Computing Routes and Delay Bounds for the Network-on-Chip of the Kalray MPPA2 Processor

Marc Boyer
ONERA
Toulouse, France

Benoît Dupont de Dinechin
Kalray
Grenoble, France

Amaury Graillat
Kalray, Verimag
Grenoble, France

Lionel Havet
RealTime-at-Work
Nancy, France

***Abstract*—The Kalray MPPA2 manycore processor implements a clustered architecture, where clusters of cores share a local memory, and communicate through a RDMA-capable network-on-chip (NoC). This NoC has been designed to allow guaranteed delays by the adequate configuration of traffic limiters at ingress and the choice of routing. We first present the challenges related to the routing of concurrent flows and a strategy that solves it. Routing challenges include deadlock-free routing, which is always an issue on wormhole switching networks, fairness of resource allocation between flows, and ensuring the feed-forward property required by deterministic network calculus (DNC). Second, we present a linear formulation based on DNC for computing end-to-end delay bounds of concurrent feed-forward flows on the MPPA2 NoC. Finally, we compare this linear formulation to a classic formulation originally designed for the AFDX avionics networks.**

1. Introduction

Network-on-Chip [1] (NoC) is the dominant paradigm for on-chip interconnects. While mostly applied to best-effort communication scenarios, NoC interconnects can also be engineered to provide service guarantees by using a Deterministic Network Calculus (DNC) approach [2]. In this paper, we present our approach to NoC management for Kalray MPPA2 processor [3], based on DNC. Assuming that application tasks are assigned endpoints on the NoC, we proceed in two steps:

1) route the flows to ensure deadlock-free operations, while fairly allocating maximum rates to the flows;
2) using DNC, compute the maximum queue backlogs and upper-bounds on the end-to-end latencies.

In order to apply DNC to the MPPA2 NoC, we need to determine the network elements traversed by each flow, which implies selecting a route between its endpoints. Moreover, application of DNC require that flows be routed over a *feed-forward* network, that is, the graph of directed links traversed by the flows has no cycles (circuits) [4].

The motivation for this work is an anticipation on how avionics certification could be applied to a MPPA manycore processor: each cluster would be considered as a shared memory multicore processor, with certification based on the EASA CRI-MCP [5] guidelines; inter-cluster communication would be carried by the NoC, where the certification principles of AFDX based on DNC analysis would apply.

Organization of the paper is as follows. Section 2 introduces the MPPA2-256 Bostan NoC architecture, discusses previous work, and summarizes basic results of Deterministic Network Calculus (DNC). In Section 3, we show that deadlock-free routing on a wormhole switching network and ensuring feed-forward flows are equivalent. In Section 4, we present a linear formulation based on DNC to compute end-to-end latencies for concurrent flows over the MPPA2 NoC. In Section 5, this linear formulation is compared to a classic formulation originally designed for the AFDX avionics networks and adapted to the MPPA2 NoC.

2. Background

2.1. The MPPA2-256 Bostan NoC

The MPPA2-256 processor [3] integrates 256 processing cores and 32 management cores on a chip, all implementing the same VLIW core architecture. The MPPA2-256 architecture is clustered with 16 compute clusters and 2 I/O clusters, where each cluster is built around a multi-banked local static memory shared by 16+1 (compute cluster) or 4+4 (I/O cluster) processing + management cores. The clusters communicate through a RDMA NoC, with one node per compute cluster and 8 nodes per I/O cluster.

The MPPA2 NoC is a direct network based on a 2D-torus topology extended with extra links connected to the otherwise unused ports of the NoC nodes on the I/O clusters (see Fig. 1). The MPPA2 NoC implements wormhole switching with source routing and without virtual channels. With wormhole switching, a packet is decomposed into flits (32-bits on the MPPA2 NoC), which travel in a pipelined fashion across the network elements, with buffering and flow control applied at the flit level. The packet follows a route determined by a bit string in the header.

The motivation for implementing wormhole switching with source routing and without virtual channels is the reduction of hardware dedicated to the network elements and interfaces. However, wormhole switching networks are sensitive to deadlocking. Fig. 2 illustrates a deadlock situation, where flow A cannot use link $R_3 \rightarrow R_2$ because flow

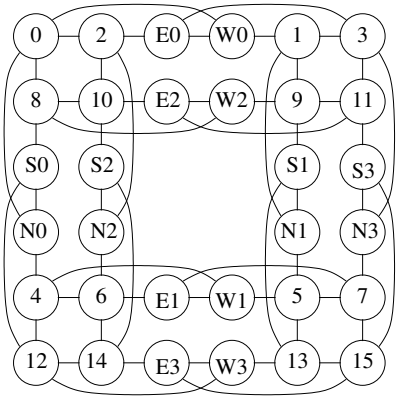

Figure 1. MPPA2 NoC topology unfolded (I/O nodes are labeled N0..N3, E0..E3, S0..S3, W0..W3).

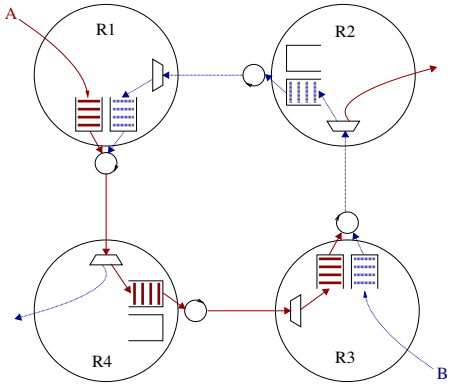

Figure 2. Deadlock with a wormhole switching NoC.

B is using it. Likewise, flow B needs $R_1 \rightarrow R_4$ held by flow A. Deadlock requires that router queues be full.

A MPPA2 NoC node is composed of a router (Fig. 3) and a cluster interface. The network elements of the MPPA2 NoC appear in Fig. 3: the links between nodes, the switches that steer packets depending on the routing information, the 'turns' inside nodes, and the link arbiters. Each outgoing link has one such arbiter, which selects whole packets by round-robin across the adjacent queues. There is one queue

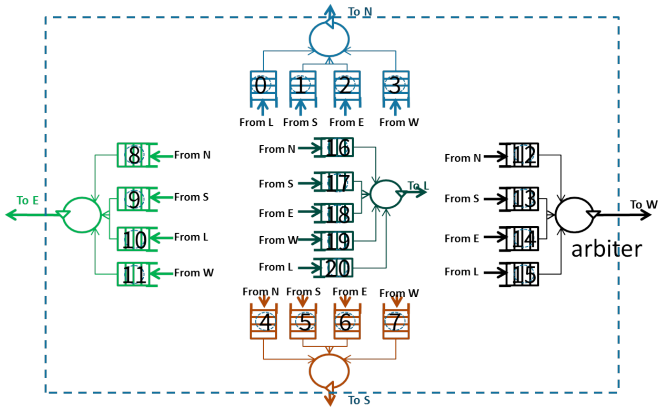

Figure 3. Structure of a MPPA2 NoC router.

per incoming direction or 'turn'.

We call a queue active if there is some flow passing through it and there is another queue in the same link arbiter with some flow. A non-active queue either has no flow passing through it, or is the only one in the link arbiter with some flow. If so, there are no effects on the packets beyond a constant delay and the queue can be ignored except for the constant delay that needs to be added to the end-to-end latency bound.

2.2. Context and Previous Work

They have been several studies designed to compute upper bounds on the worst case traversal time (WCTT) of a NoC by a data flow. Nevertheless, they mainly consider either a pure wormhole solution, with hop-level flow-control, or systems with hardware dedicated to QoS management such as packet priorization and preemption. Hop-level flow control and packet preemption are not amendable to DNC analysis. In fact, we found only a few approaches based on DNC which are applicable to the MPPA2 NoC architecture. An overview of the state of the art of NoC performance evaluation can be found in [6].

The *recursive calculus* has been designed to compute bounds on the SpaceWire technology, a wormhole-based technology [7]. It has been adapted to the MPPA NoC in [8] and compared with a DNC-based approach on an example, that will also be considered in this article. In [9], authors consider a DNC model for a NoC with wormhole switching, input queuing, and Weighted Round Robin arbitration. However, the MPPA2 NoC only has output queuing.

Both [8], [9] take into account the back-pressure mechanism associated to the wormhole switching. Back-pressure occurs when the buffer in a switch is full. In classical store and forward systems, when a buffer is full, either incoming or local packets are dropped. In wormhole switching, when a buffer is full, the upstream link stops. Then, its own buffer may fill, up to also being full. This "back-pressure" mechanism may goes back up to the source, and may also block others flows sharing the blocked links. In pathological cases, this may lead to a global network stall, cf. Fig 2.

Since the back-pressure mechanism has a unbounded impact on the NoC performances, it must not be activated in a real-time application environments. And since this mechanism is activated only when local buffers are full, one can configure the traffic shapers in a way such that NoC buffers are never fully used. From the network calculus point of view, when the back-pressure mechanism is not active, the MPPA2 NoC is simply a network using a Round-Robin arbitration and cut-through forwarding.

A network-calculus model of the Round-Robin policy has been presented in [10], [11]. It is compared in [12] with the "arbitrary multiplexing" model in the case of a NoC.

A computation of the MPPA NoC delays based on network calculus has been presented in [13]. The approach has been refined for the MPPA2 NoC in [14], whose results are summarized in Section 4. These authors study the problem

of deadlock-free feed-forward routing on the MPPA2 NoC in [15], whose main results are reported in Section 3.

Note that solving both routing and resource assignment withing the network calculus framework have also been studied in [16], [17], [18] in a different context: it considers that the flow throughput is an input of the problem and that per flow resource reservation can be done.

2.3. Deterministic Network Calculus (DNC)

Deterministic Network Calculus [4] is a framework for performance analysis of networks based on the representation of flows as cumulative data over time. Given a network element with cumulative arrival and departure functions A and A', the delay for traversing the network element and the backlog of data correspond respectively to the horizontal and the vertical deviations between A and A' (Fig. 4).

Network Calculus exploits the properties of the $(\min, +)$ algebra, in particular it introduces the following operations:

$$\text{convolution:} \quad (f \otimes g)(t) \triangleq \inf_{0 \le s \le t} f(t) + g(t-s)$$

$$\text{deconvolution:} \quad (f \oslash g)(t) \triangleq \sup_{s \ge 0} f(t+s) - g(s)$$

Let A be a cumulative data function. An *arrival curve* α is a constraint on A defined by $\forall 0 \le s \le t : A(t) - A(s) \le \alpha(t-s)$, which is equivalent to $A \le A \otimes \alpha$. Fig. 5 illustrates the smoothing constraint of the arrival curve α on the cumulative function A. This particular type of arrival curve $\alpha(t) = (\rho t + \sigma)1_{t>0}$, known as *affine* or *leaky-bucket*, is denoted $\gamma_{\rho,\sigma}$ with ρ the *rate* and σ the *burstiness*.

Let A, A' be the cumulative arrival and departure functions of a network element. This element has β as a *service curve* iff: $\forall t \ge 0 : A'(t) \ge \inf_{0 \le s \le t}(A(s) + \beta(t-s))$, which is equivalent to $A' \ge A \otimes \beta$. Fig. 6 illustrates the guarantee offered by the service curve β. This particular type of service curve $\beta(t) = R[t-T]^+$, known as *rate-latency*, is denoted $\beta_{R,T}$ with R the *rate* and T the *latency*.

Key results of Deterministic Network Calculus include:

- The arrival curve of the aggregate of two flows is the sum of their arrival curves.
- A flow A with arrival curve α that traverses a server with service curve β results in a flow A' constrained by the arrival curve $\alpha \oslash \beta$.
- The service curve of a tandem of two of servers with service curves β_1 and β_2 is $\beta_1 \otimes \beta_2$.
- If a flow has arrival curve $\alpha(t)$ on a node of service curve β, the backlog bound is $\max_{t \ge 0}(\alpha(t) - \beta(t))$ and the delay bound is $\mathit{hDev}(\alpha, \beta) = \max_{t \ge 0}\{\inf s \ge 0 : \alpha(t) \le \beta(t+s)\}$. These bounds are respectively the maximum vertical deviation and maximum horizontal deviation between α and β.

3. Max-Min Fair Feed-Forward Routing

The routing problem on the MPPA2 NoC has three objectives. First, routes must be chosen to avoid deadlock, which is always an issue on wormhole switching networks. Second, the resulting routes must compose a feed-forward network, to enable application of the main DNC results. Third, the selection of routes for each flow must optimize use of the global network capacity, while guaranteeing a fair allocation of bandwidth to the flows.

In this section, we include the main contributions of [15], with the experimental results extended to include the maximum and average of upper bounds on end-to-end delays computed according to the linear formulation of Section 4.

3.1. The Feed-Forward Routing Problem

On a wormhole switching network, the objectives of deterministic deadlock-free routing and feed-forward flows are connected:

- In a connection network, deadlock results from circuits of *agents* and *resources* connected by a *wait-for* relation [19]. With wormhole switching, agents are packets, and resources are the router internal turns and the external links between routers.
- Deadlocks can be avoided by eliminating directed cycles in the resource dependence graph. This is accomplished by imposing a partial order on the resources and then insisting that an agent allocates resources in ascending order [20].
- A definition of a feed-forward network is that it is possible to find a numbering of its links such that for any flow through the network, the numbering of its traversed links is an increasing sequence [21].

As the links considered in a feed-forward network form a subset of the resources considered for deadlock, the numbering of these resources so that they are allocated in ascending order by a packet is also a numbering of the links which are traversed in ascending order by the flows. Therefore, application of a deterministic deadlock-free routing algorithm ensures that the flows are feed-forward.

Deadlock-free routing algorithm for wormhole switching have been proposed, mostly for 2D-mesh topologies. The classic ones are X-Y or Dimension-Ordered [20] and the Turn Model [22]. The X-Y routing has no path diversity, while the Turn Model variants have irregular diversity. This problem led to the formulation of the Odd-Even [23] turn model, which has been improved as the Hamiltonian Odd-Even turn model [24] to accommodate multicasting.

Independently from the computer architecture community, the communication network community has proposed a series of techniques to ensure that flows are feed-forward. In particular, the Turn Prohibition algorithm [25] finds a set of prohibited turns that ensure the feed-forward routing property on an arbitrary network topology of bi-directional links and turns. A turn is a triple of nodes (a, b, c) connected by two links and the algorithm requires that if turn (a, b, c) is prohibited, so must be turn (c, b, a). Turn Prohibition has been improved to yield Simple Cycle Breaking [26], which still requires disabling both directions of a turn at once.

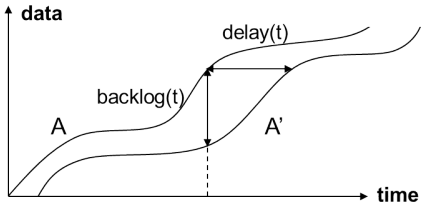

Figure 4. Flow arrival and departure as cumulative data functions over time.

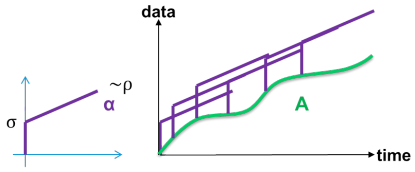

Figure 5. Arrival curve α for cumulative function A.

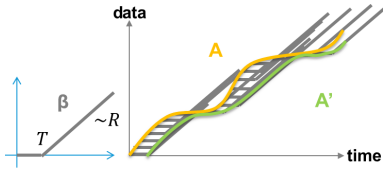

Figure 6. Service curve β for a server $A \rightarrow A'$.

In order to ease implementation of routing under Turn Prohibition, [27] introduces the Turnnet, which is a graph whose nodes correspond to the external links, and arcs to the internal turns of the network. Application of any cycle-breaking technique to ensure feed-forward flows on the network amounts to removing arcs from the Turnnet so it becomes acyclic. It is then used to compute routes, for instance by applying Dijkstra's shortest path algorithm [27].

Observe that in case of wormhole switching, the Turnnet can be expanded into the resource dependence graph by splitting each arc and inserting a node corresponding to the turn queue. As this expansion cannot introduce a (directed) cycle, an acyclic Turnnet implies an acyclic resource dependence graph for wormhole switching. So ensuring feed-forward flows on such networks implies deadlock-free routing when using the corresponding paths.

3.2. Routing with Max-Min Fairness

Among the possibly multiple paths proposed by a routing algorithm between each pair of endpoints of a flow, only one can be selected. This constraint of routing a flow under a unique path between its endpoints comes from the MPPA2 NoC architecture, as both packet ordering inside a flow and flow regulation of (rate, burstiness) at ingress can only be ensured along a single path. Global allocation of bandwidth to flows on a network is a fairness problem, and we use max-min fairness [28] whose objective is to maximize the lowest flow rate, then the next lowest flow rate, etc.

Precisely, a rate allocation is max-min fair iff an increase of any rate must be at the cost of a decrease of some already smaller rate. Max-min fair allocation is solved by the simple "Water Filling" algorithm [29] in case there is a single path available per flow. In case of multiple paths available per flow and splittable flows, the Max-Min Fair with Splittable Paths (MMFSP) problem is still of polynomial time complexity and can be solved as a series of linear programs [30]. When only a single path among those available can be assigned to the flow, the resulting Max-Min Fairness with Unsplittable Paths (MMFUP) problem is NP-hard [29].

Our first approach to routing is to solve the MMFUP problem instances by enumeration. For a given feed-forward routing algorithm, the search tree is deployed by assigning a unique path to each flow, selected among the minimal path diversity obtained by enumerating all the allowed shortest paths between each flow endpoints. On the leaves of this search tree, the problem is solved by applying the Water Filling algorithm of [28]. This gives a vector of flow rates, including the minimum rate. Water Filling is skipped if the problem instance has a link shared by n flows and its capacity divided by n is lower than a previously computed minimum rate. Finally, a solution whose sorted vector of flow rates is lexicographically maximal is selected as in [28].

Our second approach to routing is a heuristic that reduces the search space by not enumerating the paths that carry a split flow of low rate. Each internal node of the search tree corresponds to a MMFSP problem instance, which is solved by a series of linear programs. At a search tree node, iterate over three steps: select a split flow of lowest rate; remove all its path(s) not carrying the maximum amount of sub-flow; solve the resulting MMFSP problem instance. Following this iteration, either all flows follow a unique path, or there exists a flow of minimal rate equally split among its alternate paths. Recursively apply the heuristic on the new problem instances where each of the alternate paths of this flow has become its only allowed path.

3.3. Experimental Comparisons

We compare the routing approaches on two representatives of each family of algorithms: X-Y (XY) and Hamiltonian Odd-Even [24] (HOE) for deadlock-free routing; Turn Prohibition [25] (TP) and Simple Cycle Breaking [26] (SCB) for feed-forward flows. The routing problems include the example in [14], an AFDX example from [8], and also the classic Bit-Complement, Bit-Reverse, Shuffle and Tornado instances. Only the MPPA2 compute clusters, re-numbered as a 4×4 grid, are considered as endpoints (as illustrated in Fig. 8 for Bit-Complement). Except for XY, all routing algorithms allow alternate minimal paths, whose diversity is displayed in Fig. 7. Overall, HOE and SCB provide the highest minimal path diversity.

The results of max-min fair allocation of flow rates appear in Table 3, where we compare the enumeration approach based on Water Filling to the heuristic approach based on MMFSP resolutions. Columns *#L* display the number of leaves in the respective search trees, however in case of enumeration the exploration is limited to 50K steps in order to limit computation times. The main criterion for comparison is the minimum of flow rates ρ_{min}, although the average of flow rates ρ_{avg} gives an indication on how the global NoC bandwidth is exploited. The maximum and

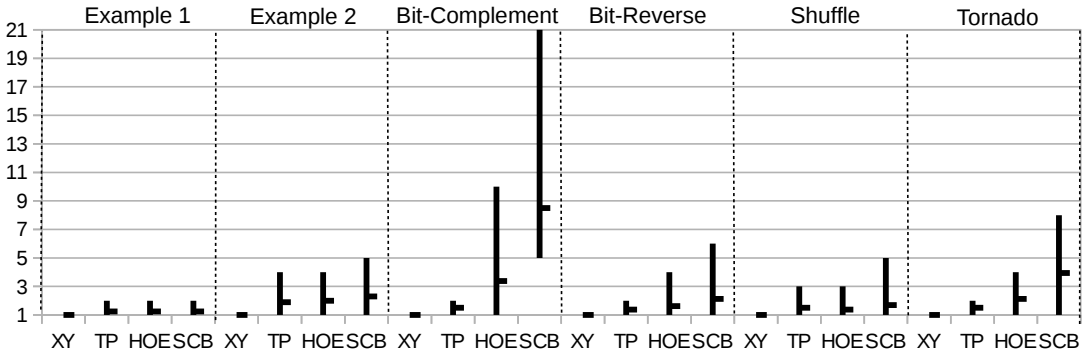

Figure 7. Minimal path diversity in path count per flow for XY, TP, HOE and SCB (horizontal line is average diversity).

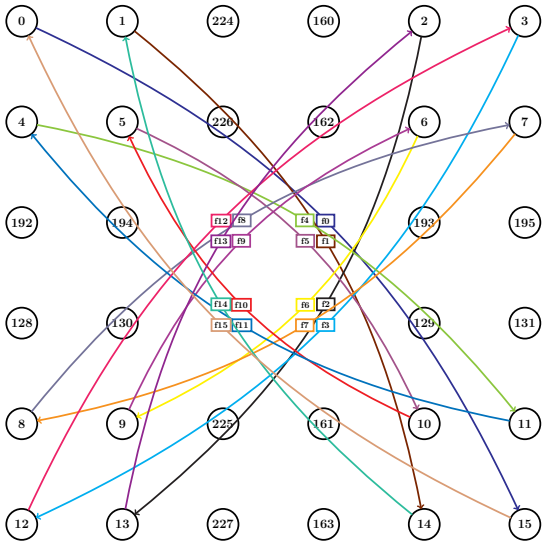

Figure 8. Bit-complement routing problem instance on the 4×4 2D-mesh subset of the MPPA2 NoC topology.

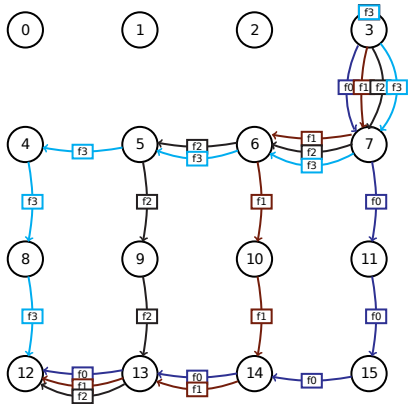

Figure 9. HOE path diversity between nodes 3 and 12 on the 4×4 2D-mesh subset of the MPPA2 NoC topology.

average end-to-end latencies $d^*_{\max}$ and d^*_{avg} are computed using the linear formulation of Section 4 for 17-flit packets.

We observe in Table 3 that the heuristic drastically reduces the number of leaves (#L) explored, while providing good solutions. In case of BC-SCB, the heuristic outperforms the enumeration because the latter suffers from the 50K steps limitation. The deadlock-free routing techniques (XY, HOE), although restricted to the 2D-mesh topology subset, appear to perform better than the feed-forward techniques (TP, SCB). Surprisingly, XY routing that has no minimal path diversity outperforms HOE on Bit-Complement and Tornado. The explanation is that HOE does not allow some XY path, as illustrated in Fig. 9.

4. Linear Formulation for the MPPA2 NoC

The MPPA2 NoC is a RDMA-capable network that can be configured for regulating the injection rate ρ_i and burstiness σ_i at ingress of each flow f_i, that is, setting the parameters of a leaky-bucket arrival curve. In this section, we summarize the linear formulation of the DNC equations for the MPPA2 NoC that has been developed in [14], [15].

The motivation of this formulation is to provide upper bounds on queue backlogs and flow end-to-end delays, while accounting for the shaping implied by the peak rate r of one flit per cycle in the network elements. Replacing the flow rates by the values computed when solving the Feed-Forward Routing problem (Section 3) yields a system of linear inequalities with the flow burstiness as variables. Thanks to the feed-forward flows, the system of linear inequalities is acyclic and solved in one pass.

4.1. Effects of Link Shaping

In this linear formulation, each flow f_i is associated with a series of arrival curves $\gamma_{\rho_i,\sigma_i^j}$, where ρ_i is its constant rate and σ_i^j its burstiness in front of queue q^j. Queue q^j receives the aggregates of flows F^j passing through it, so its arrival curve is of leaky-bucket type γ_{ρ^j,σ^j} with $\rho^j = \sum_{f_i \in F^j} \rho_i$ and $\sigma^j = \sum_{f_i \in F^j} \sigma_i^j$, shaped by the turn peak rate r. This yields the arrival curve $\min(rt, \sigma^j + \rho^j t)1_{t>0}$, which is a special case of the standard T-SPEC arrival curve $\alpha(t) = \min(M + pt, rt + b)1_{t>0}$ used in IntServ [31].

Assume that a link arbiter offers a rate-latency service curve β_{R^j,T^j} to queue q^j with $R^j \leq r$. The delay d^j for queue q^j is the maximum horizontal deviation between the

arrival curve and the service curve. Application of the T-SPEC arrival curve on such service curve yields [4]:

$$d^j = T^j + \frac{\sigma^j(r - R^j)}{R^j(r - \rho^j)} \quad (1)$$

Let $l_i^{\max}$ be the maximum packet size for flow f_i. At ingress, whole packets are atomically injected at rate r. Call θ the date when injection ends. We have $r\theta = l_i^{\max}$ and $l_i^{\max} \leq \sigma_i + \rho_i\theta$, so:

$$\forall f_i \in F : \sigma_i \geq \sigma_i^{\min} \triangleq l_i^{\max}\frac{r - \rho_i}{r} \quad (2)$$

We now express the values ρ_i^j and σ_i^j for all flows $f_i \in F^j$ for an active queue q^j. If q^j is the first active queue traversed by the flow, then $\sigma_i^j = \sigma_i$. Else, let q^k be predecessor of q^j in the sequence of active queues traversed by flow f_i, with β_{R^k,T^k} its service curve. When flow f_i traverses queue q^k, its burstiness increases differently whether it is alone or aggregated with other flows in q^k.

If the flow is alone in queue q^k, we apply the classic result of the effects of a rate-latency service curve $\beta_{R,T}$ on a flow constrained by an affine arrival curve $\gamma_{\rho,\sigma}$. The result is another affine arrival curve $\gamma_{\rho,\sigma+\rho T}$ [4], so:

$$\sigma_i^j = \sigma_i^k + \rho_i T^k \quad (3)$$

Else, we apply **Theorem 6.2.2 (Burstiness Increase Due to FIFO Multiplexing, General Case)** [4], where flow 1 is token bucket constrained with rate ρ_1 and burstiness σ_1, and flow 2 is constrained by a sub-additive arrival curve α_2. Because of link shaping in q^k, $\alpha_2(t) = \min(rt, \rho_2 t + \sigma_2)1_{t>0}$. As a result, $b_1 = \sigma_1 + \rho_1(T + \frac{\sigma_2(r+\rho_1-R)}{R(r-\rho_2)})$ [14]. With $\rho_1 = \rho_i$, $\sigma_1 = \sigma_i^k$, $b_1 = \sigma_i^j$, $\rho_2 = \sum_{l\in F^k, l\neq i}\rho_l$, and $\sigma_2 = \sum_{l\in F^k, l\neq i}\sigma_l^k$, this yields:

$$\sigma_i^j = \sigma_i^k + \rho_i\left(T^k + \frac{(\sum_{l\in F^k, l\neq i}\sigma_l^k)(r + \rho_i - R^k)}{R^k(r - \sum_{l\in F^k, l\neq i}\rho_l)}\right) \quad (4)$$

Not accounting for the link shaping would yield $b_1 = \sigma_1 + \rho_1(T + \frac{\sigma_2}{R})$, by application of **Corollary 6.2.3 (Burstiness Increase due to FIFO)** [4]. This result is worse than Eq. (4) because $\rho_1 + \rho_2 < R \Rightarrow r + \rho_1 - R < r - \rho_2$.

4.2. Link Arbiter Service Curves

On the MPPA2 NoC, the output link arbiters operate in round-robin on turn queues at the packet granularity, while each queue contains flows aggregated in FIFO. As the packets presented to a link arbiter are not processed in FIFO order, previous work (e.g. [11]) would have to assume blind multiplexing between all flows and fail to exploit FIFO aggregation. This is addressed in [14] by exposing the service offered to each queue of a link arbiter: either, the rate and latency ensured by round-robin packet scheduling; or, the residual service guaranteed by blind multiplexing across queues when the round-robin service does not apply. Then, aggregation need only be considered withing the scope of single queues so is FIFO.

The service curve offered by a link arbiter to each of its queues is abstracted as a rate-latency function $\beta^j = \beta_{R^j,T^j}$. The first approach to derive this curve is to consider the behavior of the round-robin arbiter, assuming that each flow f_i has its packet sizes bounded by a minimum $l_i^{\min}$ and a maximum $l_i^{\max}$. Let $l_{F^j}^{\min} \triangleq \min_{f_i\in F^j} l_i^{\min}$ and $l_{F^j}^{\max} \triangleq \max_{f_i\in F^j} l_i^{\max}$ be respectively the minimum and maximum packet sizes for q^j. Let A^j be the set active queues in the link arbiter of q^j, and $B^j \triangleq A^j - \{q^j\}$. The general round-robin service curve $\beta^j = \beta_{R^j,T^j}$ for q^j is:

$$R^j = \frac{r l_{F^j}^{\min}}{l_{F^j}^{\min} + \sum_{k\in B^j} l_{F^k}^{\max}} \text{ and } T^j = \frac{\sum_{k\in B^j} l_{F^k}^{\max}}{r} \quad (5)$$

The second approach to derive a service curve for queue q^j is to consider that the round-robin arbiter serves packets at peak rate r according to a blind multiplexing strategy across the queues. Application of **Theorem 6.2.1 (Blind Multiplexing)** [4] yields the blind multiplexing service curve $\beta^j = \beta_{R^j,T^j}$ for q^j:

$$R^j = r - \sum_{k\in B^j}\rho^k \text{ and } T^j = \frac{\sum_{k\in B^j}\sigma^k}{r - \sum_{k\in B^j}\rho^k} \quad (6)$$

The blind multiplexing service curve must be used whenever the sum of flow rates inside q^j exceeds R^j in Eq. (5). Else, we select the formula that evaluates to the lowest T^j.

4.3. End-to-End Latency Bound

For computing an upper bound on the end-to-end latency of any particular flow f_i, we proceed in three steps. First, compute the *left-over* (or residual) service curve β_i^j of each active queue q^j traversed by f_i. Second, find the equivalent service curve β_i^* offered by the NoC to flow f_i through the convolution of the left-over service curves β_i^j. Last, find the end-to-end latency bound by computing d_i^* the delay between α_i the arrival curve of flow f_i and β_i^*. Adding d_i^* to the constant delays of flow f_i such as the traversal of non-active queues and other logic and wiring pipeline yields the upper bound. This approach is similar in principle to the Separated Flow Analysis (SFA) [11], even though the latter is formulated in the setting of aggregation under blind multiplexing, while we use FIFO multiplexing.

For the first step, we have two cases to consider at each active queue q^j. Either f_i is the only flow traversing q^j, and $\beta_i^j = \beta_{R^j,T^j}$ from equations (5) or (6). Or, f_i is aggregated in q^j with other flows in F^j. Packets from the flow aggregate F^j are served in FIFO order, so we may apply **Corollary 6.2.3 (Burstiness Increase due to FIFO)** [4]. This yields the left-over service curve $\beta_i^j = \beta_{R_i^j,T_i^j}$ for an active queue q^j traversed by f_i:

$$R_i^j = R^j - \sum_{l\in F^j, l\neq i}\rho_l \text{ and } T_i^j = T^j + \frac{\sum_{l\in F^j, l\neq i}\sigma_l^j}{R^j} \quad (7)$$

For the second step, we compute the convolution β_i^* of the left-over service curves β_i^j. Let Q_i denote the set of

active queues traversed by flow f_i. Thanks to the properties of rate-latency curves [4], β_i^* is a rate-latency curve whose rate R_i^* is the minimum of the rates and the latency T_i^* is the sum of the latencies of the left-over service curves β_i^j:

$$R_i^* = \min_{j \in Q_i} R_i^j \text{ and } T_i^* = \sum_{j \in Q_i} T_i^j \tag{8}$$

For the last step, we compute the delay d_i^* between α_i the arrival curve of flow f_i at ingress and β_i^*. This flow is injected at rate ρ_i and burstiness σ_i, however it is subjected to link shaping at rate r as it enters the network. As a result, $\alpha_i = \min(rt, \sigma_i + \rho_i t)1_{t>0}$ and we may apply Eq. (1):

$$d_i^* = T_i^* + \frac{\sigma_i(r - R_i^*)}{R_i^*(r - \rho_i)} \tag{9}$$

5. Comparing DNC Formulations

In this section, we compare the linear DNC formulation of Section 4 to a "local" DNC formulation, which is an adaptation to the MPPA2 NoC of an algorithms designed to analyze AFDX networks [32], [33], [34]. In these comparisons, we assume that the flow paths are pre-determined.

5.1. Local Formulation

An AFDX switch applies a 2-levels non-preemptive static priority scheduling, meaning that in each output port, there are two queues. All flows from the same priority level are set in the same queue. The queue with the low priority is selected only if the queue with the high priority is empty. All flows sharing a queue are served with a FIFO policy.

The network calculus analysis consists in: 1) computing a residual service, β_q for each queue q; 2) computing the global arrival curve in this queue α_q by summing the arrival curves of all individual flows, and considering the shaping; and 3) computing $d_q = hDev(\alpha_q, \beta_q)$ a bound on the delay of the queue, which is also a bound on delay for all flows in the queue due to FIFO aggregation.

The delay bound of a flow f is then computed as the sum of the queue delay bound for all crossed servers. This approach is quite different from the linear formulation presented in Section 4, in several points.

First, it relies on a class more generic than linear curves, which is more accurate in general. For example, consider a periodic (resp. sporadic) flow sending one packet of maximal size L every P time unit (resp. not quicker than every P time unit). Two arrival curves can be computed for this kind of flow: a linear one, $\alpha^l(t) = L + \frac{L}{P}t$, or an exact one, $\alpha^e(t) = \lceil \frac{tL}{P} \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function (cf. Fig. 10). The AFDX tool, presented in [34] can handle the *Ultimately Pseudo Periodic* (UPP) class of functions, that generalizes both kinds of curves.

Second, it does no perform the end-to-end SFA analysis but computes per queue local delay bound.

Third, it does not apply **Corollary 6.2.3 (Burstiness Increase due to FIFO)** [4] since it can be applied only in the case of linear arrival curves.

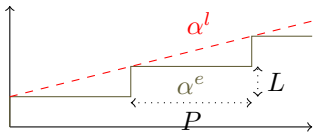

Figure 10. Two arrival curves for a periodic/sporadic flow of period P and packet length L

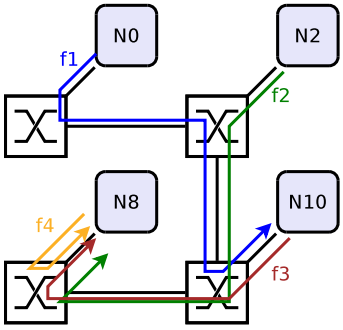

Figure 11. Case study from [14]

Last, AFDX switches use static priorities to share the bandwidth between queues connected to an output port. For the comparisons, the AFDX tool has been adapted to consider round-robin output arbitration policy, using Eq. (5), like the linear formulation.

5.2. First Example

The first example comes from [14]. Four flows are considered, $f_1, \ldots, f_4$, with a maximum packet size of 17 flits. All flows have a long-term rate $\frac{1}{3}$ but f_1 that have $r_1 = \frac{2}{3}$. The admissible bursts at network ingress are $\frac{34}{3}$ but f_1 that have $b_1 = \frac{17}{3}$.

The upper bounds on delays for this example, computed with the linear and the classic formulation, are reported in Table 1. The results are comparable, but the linear formulation gives better (smaller) bounds. This is due to several differences between an AFDX network and the MPPA2 NoC, and also to the fact that the linear formulation fits very well the MPPA2 NoC architecture.

The flow contract is the first difference: in AFDX, a flow (called *Virtual Link*) is characterized as a sporadic flow, with a maximal frame size and an inter-frame minimal time. As shown in Fig. 10, a linear model cannot capture exactly this behavior, it only does an upper approximation. In the MPPA2 NoC, traffic shaping is done at ingress by a token-bucket regulator, which enforces a linear constraint. Whereas it had been shown in [34] than for typical AFDX networks, the linear formulation is 18% pessimistic with respect to a more accurate one, it is sufficient for the MPPA2 NoC.

A second difference is related to the queuing architecture and its impact on bursts. In an AFDX switch, the flows from

	f_1	f_2	f_3	f_4
Linear formulation	25	110	100	34
Local formulation	25	170	136	34

TABLE 1. BOUNDS ON NOC DELAY PER FLOW IN CYCLES.

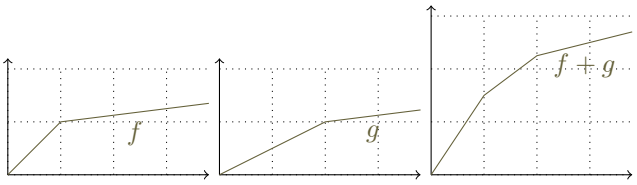

Figure 12. Sum of two-sloped functions

two different input ports can be merged into the same output port. In a MPPA2 NoC router, for each output port, there is a queue dedicated to each input port (cf. Fig. 3). In an AFDX switch, each queue can receive traffic from several input links, whereas in the MPPA2 NoC, it received only from one input link. This changes the kind of curve required to accurately model a switch. As shown in Fig. 12, the sum of two piece-wise linear functions, each one having two slopes, leads to a piece-wise linear function with three slopes. This implies that a linear model with two slopes is sufficient to capture the MPPA2 NoC behavior, whereas a more complex one is required for AFDX.

A third difference stems is the use of **Corollary 6.2.3** [4] by the linear formulation for the left-over service in Eq. (7). Considering a set of flows $(f_i)_{i\in[1,n]}$ sharing a FIFO queue, each flow having a leaky-bucket arrival curve γ_{r_i,b_i}, the increase of the burst of the flow f_i does not depend on its own burst size, but only on the sum of the others ones. In AFDX networks, since a queue is in general shared by a lot of flows, removing its own burst size has a limited positive impact, whereas approximating a sporadic flow as a leaky-bucket one has a large negative impact. This explains why the the local formulation originally designed for AFDX does not implement this result.

These differences explain the numerical results observed in this example. Consider the flow f_2: its crosses three routers. In the first router, R_2, it shares the output link with the flow f_1, with a Round-Robin policy, each flow being alone in its own queue. The situation is the same in the second crossed router, R_{10}, with the flow f_3. In the last router, R_8, the flows f_2 and f_3 share a common queue, with a FIFO aggregation policy, and both share the output link with the flow f_4 with a Round Robin policy.

In the linear formulation, the Round-Robin link arbitration is used to compute the residual service in routers R_2 and R_{10}, using Eq. (5), leading in both cases to a latency $T^2 = T^{10} = 17$, since the flow have to wait at most one packet of the competing flow before getting access to the output link, and a rate $R^2 = R^{10} = \frac{1}{2}$, since all flows having the same packet sizes, the round-robin arbiter shares the throughput in two equal parts In the last router, R_8, the flows f_2 and f_3 share a common queue. First, one has to select the residual service offered by the link arbiter. Since the total load of both f_2 and f_3 is $\frac{2}{3}$, it overtakes the service offered by the Round-Robin arbiter, so the Blind Multiplexing result of Eq. (6) is used. It leads to a latency $T^8 = 17$, the latency associated to one packet, and a rate $R^8 = \frac{2}{3}$ since the other flow uses only $\frac{1}{3}$ of the throughput. But this service is shared by f_2 and f_3. The left-over service dedicated to flow f_2 is computed using Eq. (7), leading to $R_2^8 = \frac{2}{3} - \rho_3 = \frac{1}{3}$ and $T_2^8 = T^8 + \frac{\sigma_1^8}{R^8} = 17 + \frac{\sigma_1^8}{2/3} = 17 + \frac{17}{2/3} = 42.5$.

The interpretation is that, first, the residual throughput is $\frac{1}{3}$, since f_2 shares the output link with two flows, each of throughput $\frac{1}{3}$, second, the latency is decomposed into two terms, the 17 is the frame size of the flow f_4, in the other queue, and the $\frac{17}{2/3}$ is the latency due to the sharing of the queue with flow f_4, which has a frame size of 17, and this shared queue has a residual service of $\frac{2}{3}$. Once the per router residual services have been computed, one can compute the end-to-end delay, using Pay Burst Only Once principle (PBOO), with Eq.(8). The latency term is $T_2^* = T_2^2 + T_2^{10} + T_2^8 = 17 + 17 + 42.5 = 76,5$. The minimal rate is the one observed in the last router, $R_2^8 = \frac{1}{3}$. The application of Eq. (9) leads to 110.

In the local formulation, a per queue delay is computed. In the first and the second routers, the Round-Robin residual service is, like for the linear formulation, a service of rate $\frac{1}{2}$ and latency 17. Eq. (9) can be applied to compute the local delay, leading to a delay of 34: 17 cycles due to the Round-Robin arbitration latency, and 17 cycles related to the service of a packet. In the last router, one can also apply the Blind Multiplexing result of Eq. (6), leading also to a latency of 17 cycles and a throughput of $\frac{2}{3}$ for the queue that is shared between f_2 and f_3. The delay associated to this queue is 102 cycles: 17 related to the latency and 85 related to the service of the burst at rate $\frac{2}{3}$.

We can now compare the approaches. Both linear and local formulations consider a latency of 17 cycles per router, since at each router, one packet of the other queue can be served before having access to the output link. But the local formulation also has a latency related to the handling of the burst in each queue, whereas the linear formulation pays burst only once by using the SFA principles. Moreover, the use of **Corollary 6.2.3** [4] for the left-over service in Eq. (7) allows for a better handling of the FIFO aggregation.

5.3. Second Example

The second example comes from [8]. It considers 6 flows, $f_1, \ldots, f_6$, each one being strictly periodic with a packet size of 50 flits an a period of 1000 cycles. This yields a rate of 5% on each flow. Unlike with the previous example, the network capacity is not fully exploited.

In [8], the authors compare on this example the recursive calculus (RC) and a network calculus model given in [13], presented in here as "Naive" since it was a first DNC model for the NoC of the first-generation MPPA processor. The values of the NoC delay bounds are reported in Table 2 augmented with the results of the local and linear formulations. Note that the results presented here are not exactly the same as in [8]. There, the delay includes the arbitration in the node, whereas the methods presented here only consider the NoC delays. Then, the node arbitration delays have been removed for flows f_1, f_2 and f_3.

A first observation is that the linear and the local formulations both outperform recursive calculus and an early

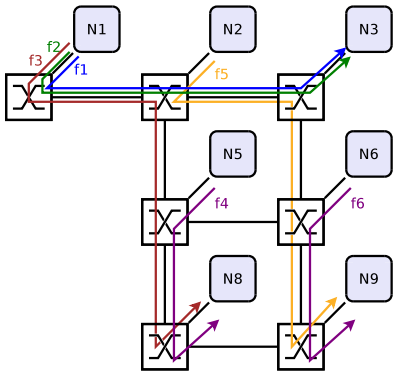

Figure 13. Case study from [8]

	f_1	f_2	f_3	f_4	f_5	f_6
NC, Naive form. [13]	4104	4104	5155	3153	5255	3153
Recursive Calculus [8]	158	158	107	103	156	103
NC, Local form.	57	57	54	62	153	58
NC, Linear form.	105	105	52	52	150	100

TABLE 2. BOUNDS ON NOC DELAY PER FLOW, IN CYCLES.

DNC formulation for the MPPA [13]. Whereas in the previous experiment, the linear formulation always gave smaller bounds, in this experiment, the local formulation is tighter for half or the flows (f_1, f_2, f_6).

Consider the flow f_1. Its queuing delays in the first and last routers crossed (R_1 and R_3) are null, since the output throughput is always less than the input throughput. This is handled by both the local and the linear formulation. The difference come from the handling of the second router. In this router, f_1 shares with f_2 a queue, (of index q) with a FIFO policy, and both share the output link with f_5 with a Round-Robin policy. Both models the residual service offered to the shared queue using Blind Multiplexing, leading to a rate-latency service with latency $T^q = 50$ (one frame from competing flow f_5) and a rate $R^q = 0.95$ (since the rate of f_5 is 0.05). Then comes a difference.

The linear formulation computes a residual service for the flow f_1, using Eq. (7), leading to $R_1^q = 0.95$ and $T_1^q = 100 = 50 + 50$, since the flow f_2 may have to wait one frame from the flow f_5 (of size 50) and one frame from the flow f_2 (also of size 50). The local formulation considers that f_1 is part of the aggregate flow $f_1 + f_2$. This flow has a burst size of 100 flits (two frames), a long rate $\rho_{1,2} = 0.1$ and is shaped by the incoming link at a rate $r = 1$. It is served at rate $R^q = 0.95$, after latency $T^q = 50$, leading to a local delay of 56, by local application of Eq. (9).

The *interpretation* of this result is that the local formulation better captures the "serialization" between f_1 and f_2 coming from the same input link than the linear formulation.

6. Conclusions

This paper addresses three related problems of the MPPA2 NoC exploitation. The first is the routing between given endpoints, the second the NoC resource allocation and the third is the end-to-end latency bound computation.

	Enumeration					Heuristic		
	ρ_{min}	ρ_{avg}	d^*_{max}	d^*_{avg}	$\#L$	ρ_{min}	ρ_{avg}	$\#L$
E1-XY	0.333	0.500	68	51.3	1	0.333	0.500	1
E1-HOE	0.333	0.500	94	55.5	9	0.333	0.500	4
E1-SCB	0.333	0.417	111	68.0	1	0.333	0.417	1
E1-TP	0.333	0.417	111	68.0	64	0.333	0.417	8
E2-XY	0.333	0.500	128	56.2	1	0.333	0.500	1
E2-HOE	0.333	0.500	77	46.0	192	0.333	**0.383**	13
E2-SCB	0.333	0.500	85	51.1	1029	0.333	**0.433**	22
E2-TP	0.250	0.425	229	89.3	8	0.250	0.425	5
BC-XY	0.500	0.500	51	51.0	1	0.500	0.500	1
BC-HOE	0.333	0.427	145	77.6	3.69E06	0.333	**0.417**	20
BC-SCB	**0.143**	**0.143**	681	282.3	9.41E13	0.200	0.292	11
BC-TP	0.125	0.125	479	315.7	256	0.125	0.125	9
BR-XY	0.333	0.563	94	42.8	1	0.333	0.562	1
BR-HOE	0.500	0.750	34	17.5	256	0.500	**0.656**	9
BR-SCB	0.500	0.813	51	19.8	900	**0.333**	**0.729**	12
BR-TP	0.250	0.500	255	105.4	64	0.250	0.500	8
S-XY	0.500	0.750	34	17.5	1	0.500	0.750	1
S-HOE	0.500	0.875	34	9.3	36	0.500	**0.844**	7
S-SCB	0.500	0.844	51	12.4	200	0.500	**0.750**	11
S-TP	0.250	0.500	272	110.5	64	0.250	**0.469**	7
T-XY	0.500	0.500	51	51.0	1	0.500	0.500	1
T-HOE	0.333	0.438	128	76.8	1.19E07	**0.250**	**0.391**	4
T-SCB	0.500	0.594	68	35.3	5.54E06	**0.250**	**0.437**	15
T-TP	0.125	0.125	479	315.7	256	0.125	0.125	10

TABLE 3. COMPARISON OF MAX-MIN FAIRNESS ROUTING METHODS. UNDER-PERFORMING SOLUTIONS IN A ROW APPEAR IN BOLD.

For routing, we show that both deterministic deadlock-free wormhole switching routing algorithms from the computer architecture community and feed-forward routing techniques from the communication networks community can be used. On a wormhole switching network, either ensures deadlock-freedom and feed-forward flows, so that deterministic network calculus (DNC) can be applied.

For network resource allocation, we rely on the max-min fairness objective in order to select a single path for each flow among the minimal path diversity allowed by routing. As exhaustive enumeration of minimal path combinations becomes infeasible, we design an LP-based heuristic to reduce the number of combinations to explore. We observe that XY and HOE deterministic deadlock-free routing algorithms perform best in term of higher minimal rates. XY, in spite of having no path diversity, sometimes outperforms HOE, as some XY paths are not included in HOE.

The evaluation of the NoC latency has been done using two DNC-based methods. The first one, called *linear formulation*, is designed specifically for the MPPA2 NoC architecture. The second one, called *local formulation*, is adapted from a classic AFDX analysis. Both are compared on two cases and we observe that, depending on the configuration, one or the other gives the best (*i.e.* smaller) bounds. Some insight is given on the parameters that influence the performance of each method: number of flows sharing a FIFO queue, path length, etc. Nevertheless, both methods achieve comparable results.

From a research point of view, these different results provide the seeds of a cross-fertilization. This comparison has allowed us to highlight the weak points of each method, and to identify some progress perspective. From a production perspective, the lightweight computations involved by

XY routing and solving the linear formulations open opportunities for on-line reconfiguration of the MPPA2 NoC.

Acknowledgment

This work was supported by the French DGE and Bpifrance through the "Investissements d'Avenir" program CAPACITES.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip inteconnection networks," in *Proceedings of the 38th Annual Design Automation Conference*. New York,USA: ACM, 2001.

[2] Z. Lu, M. Millberg, A. Jantsch, A. C. Bruce, P. van der Wolf, and T. Henriksson, "Flow regulation for on-chip communication," in *Design, Automation and Test in Europe, DATE 2009, Nice, France, April 20-24, 2009*, 2009, pp. 578–581.

[3] S. Saidi, R. Ernst, S. Uhrig, H. Theiling, and B. D. de Dinechin, "The shift to multicores in real-time and safety-critical systems," in *2015 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2015, Amsterdam, Netherlands, October 4-9, 2015*, 2015, pp. 220–229.

[4] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Heidelberg: Springer-Verlag, 2012.

[5] EASA, "The use of multi-core processors in safety-critical applications," EASA, Tech. Rep., 02 2016.

[6] A. E. Kiasari, A. Jantsch, and Z. Lu, "Mathematical formalisms for performance evaluation of networks-on-chip," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 38, 2013.

[7] T. Ferrandiz, F. Frances, and C. Fraboul, "Worst-case end-to-end delays evaluation for spacewire networks," *Discrete Event Dynamic Systems*, vol. 21, no. 3, pp. 339–357, 2011.

[8] H. Ayed, J. Ermont, J.-l. Scharbarg, and C. Fraboul, "Towards a unified approach for worst-case analysis of tilera-like and kalray-like noc architectures," in *Proc. of the 12th IEEE World Conf. on Factory Communication Systems (WFCS 2016), WiP Session*. Aveiro, Portugal: IEEE, 2016.

[9] Y. Qian, Z. Lu, and W. Dou, "Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip," in *Proc. of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NoCS 2009)*. IEEE, 2009, pp. 44–53.

[10] J.-P. Georges, T. Divoux, and E. Rondeau, "Network calculus: application to switched real-time networking," in *Proc. of the 5th Int. ICST Conf. on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS '11, 2011, pp. 399–407.

[11] A. Bouillard and G. Stea, "Worst-Case Analysis of Tandem Queueing Systems Using Network Calculus," in *Quantitative Assessments of Distributed Systems*, Bruneo and Distefano, Eds., 2015.

[12] Y. Long, Z. Lu, and X. Yan, "Analysis and evaluation of per-flow delay bound for multiplexing models," in *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 2014, pp. 1–4.

[13] B. Dupont de Dinechin, D. Van Amstel, M. Poulhiès, and G. Lager, "Time-critical computing on a single-chip massively parallel processor," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 2014, pp. 1–6.

[14] B. Dupont de Dinechin and A. Graillat, "Network-on-chip service guarantees on the kalray mppa-256 bostan processor," in *Proc. of the 2nd Inter. Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems – AISTECS '17*, 2017.

[15] ——, "Feed-forward routing for the wormhole switching network-on-chip of the kalray mppa2 processor," in *Proceedings of the 10th International Workshop on Network on Chip Architectures*, ser. NoCArc'17. New York, NY, USA: ACM, 2017, pp. 10:1–10:6.

[16] A. Frangioni, L. Galli, and G. Stea, "Optimal joint path computation and rate allocation for real-time traffic," *The Computer Journal*, vol. 58, no. 6, pp. 1416–1430, 2014.

[17] ——, "Delay-constrained routing problems: Accurate scheduling models and admission control," *Computers & Operations Research*, vol. 81, no. Supplement C, pp. 67 – 77, 2017.

[18] ——, "Qos routing with worst-case delay constraints: Models, algorithms and performance analysis," *Computer Communications*, vol. 103, no. Supplement C, pp. 104 – 115, 2017.

[19] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, May 1987.

[20] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[21] J. B. Schmitt and F. A. Zdarsky, "The DISCO network calculator: a toolbox for worst case analysis," in *Proc. of the 1st Int. Conf. on Performance Evaluation Methodologies and Tools, VALUETOOLS 2006, Pisa, Italy, October 11-13*, 2006.

[22] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proceedings of the 19th Annual International Symposium on Computer Architecture*, ser. ISCA '92, 1992, pp. 278–287.

[23] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 729–738, Jul. 2000.

[24] P. Bahrebar and D. Stroobandt, "The Hamiltonian-based Odd-even Turn Model for Maximally Adaptive Routing in 2D Mesh Networks-on-chip," *Comput. Electr. Eng.*, vol. 45, no. C, pp. 386–401, Jul. 2015.

[25] D. Starobinski, M. G. Karpovsky, and L. Zakrevski, "Application of network calculus to general topologies using turn-prohibition," in *Proc. IEEE INFOCOM 2002, USA, June*, 2002, pp. 1151–1159.

[26] L. B. Levitin, M. G. Karpovsky, and M. Mustafa, "Minimal sets of turns for breaking cycles in graphs modeling networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 9, pp. 1342–1353, 2010.

[27] M. Fidler and G. Einhoff, "Routing in turn-prohibition based feed-forward networks," in *NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication, Third Int. IFIP-TC6 Networking Conference, Athens, Greece, May 9-14*, 2004.

[28] S. Chen and K. Nahrstedt, "Maxmin Fair Routing in Connection-Oriented Networks," in *Proc. of Euro-Parallel and Distributed Systems Conference (Euro-PDS '98)*, 1998, pp. 163–168.

[29] E. Amaldi, S. Coniglio, L. G. Gianoli, and C. U. Ileri, "On single-path network routing subject to max-min fair flow allocation," *Electronic Notes in Discrete Mathematics*, vol. 41, pp. 543–550, 2013.

[30] D. Nace, "A linear programming based approach for computing optimal fair splittable routing," in *Proc. of the Seventh Int. Symposium on Computers and Communications (ISCC'02)*, 2002.

[31] V. Firoiu, J. Y. L. Boudec, D. Towsley, and Z.-L. Zhang, "Theories and models for internet quality of service," *Proc.of the IEEE*, vol. 90, no. 9, pp. 1565–1591, September 2002.

[32] F. Frances, C. Fraboul, and J. Grieu, "Using network calculus to optimize AFDX network," in *Proceeding of the 3thd European congress on Embedded Real Time Software (ERTS06)*, Toulouse, January 2006.

[33] M. Boyer, J. Migge, and N. Navet, "An efficient and simple class of functions to model arrival curve of packetised flows," in *Proc. of the 1st Int. Workshop on Worst-Case Traversal Time (WCTT'2011)*. New York, NY, USA: ACM, novembre 2011, pp. 43–50.

[34] M. Boyer, N. Navet, and M. Fumey, "Experimental assessment of timing verification techniques for AFDX," in *Proc. of the 6th Int. Congress on Embedded Real Time Software and Systems*, Toulouse, France, February 2012.