Worst-Case Communication Time Analysis of Networks-on-Chip with Shared Virtual Channels

Eberle A. Rambo, and Rolf Ernst
Institute of Computer and Network Engineering
TU Braunschweig, Germany
{rambo|ernst}@ida.ing.tu-bs.de

Abstract—**Network-on-Chip (NoC) based multi- and many-core architectures show high potential for use in real-time applications due to their superior efficiency. In real-time systems, it is necessary to guarantee that the application's timing requirements are met through the analysis of the worst-case behavior. A typical approach to guarantee real-time is the exclusive assignment of virtual channels to tasks or cores. Virtual channels, however, are a limited resource in NoCs. In future systems, there will be more tasks than virtual channels (VCs) in the network. In this paper, we propose a worst-case communication analysis of wormhole-switched best-effort NoCs (no special QoS mechanism) with SLIP arbitration and support to shared VCs. The approach is based on Compositional Performance Analysis, which enables non-symmetrical guarantees for the streams. The analysis is evaluated experimentally and compared with simulation and related work.**

I. Introduction

Network-on-Chip (NoC) based multi- and many-core architectures show high potential for use in real-time applications considering their superior efficiency. In such systems, it is necessary to provide guarantees that in the worst-case behavior, the application's timing requirements are still met.

Formal real-time analysis methods derive upper and lower bounds for latency and throughput of a network by computing the worst possible interference caused by competing traffic [1]. The worst-case latency bounds for accessing the interconnect are the base to any worst-case response time guarantee and they need to be as tight as possible in order to fully exploit the system performance (over-conservativeness leads to unnecessary low resource usage and even to an apparent unschedulability). An accurate analysis is not trivial and is a known outstanding research problem in NoC design [2].

The majority of research has focused on the analysis of traditional off-chip networks [3]–[5], which can not be directly applied to NoCs [6]. Indeed, in off-chip networks, the topology is not fixed, the behavior of nodes is often not predictable and there are different constraints, such as power and area.

Quality-of-Service (QoS) mechanisms have been widely used to enforce predictability in off- and on-chip networks, simplifying the worst-case analysis and enabling some guarantees. Time-Division Multiple Access (TDMA) has been exploited in [7], [8]. It eliminates interference by separating the transmission time of different streams in time slots. Another option is the use of priorities, investigated in [9]–[11]. By assigning different priorities to each stream, the interference that a stream may suffer is limited to the streams with higher priority. For instance, [11] analyzes the schedulability of a set of streams on a priority-based, wormhole-switched, transmission preemptive NoC, but limits one stream per virtual channel and each virtual channel must have a distinct priority. Although QoS simplifies the worst-case analysis, such mechanisms imply additional hardware overhead and lower resource utilization thus limiting the overall performance. In order to make better use of the shared resources of the network, best-effort networks are preferred.

Best-effort NoCs do not reserve resources but switch packets/flits as soon as possible and the traffic receives equal treatment. This complicates the analysis by introducing direct and indirect interference. These NoCs have been addressed in [1], [12]–[14]. The authors in [13] analyze and provide worst-case latency bounds for packet-switched best-effort NoCs. Bounds for individual traffic streams are given using network calculus. This work was extended in [12] to consider wormhole-switched NoCs. Both works consider only one buffer per input port (i.e. one virtual channel) and a simple round robin scheduler, which is not realistic for a fully-functional real-time multi-core system.

A more efficient and complex two-stage scheduler for input-buffered switches (iSLIP) [15] has been studied in [1], [14]. The work of [14] assumes worst-case traffic injection for all streams, which enables only symmetrical guarantees (i.e. same guarantees for all streams) unless QoS support is added. This limitation was overcome in [1], which models the problem as a response time analysis of multi-core processors with shared resources. The analysis yields tighter bounds and is based on Compositional Performance Analysis (CPA) [16], which enables exploiting the traffic behavior of individual streams and providing non-symmetric guarantees, i.e. a different guarantee to each stream, according to its traffic behavior. Both approaches [1], [14] support more than one virtual channel but assume that every stream uses a different virtual channel.

Real-time multi-core systems will contain more tasks than virtual channels due to the constantly increasing number of tasks mapped to a single system. The tasks will eventually need to share virtual channels. This raises new analysis challenges to be addressed. The authors in [17] present a method for global arbitration of accesses to groups of locally-arbitrated resources (e.g. virtual channels or output ports on an end-to-end path through the NoC). This corresponds to temporal virtual channel reservation which includes reservation setup time, while in our case wormhole routing serializes individual coherent flit streams. This asks for a different analysis. To the best of our knowledge, no worst-case communication time analysis method has been proposed for best-effort, two-stage arbitrated NoCs with shared virtual channels.

The **contribution** of this paper is a worst-case communi-

This work has been partially funded by the FP7 project Certification of Real Time Applications Designed for Mixed Criticality (CERTAINTY), agreement no. 288175, and by German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 – spp1500.itec.kit.edu).

cation time analysis of wormhole-switched, best-effort NoCs with two-stage iSLIP arbitration and static routing, extending [1]. No special QoS mechanisms for real-time traffic are assumed and thus all streams are treated equally. The approach supports virtual channel sharing, i.e. more than one stream can use the same virtual channel. The traffic inside a virtual channel follows a FIFO scheduling. The analysis is based on CPA, which allows specifying traffic injection patterns for each stream individually and enables non-symmetric guarantees. We assume that no back-pressure occurs, which can be validated with the worst case analysis provided in this paper.

II. The Network-on-Chip

The analysis assumes a typical NoC for real-time systems implementing wormhole switching, which allows the use of smaller buffers, and deterministic source routing, where both route and virtual channel are defined by the source. Due to wormhole switching, packets consist of fixed-sized flits and the arbitration and buffer space management are performed at a flit granularity.

The routers are input-buffered and there are separate FIFO queues for each virtual channel (VC). The flits are switched using a crossbar switch with one port for each input and output ports, which implies that at most one VC per input port can be switched at a time. The scheduling algorithm implemented by the routers is the 2-stage SLIP. The SLIP algorithm was presented in [15] for conventional networks and has been adapted for NoCs in [18]. The algorithm can be iterative (iSLIP) but the non-iterative variant is considered because only one iteration can be performed in a NoC environment that requires low arbitration latencies, and because it is hard to make safe assumptions on further iterations.

The SLIP scheduler is essentially a two-stage round-robin arbitration scheme. First, each input port sends requests to the output ports being requested by its VCs. The first stage then arbitrates at each output port and grants the request to an input port following a fixed round robin schedule. The first-stage arbitration is also aware of the states of the VCs, since it is a wormhole-switched NoC. The pointer is updated only when the input port accepts the grant, setting the lowest priority to the most recently served. The second stage then arbitrates at the input port, also based on a fixed round robin schedule, selecting one of the VCs that had its request granted in the first stage.

III. Compositional Performance Analysis

The analysis is based on Compositional Performance Analysis (CPA). It relies on independent local analyses of the system resources, such as buses and CPUs, and a global analysis loop that assembles the local results [16]. The system model of CPA is based on three components: resources, tasks, event models. Besides providing bounds to worst-case response times and jitter of tasks, CPA also provides bounds to end-to-end latencies and buffer sizes.

The Network-on-Chip is mapped to CPA as a multi-core processor with shared resources, as proposed in [1]. Each output port of the router is modeled as a *processing resource* and the input port as a *shared resource* with mutually exclusive access. The mutex-protected shared resource models the limitation of the input port of sending only one flit at a time to an output port. A traffic stream is modeled as a *chain of tasks* mapped to resources according to its path in the NoC.

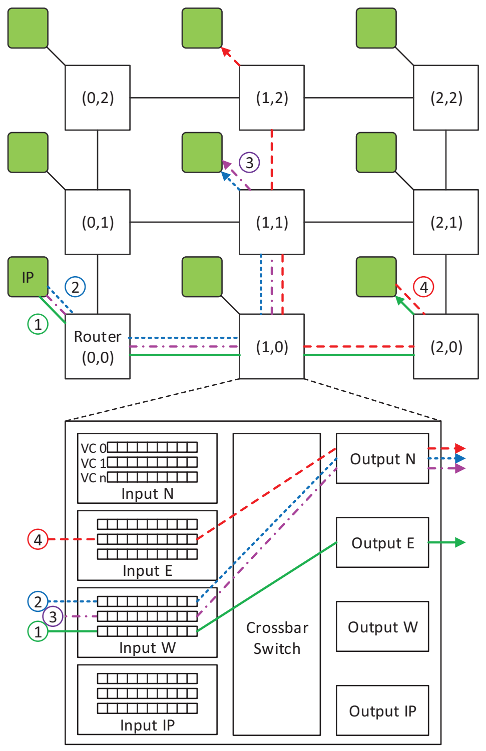

Fig. 1. 2D-mesh NoC platform and detail of router $(1, 0)$ with four streams.

Figure 1 shows an example with a 2D-mesh NoC and four streams, adapted from [1]. The lower part of Fig. 1, shows the router $(1, 0)$ in detail, where the streams enter through an input port and leave through an output port. Figure 2 shows the mapping for the router $(1, 0)$. Task τ_1 (stream 1) is mapped to the output port E (processing resource) and shares the input port W (shared resource) with τ_2 and τ_3.

The arrival of a flit at the router is a task activation at the processing resource and the transmission of a flit at an output port is the *execution of that task*. In CPA, each task is annotated with a lower and upper bound of its execution time (C_i^- and C_i^+). For the sake of simplicity and since it is the usual case in NoCs, we consider that all flits have the same transfer time ($\forall i, C_i^- = C_i^+ = C$). A task activation is triggered by events coming from other tasks (routers) or an external source (network interface). The timing of the activations in an event stream, the event model, is abstracted by event arrival curves $\eta_i^+(\Delta t)$ and $\eta_i^-(\Delta t)$, which returns the maximum and

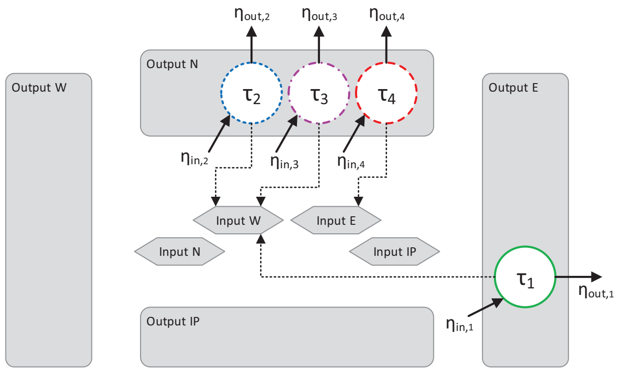

Fig. 2. Mapping for the router $(1, 0)$ from Fig. 1 to CPA components.

minimum number of events that can arrive in a given time interval Δt. Their pseudo-inverse counterpart functions $\delta_i^-(n)$ and $\delta_i^+(n)$, return the minimum and maximum time interval between the first and last of any sequence of n event arrivals. A conversion method is presented in [19].

The local analysis uses the busy windows approach (former busy period) [20]. It constructs a critical instant, which marks the beginning of the busy window and considers the worst-case arrival sequence of events (input models) and the scheduling policy to build the maximum delay that a task can suffer. This maximizes a task's response time (time from its activation until the end of its execution). The output event model is then

$$\delta_{out,i}^-(q) = \max\{(q-1)\cdot C, \delta_{in,i}^-(q) - J_i^{resp}\} \tag{1}$$

where $(q-1)\cdot C$ is the best-case execution time, $\delta_{in,i}^-(q)$ is the input event model and J_i^{resp} (Section IV-B) is the response time jitter (difference between the worst and best-case execution times).

In the global analysis loop, output event models of tasks are propagated to their dependent tasks, which are activated by the completion of other tasks, and become then their input event models. The analysis stops when all event models are stable (a fix point is reached) or when predefined constraints are violated (e.g. maximum stream latency).

IV. Worst-Case Analysis

In order to perform a worst-case communication time analysis of the NoC as a performance analysis according to CPA, we need to derive the local analysis. It requires the definition of the q-event queueing delay $w_i(q)$ (cf. busy-period in [20]). The queueing of a task τ_i is the maximum time from the beginning of the busy-window until q-th activation can be transmitted.

To conservatively account for all sources of blocking of a flit's transmission, we decompose the queueing delay into individual sources of blocking:

- Flit transfer time C: the execution time of task τ_i is determined by the execution time excluding any kind of blocking. We consider a flit transfer time equals 1 cycle because the flit has constant size.
- Direct input blocking B_i^{DI} and B_i^{DIvc}: task τ_i can be blocked by a task τ_j, which shares the same input port but not the output port ($\tau_j \in In^i$); E.g. task τ_1 suffers direct input blocking caused by τ_2 and τ_3 in Fig. 2.
- Direct output blocking B_i^{DO}: task τ_i can be blocked by a task τ_j, which shares the output port but not the input port ($\tau_j \in Out^i$); E.g. task τ_4 suffers direct output blocking caused by τ_2 and τ_3 in Fig. 2.
- Indirect output blocking B_i^{IO}: task τ_i can be indirectly blocked by a task τ_j, which shares the same output port but not the input port ($\tau_j \in Out^i$), because of another task τ_k, which shares the input port with τ_j ($\tau_k \in In^j$). E.g. task τ_4 experiences indirect input blocking through τ_2 and τ_3 caused by τ_1 in Fig. 2.
- Overlap blocking B_i^{OV} and B_i^{OVvc}: task τ_i can be blocked by a task τ_j, which shares the same input and same output port ($\tau_j \in Overlap^i$), i.e. an overlapping stream. E.g. tasks τ_2 and τ_3 block each other in Fig. 2.

In a router implementing the SLIP arbitration scheme, the queueing delay $w_i(q, a_i^q)$ is then given by the sum of all the individual sources of blocking:

$$\begin{aligned} w_i(q, a_i^q) = (q-1)\cdot C + B_i^{DIvc}(a_i^q) + B_i^{OVvc}(a_i^q) \\ +B_i^{DO}(w_i(q, a_i^q), q) + B_i^{IO}(w_i(q, a_i^q), q) \\ +B_i^{OV}(w_i(q, a_i^q)) + B_i^{DI}(w_i(q, a_i^q), q) \end{aligned} \tag{2}$$

where C is the flit transfer time in cycles and a_i^q is the arrival time of the q-th activation of τ_i.

Typically seen in busy-window based scheduling analyses, Eq. 2 forms an integer fixed point problem, as $w_i(q, a_i^q)$ appears on both sides of the equation. It can be solved iteratively, starting with $w_i(q, a_i^q) = (q-1)\cdot C$.

A. Sources of Blocking

Now we derive the individual sources of blocking. It requires defining two auxiliary functions $\rho_i^+(\Delta t)$ and $\varrho_i^-(q)$.

Let $\rho_i^+(\Delta t)$ be the maximum number of flits that can arrive in a time interval Δt at a stream i considering whole packets.

$$\rho_i^+(\Delta t) = \lceil \eta_i^+(\Delta t) \div size(i) \rceil \cdot size(i) \tag{3}$$

where $size(i)$ is the size of packet of stream i in flits.

Let $\varrho_i^-(q)$ be the minimum distance between arrivals of the first flit and the head flit of the packet containing the q-th flit at a stream i.

$$\varrho_i^-(q) = \delta_i^-\Big((\lceil q \div size(i) \rceil - 1)\cdot size(i) + 1\Big) \tag{4}$$

The **direct output blocking** $B_i^{DO}(\Delta t)$ can be bounded by

$$\begin{aligned} B_i^{DO}(\Delta t, q) = \sum_{c \in V(i)} C \cdot \min\Big\{q, \sum_{j \in Out_c^i} \eta_j^+(\Delta t)\Big\} \\ + \sum_{j \in Out_{v(i)}^i} \rho_j^+(\Delta t)\cdot C \end{aligned} \tag{5}$$

where $V(i)$ is the set of all VCs except the virtual channel used by stream i; $v(i)$ is the virtual channel used by stream i; Out_c is the set of streams mapped to the same output port and to virtual channel c; $Out_{v(i)}^i$ is the set of streams mapped to the same output port and to the same virtual channel as stream i.

The first term of Eq. 5 accounts for the blocking caused by streams using different VCs. Due to round robin, stream i may be blocked at most q times for q activations of stream i and also no more than the sum of activations of all tasks sharing a given VC, since accesses to a same VC are serialized. That is captured by the min-function. The second term of Eq. 5 accounts for the blocking caused by streams using the same VCs. Due to wormhole switching, once the scheduler at an output port grants access to a VC to an input port, no other input port can access the same VC and output port until the corresponding VC-reservation is released, i.e. the packet is fully transmitted. This is captured by ρ_j^+, which considers that, when a head flit from j arrives within the time interval Δt, the whole packet will be served before i.

The **indirect output blocking** $B_i^{IO}(\Delta t)$ is bounded by

$$\begin{aligned} B_i^{IO}(\Delta t, q) = \sum_{c \in V(i)} \min\Big\{q \cdot C, \sum_{j \in Out_c^i} B_j^{DI}(\Delta t, \eta_j^+(\Delta t))\Big\} \\ + \sum_{j \in Out_{v(i)}^i} B_j^{DI}(\Delta t, \rho_j^+(\Delta t)) \end{aligned} \tag{6}$$

This kind of blocking happens when a stream j, which shares the output port with stream i, wins the arbitration at the output port but loses the arbitration at the input port to another stream k. The first term of Eq. 6 accounts for the direct input blocking of j when $v(j) \neq v(i)$. Due to round robin on the output port, stream i may be blocked at most q times for q activations. This is captured by the min-function. The second term accounts for the direct input blocking of j when $v(j) = v(i)$. Due to wormhole switching, we have to consider the blocking suffered by the activations of entire packets of j. The cases when $v(j) = v(k)$ have no impact on the worst case, since considering any activation of k before j would only maintain or decrease the indirect blocking experienced by i.

The **blocking by overlapping streams** is given by the blocking caused by streams on other VCs $B_i^{OV}(\Delta t)$ and by streams on the same VC $B_i^{OVvc}(a_i^q)$.

The blocking caused streams on other VCs is bounded by

$$B_i^{OV}(\Delta t) = \sum_{c \in V(i)} \sum_{j \in Overlap_c^i} \eta_j^+(\Delta t) \cdot C \tag{7}$$

where $Overlap_c^i$ is the set of streams mapped to the VC c and to the same input and output ports as stream i.

Eq. 7 sums the direct interference from another overlapping stream j. Since streams i and j share the same input port and output port, they share exactly the same interferers. And because of that, all interference experienced by j has already been accounted for when analyzing the interference for i before considering overlapping streams. Moreover, the round robin scheduling does not help in the worst case, because it is possible that streams i and j "steal" the output port grant of each other, therefore we must conservatively assume that every activation of j will be served before i [1].

The blocking caused by overlapping streams sharing a VC $B_i^{OVvc}(a_i^q)$ differs from the other blocking sources because of the FIFO scheduling inherent in a virtual channel. The worst-case response-time analysis under FIFO scheduling has been studied in [21]. Thus, an upper bound for the blocking is

$$B_i^{OVvc}(a_i^q) = \sum_{j \in Overlap_{v(i)}^i} \rho_j^+(a_i^q) \cdot C \tag{8}$$

where a_i^q is the arrival time of the q-th event of stream i.

In this case, the interference is caused by complete packets, since there is no flit-interleaving between different packets in the same virtual channel. This is captured by ρ_j^+. Moreover, the packets follow a FIFO scheduling at the input port. Thus, Eq. 8 sums the largest number of packets of j that may arrive before and which will therefore be served before the q-th activation of i [21].

The **direct input blocking** is also differentiated between blocking caused by streams on other VCs $B_i^{DI}(\Delta t)$ and the blocking caused by streams on the same VC $B_i^{DIvc}(a_i^q)$.

The direct input blocking caused by streams using different virtual channels $B_i^{DI}(\Delta t)$ can be bounded by

$$B_i^{DI}(\Delta t, q) = \sum_{c \in V(i)} C \cdot \min\left\{q, \sum_{j \in In_c^i} \tilde{\eta}_j^+(\Delta t)\right\} \tag{9}$$

where In_c^i is the set of streams mapped to VC c and to the same input port as stream i, and $\tilde{\eta}_i^+(\Delta t) = \eta_i^+(\Delta t + R_i^+)$ is the the maximum number of requests that may be issued by a stream i for an input port in a given time interval Δt [1].

Due to round robin at the input port, stream i may be blocked at most q times per each VC in use, but also no more than the number of task activations seen on that VC.

The direct input blocking $B_i^{DIvc}(\Delta t)$ caused by streams using the same virtual channel needs not only to consider the transmission time of stream j's flits but also the direct and indirect output blocking that they experience. The blocking $B_i^{DIvc}(\Delta t)$ is bounded by

$$\begin{aligned} B_i^{DIvc}(a_i^q) = \sum_{j \in In_{v(i)}^i} \Big\{ & \rho_j^+(a_i^q) \cdot C \\ & + B_j^{DO}(\delta_j^-(\rho_j^+(a_i^q)) - C) + B_j^{IO}(\delta_j^-(\rho_j^+(a_i^q)) - C) \Big\} \end{aligned} \tag{10}$$

The blocking caused by other streams in the same virtual channel at the input consists on the number of packets of j that will be transmitted before i and the blocking that those packets will suffer. Since j uses the same input port as i and since his activations are included in the busy-window of i, the interference at the input port (B_i^{DI} and B_i^{DIvc}) is already being considered. The interference seen by j on the output port however needs to be included. Therefore, Eq. 10 sums the time needed to execute $\rho_j^+(a_i^q)$ activations and sums the direct and indirect output blocking experienced until the completion of those activations.

B. Metrics for a single router

After deriving the worst-case queueing time for q activations in a router, we can bound the worst-case response time $R_i(q)$ for the q-th activation of stream i by finding the maximum distance between the processing of q activations ($w_i(q, a_i^q)$) and the arrival of the q-th activation a_i^q.

$$R_i(q) = \max_{a_i^q \in A_i^q} \left\{ w_i(q, a_i^q) - a_i^q \right\} + C + O_r \tag{11}$$

where O_r is the routing overhead at the router.

The equation forms a nested optimization problem: for every number of activations q within a busy-time, all possible arrival times a_i^q of the q-th event have to be considered to find the maximum response time [21]. In the NoC, the number of these arrival candidates can be reduced to a finite amount by exploiting the fact that time can be discretized to clock cycles. It has been shown in [21] that it is sufficient to consider only arrival candidates a_i^q that coincide with activations of the interfering workload ($I(i)$). Also, considering an earlier or later activation of i would only reduce the interference.

$$A_i^q = \bigcup_{j \in I(i)} \left\{ \delta_j^-(n) \mid \delta_i^-(q) \leq \delta_j^-(n) < \varrho_i^-\big(q + size(i)\big) \right\}_{n \geq 1} \tag{12}$$

where $I(i) = In_{v(i)}^i \cup Overlap_{v(i)}^i$ is the considered interfering workload.

The **worst-case response time** or single hop latency for a flit of stream i at a router is the maximum among all $R_i(q)$.

$$R_i^+ = \max_{1 \leq q \leq \hat{q}_i} \left\{ R_i(q) \right\} \tag{13}$$

where $\hat{q}_i = \eta_i^+(\hat{w}_i)$ is the maximum number of activations in the longest busy window $\hat{w}_i$.

The response time jitter J_i^{resp} of a flit of stream i in a router is derived from the difference between it's worst- and best-case response times.

$$J_i^{resp} = R_i^+ - R_i^- \tag{14}$$

where a conservative lower bound for R_i^- is $C + O_r$.

The **worst-case service** $\beta_i^-(\Delta t)$ is the minimum number of served flits guaranteed for stream i in a time interval Δt [1].

$$\beta_i^-(\Delta t) = \max_{\forall q \geq 0}\{q \mid R_i(q) \leq \Delta t \wedge q < \eta_i^+(R_i(q))\} \tag{15}$$

The **worst-case backlog** of a stream i b_i is the maximum number of unserved flits and can be calculated as follows [1]:

$$b_i = \max\{\eta_i^+(\Delta t) - \beta_i^-(\Delta t)\} \tag{16}$$

The analysis assumes that back-pressure does not occur. This means that b_i does not exceed the available buffer capacity. It can be validated during analysis.

C. Metrics for the network

After the analysis reaches a stable point (i.e. it is schedulable), we can calculate the metrics for the whole network.

The **maximum end-to-end latency** $l_i^+(q)$ for q activations of stream i can be bounded by the time it takes to inject q flits in the network, the worst-case response time for each hop in its path, and overheads.

$$l_i^+(q) = \delta_{First(i)}^-(q) + O_p + \sum_{j \in Tasks(i)} R_j^+ \tag{17}$$

where $First(i)$ denotes the first task, $Tasks(i)$ is the set of all tasks of stream i, and O_p is the de/packetization overhead.

The **maximum injection throughput** $r_i^+(\Delta t)$ is the number of bytes that can be injected by stream i within the time interval Δt [1]. It is given by

$$r_i^+(\Delta t) = f \cdot \eta_{First(i)}^+(\Delta t) \div \Delta t \tag{18}$$

where f is the size of a flit (bytes).

V. Experiments

The proposed analysis was evaluated and compared with simulation and two other analyses: the baseline approach [1] ("baseline") and a version of [14] ("symmetric") extended to support overlapping streams. The analyses were implemented in the PyCPA framework [19]. The simulations were carried out using the OMNeT++ network simulation framework [22] and HNOCS library [23].

The system analyzed in the experiments is the one presented in the example (Figures 1 and 2), containing four streams. The system configuration is relatively simple but shows the relevant aspects of the analysis. The streams injected packets periodically in the NoC according to the following input event models: stream 1 has 2 activations (i.e. a 2-flit packet) every 9 cycles; stream 2 has 5 activations every 50 cycles (except in Figure 3, where they vary); stream 3 has 2 activations every 30 cycles; and stream 4 has 3 activations every 21 cycles. The virtual channel configuration varies according to the experiment. A 4-cycle routing overhead at each router (O_r) and a 10-cycle de/packetization (O_p) overhead were also considered.

Since the CPA consists of local analyses and the propagation of their results, we first show and compare the results of the worst-case response time of the tasks (WCRT) in a single router. Then we address the results for the whole network.

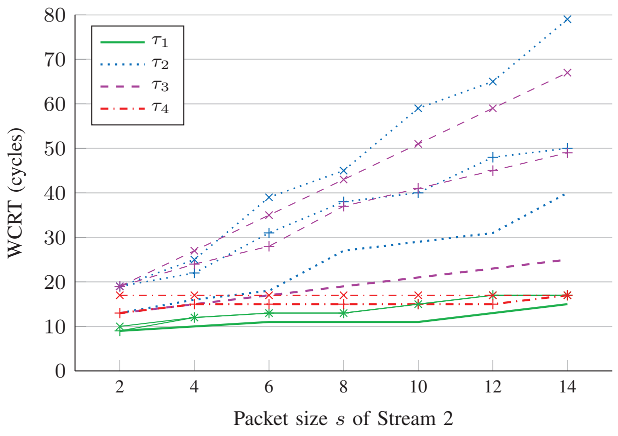

Fig. 3. Worst-case response time of tasks in router $(1, 0)$.

A. Single Router

Figure 3 shows the worst-case response times (WCRTs) for the four tasks at router $(1, 0)$, which is presented in Figures 1 and 2. In this experiment, stream 2 has s activations every $s \cdot 10$ cycles, where s increases according to the graphic. All streams are allocated to different virtual channels in order to enable the comparison to the other approaches.

Notice that the input event model of a stream at the router in question is the output event model propagated from the immediate upstream router. For instance, the input event model for τ_1 (stream 1) at router $(1, 0)$ is the output event model for stream 1 at router $(0, 0)$. As a result, effects of the jitter in the analysis can be seen.

Figure 3 compares the WCRT as the packet size s of stream 2 increases produced by ours (no mark), baseline $(+)$, and symmetric $(\times)$. The proposed analysis presented tighter bounds for all tasks except τ_4. The major improvement in this work concerns overlapping streams. Since stream 4 does not overlap (i.e. share both input and output ports) with any other stream in the current or in the previous router, the values remain the same as the baseline. Task τ_1's WCRT already shows some improvement and also shows lower impact when increasing the packet size of stream 2. This is due to its low burstiness (packet size) and low interference from the other tasks (only in router $(0, 0)$). Also, the observed improvement was actually propagated from the router upstream, since τ_1 does not share the output port with any other task in router $(1, 0)$. Tasks τ_2 and τ_3's WCRT bounds are tighter and expose the analysis' superior capability of handling overlapping streams.

B. Network

We now address the results for the whole network. Figure 4 shows the worst-case end-to-end latency for receiving a given number of flits. It contains values for ours (no mark), baseline $(+)$, and symmetric $(\times)$. In this case, the streams are also allocated to distinct virtual channels. Our approach provided tighter bounds in all cases, especially for streams that overlap more (streams 2 and 3).

Figure 5 compares the maximum end-to-end latencies given by the analysis (no mark) and the maximal ones observed in simulation $(\bigcirc)$. The latencies are shown as the virtual channel configuration is varied. For example, the configuration "1-2-3-4" means that the streams 1, 2, 3, and 4 were mapped to virtual

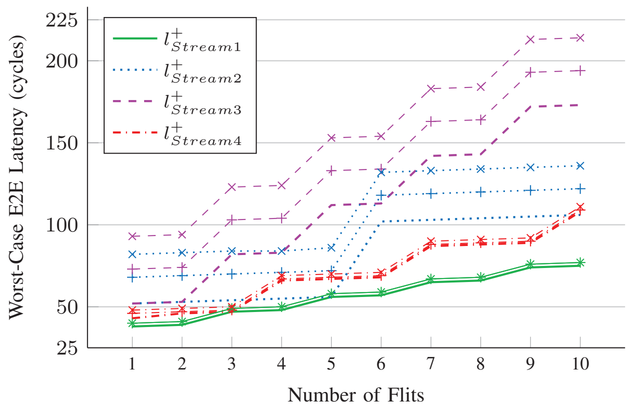

Fig. 4. Worst-case end-to-end flit latency upper bounds.

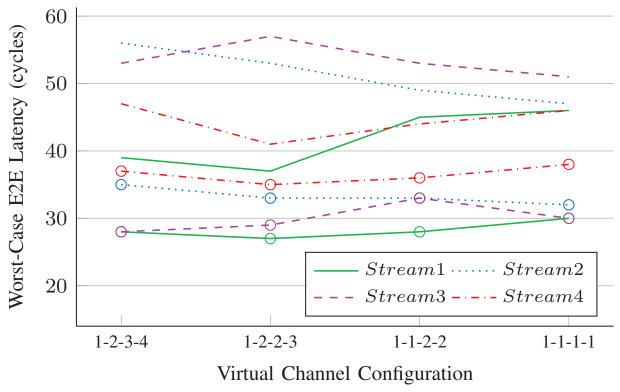

Fig. 5. Maximum end-to-end packet latency: analysis vs. simulation.

channels 1, 2, 3, and 4, respectively. Every latency observed in simulation was shorter than the computed by analysis, as expected. The fact that the maximum latency in simulation never equals the ones from analysis is due to the inability of the simulator to reach the combination of factors leading to the worst-case scenario and also due to a over-conservativeness of the analysis.

Let us now address the results for configuration "1-2-2-3", in comparison to "1-2-3-4". The worst-case latency bounds of streams 1 and 4 decrease as a result of a smaller interference caused by streams 2 and 3. Since 2 and 3 share a virtual channel in this configuration, only one request per virtual channel at a given input port (FIFO scheduled) can request access to an output port (round-robin scheduling), causing interference. The latency bound of stream 3 increases because, in the worst case, its packet will have to wait for a whole packet from stream 2 to be served instead of just some flits (FIFO in the first case and round-robin in the second). It is interesting that stream 2's bound decreases because of FIFO scheduling, as the subsequent arriving flits from stream 3 will not cause delay to packets of 2 that arrived earlier (round-robin would cause delay here).

VI. Conclusion

In this paper we have presented a worst-case communication time analysis of best-effort NoCs with a two-stage SLIP arbitration. The analysis provides worst-case latency bounds for streams individually and is able to do so also for streams that share the same virtual channel and the same path. We provided experimental evaluation and validation against simulation, which showed that the values are conservative and tighter than previous work. Techniques, such as high level arbitration, can profit from the tight bounds from this analysis in order to provide their own worst-case bounds.

References

[1] J. Diemer, J. Rox, M. Negrean, S. Stein, and R. Ernst, "Real-Time Communication Analysis for Networks with Two-Stage Arbitration," in *EMSOFT'11*, October 2011.

[2] R. Marculescu, U. Ogras, L. Peh, N. Jerger, and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, p. 3, 2009.

[3] S. Han, K. Kang, and J. Ryu, "Determination of delay bound over multi-hop real-time switches with virtual output queuing," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, May 2014, pp. 892–898.

[4] S. Lee, "Real-time wormhole channels," *Journal of Parallel and Distributed Computing*, vol. 63, no. 3, pp. 299–311, 2003.

[5] B. Kim, J. Kim, S. Hong, and S. Lee, "A real-time communication method for wormhole switching networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, no. 12, 2002.

[6] L. Benini and G. De Micheli, "Powering networks on chips: Energy-efficient and reliable interconnect design for SoCs," in *Proceedings of the 14th International Symposium on Systems Synthesis*, ser. ISSS '01. New York, NY, USA: ACM, 2001, pp. 33–38.

[7] K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE DESIGN & TEST*, vol. 22, no. 5, pp. 414–421, 2005.

[8] Q. Wang, S. Gopalakrishnan, X. Liu, and L. Sha, "A switch design for real-time industrial networks," in *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE*, 2008.

[9] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS Architecture and Design Process for Network on Chip," *J. Syst. Archit.*, vol. 50, no. 2-3, pp. 105–128, 2004.

[10] J. Lee, M. C. Ng, and K. Asanovic, "Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks," in *ISCA*, 2008.

[11] Z. Shi and A. 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*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 161–170.

[12] 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*, May 2009, pp. 44–53.

[13] ——, "Analysis of worst-case delay bounds for on-chip packet-switching networks," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 5, pp. 802–815, May 2010.

[14] S. Gopalakrishnan, M. Caccamo, and L. Sha, "Switch scheduling and network design for real-time systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 289–300.

[15] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 2, pp. 188–201, 1999.

[16] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System Level Performance Analysis–the SymTA/S Approach," *IEE Proceedings-Computers and Digital Techniques*, vol. 152, no. 2, 2005.

[17] A. Kostrzewa, S. Tobuschat, P. Axer, and R. Ernst, "Supervised sharing of virtual channels in networks-on-chip," in *In Proc. of SIES*, 2014.

[18] T. Kranich and M. Berekovic, "NoC switch with credit based guaranteed service support qualified for GALS systems," in *13th Euromicro Conference on Digital System Design*. CPS, 2010.

[19] J. Diemer, P. Axer, and R. Ernst, "Compositional performance analysis in python with PyCPA," *Proc. of WATERS*, 2012.

[20] K. Tindell, A. Burns, and A. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," *Real-Time Systems*, vol. 6, no. 2, 1994.

[21] J. Diemer, D. Thiele, and R. Ernst, "Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching," in *Industrial Embedded Systems (SIES), 2012 7th IEEE International Symposium on*, June 2012, pp. 1–10.

[22] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM2001)*, vol. 9, 2001, p. 185.

[23] Y. Ben-Itzhak, E. Zahavi, I. Cidon *et al.*, "HNOCS: modular open-source simulator for heterogeneous NoCs," in *Embedded Computer Systems (SAMOS), 2012 International Conference on*. IEEE, 2012, pp. 51–57.