2004.
- [3] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, Sep.-Oct. 2005.
- [4] C. Paukovits and H. Kopetz. Concepts of switching in the time-triggered network-on-chip. In *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA '08. 14th IEEE International Conference on*, pages 120–129, 2008.
- [5] M.D. Harmanci, N.P. Escudero, Y. Leblebici, and P. Ilenne. Providing qos to connection-less packet-switched noc by implementing diffserv functionalities. In *System-on-Chip, 2004. Proceedings. 2004 International Symposium on*, pages 37–40, Nov 2004.

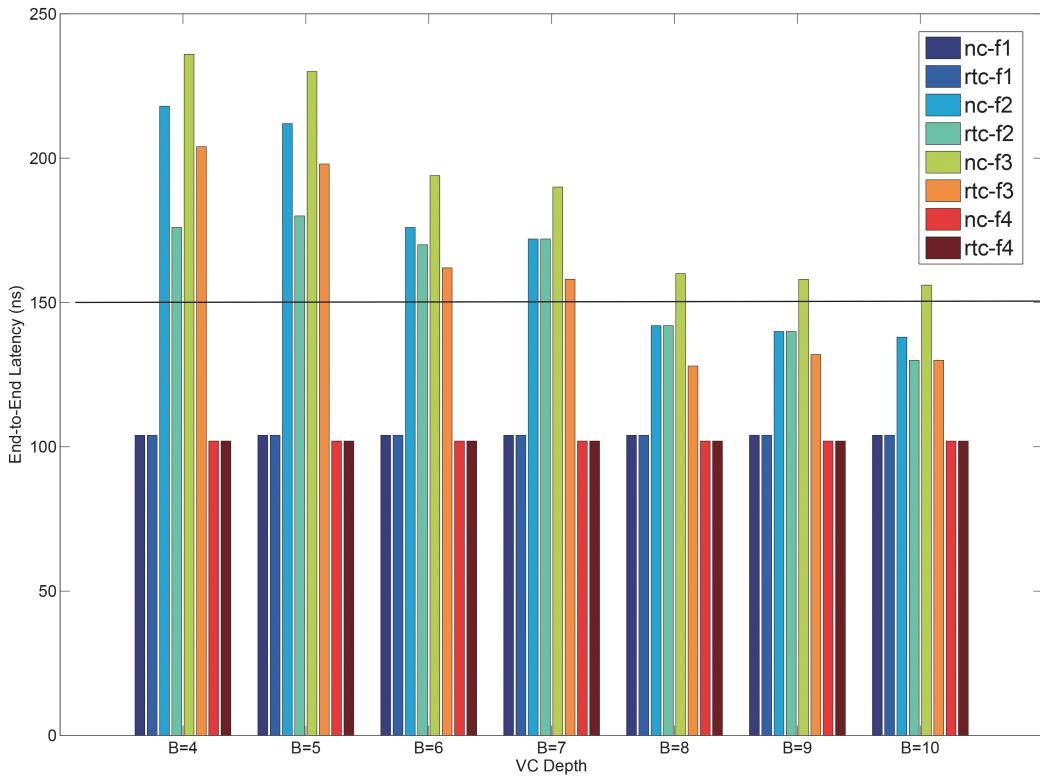


Fig. 9: End-to-end flows with different buffer size

- [6] Zheng Shi and Alan Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '08, pages 161–170, Washington, DC, USA, 2008. IEEE Computer Society.
- [7] Byungjae Kim, Jong Kim, SungJe Hong, and Sunggu Lee. A real-time communication method for wormhole switching networks. In *Parallel Processing, 1998. Proceedings. 1998 International Conference on*, pages 527–534, 1998.
- [8] S.L. Hary and F. Ozguner. Feasibility test for real-time communication using wormhole routing. *Computers and Digital Techniques, IEE Proceedings -*, 144(5):273–278, 1997.
- [9] Jong-Pyng Li and Matt W. Mutka. Real-time virtual channel flow control. *Journal of Parallel and Distributed Computing*, 32(1):49 – 65, 1996.
- [10] S. Balakrishnan and F. Ozguner. A priority-driven flow control mechanism for real-time traffic in multiprocessor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 9(7):664–678, 1998.
- [11] Hany Kashif, Hiren D. Patel, and Sebastian Fischmeister. Using link-level latency analysis for path selection for real-time communication on noCs. In *Proceedings of the Asia South Pacific Design Automation Conference (ASPDAC)*, pages 499–504, Sydney, Australia, February 2012.
- [12] Hany Kashif and Hiren Patel. Bounding buffer space requirements for real-time priority-aware networks. In *Proceedings of the Asia South Pacific Design Automation Conference (ASPDAC)*, SunTec, Singapore, January 2014.
- [13] Yue Qian, Zhonghai Lu, and Qiang Dou. QoS scheduling for noCs : Strict priority queueing versus weighted round robin. In *2010 IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN*, Proceedings IEEE International Conference on Computer Design, pages 52–59, 2010.
- [14] S. Chakraborty, S. Kunzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 190–195, 2003.
- [15] Partha Kundu. On-die interconnects for next generation cmps. In *2006 Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems*, Stanford, CA, USA, December 2006.
- [16] G. Michelogiannakis, D. Sanchez, W.J. Dally, and C. Kozyrakis. Evaluating bufferless flow control for on-chip networks. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 9–16, May 2010.
- [17] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, Jun. 2001.
- [18] P. Poplavko, T. Basten, M. Bekooij, J. van Meerbergen, and B. Mesman. Task-level timing models for guaranteed performance in multiprocessor networks-on-chip. In *Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 63–72. ACM, 2003.
- [19] Y. Qian, Z. Lu, and W. Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, pages 44–53. IEEE, 2009.
- [20] Gaoming Du, Cunqiang Zhang, Zhonghai Lu, Alberto Saggio, and Minglun Gao. Worst-case performance analysis of 2-d mesh noCs using multi-path minimal routing. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '12, pages 123–132, New York, NY, USA, 2012. ACM.
- [21] Sunggu Lee. Real-time wormhole channels. *J. Parallel Distrib. Comput.*, 63(3):299–311, March 2003.
- [22] D. Rahmati, S. Murali, L. Benini, F. Angiolini, G. De Micheli, and H. Sarbazi-Azad. Computing accurate performance bounds for best effort networks-on-chip. *Computers, IEEE Transactions on*, 62(3):452–467, 2013.
- [23] Ciprian Seiculescu, Dara Rahmati, Srinivasan Murali, Luca Benini, Giovanni De Micheli, and Hamid Sarbazi-Azad. Designing Best Effort Networks-on-Chip to Meet Hard Latency Constraints. *ACM Transactions on Embedded Computing Systems*, 12(4), 2013.
- [24] Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu. Mathematical formalisms for performance evaluation of networks-on-chip. *ACM Comput. Surv.*, 45(3):38:1–38:41, July 2013.
- [25] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture, Special Issue on Networks on Chip*, 50(2-3):105–128, Feb. 2004.
- [26] Zhonghai Lu, Axel Jantsch, and Ingo Sander. Feasibility analysis of messages for on-chip networks using wormhole routing. In *Proceedings*

- of the Asia and South Pacific Design Automation Conference*, pages 960–964, Shanghai, China, Jan. 2005.
- [27] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, Berlin, Heidelberg, 2001.
- [28] M. Gries, C. Kulkarni, C. Sauer, and K. Keutzer. Comparing analytical modeling with simulation for network processors: a case study. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 256–261 suppl., 2003.
- [29] D.B. Chokshi and P. Bhaduri. Modeling fixed priority non-preemptive scheduling with real-time calculus. In *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA '08. 14th IEEE International Conference on*, pages 387–392, 2008.
- [30] Andrei Hagiescu, Unmesh D. Bordoloi, Samarjit Chakraborty, Prahladavaradan Sampath, P. Vignesh V. Ganesan, and S. Ramesh. Performance analysis of flexray-based ecu networks. In *Proceedings of the 44th Annual Design Automation Conference, DAC '07*, pages 284–289, New York, NY, USA, 2007. ACM.
- [31] Lothar Thiele, Ernesto Wandeler, and Samarjit Chakraborty. Performance analysis of multiprocessor dsps: a stream-oriented component model. *Signal Processing Magazine, IEEE*, 22(3):38–46, 2005.
- [32] Ernesto Wandeler and Lothar Thiele. Real-time calculus (rtc) toolbox. <http://www.mpa.ethz.ch/Rtctoolbox>, 2006.
- [33] N.E. Jerger and L.S. Peh. On-chip networks. *Synthesis Lectures on Computer Architecture*, 4(1):1–141, 2009.
- [34] W.J. Dally and S.G. Tell. The even/odd synchronizer: A fast, all-digital, periodic synchronizer. In *Asynchronous Circuits and Systems (ASYNC), 2010 IEEE Symposium on*, pages 75–84, May 2010.
- [35] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis: a case study. *INTERNATIONAL JOURNAL ON SOFTWARE TOOLS FOR TECHNOLOGY TRANSFER (STTT)*, 8(6):649–667, 2006.
- [36] Henrik Schiøler, Jan Jakob Jessen, Jens Dalsgaard Nielsen, and Kim Guldstrand Larsen. Network calculus for real time analysis of embedded systems with cyclic task dependencies. In *Proc. 20th International Conference on Computers and Their Applications, CATA 2005*, pages 326–332, 2005.
- [37] Lothar Thiele and Nikolay Stoimenov. Modular performance analysis of cyclic dataflow graphs. In *Proceedings of the Seventh ACM International Conference on Embedded Software, EMSOFT '09*, pages 127–136, New York, NY, USA, 2009. ACM.
- [38] Anne Bouillard and Éric Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18:3–49, March 2008.
- [39] Zheng Shi and A. Burns. Real-time communication analysis with a priority share policy in on-chip networks. In *Real-Time Systems, 2009. ECRTS '09. 21st Euromicro Conference on*, pages 3–12, July 2009.
- [40] Jia Zhan, N. Stoimenov, Jin Ouyang, L. Thiele, V. Narayanan, and Yuan Xie. Designing energy-efficient noc for real-time embedded systems through slack optimization. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6, May 2013.
- [41] S. Manolache, P. Eles, and Zebo Peng. Buffer space optimisation with communication synthesis and traffic shaping for nos. In *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, volume 1, pages 1–6, March 2006.
- [42] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and a. kolodny. Hnocs: Modular open-source simulator for heterogeneous nos. In *Embedded Computer Systems (SAMOS), 2012 International Conference on*, pages 51–57, July 2012.