NoCo: ILP-based Worst-Case Contention Estimation for Mesh Real-Time Manycores

Jordi Cardona
Univ. Politecnica de Catalunya and Barcelona Supercomputing Center
Barcelona, Spain
jordi.cardona@bsc.es

Carles Hernandez, Enrico Mezzetti, Jaume Abella
Barcelona Supercomputing Center
Barcelona, Spain
name.surname@bsc.es

Francisco J. Cazorla
Barcelona Supercomputing Center and IIIA-CSIC
Barcelona, Spain
francisco.cazorla@bsc.es

***Abstract*—Manycores are capable of providing the computational demands required by functionally-advanced critical applications in domains such as automotive and avionics. In manycores a network-on-chip (NoC) provides access to shared caches and memories and hence concentrates most of the contention that tasks suffer, with effects on the worst-case contention delay (WCD) of packets and tasks' WCET. While several proposals minimize the impact of individual NoC parameters on WCD, e.g. mapping and routing, there are strong dependences among these NoC parameters. Hence, finding the optimal NoC configurations requires optimizing all parameters simultaneously, which represents a multidimensional optimization problem. In this paper we propose NoCo, a novel approach that combines ILP and stochastic optimization to find NoC configurations in terms of packet routing, application mapping, and arbitration weight allocation. Our results show that NoCo improves other techniques that optimize a subset of NoC parameters.**

***Keywords*-NoC; mesh; WCET; ILP; contention**

I. Introduction

Computing performance needs in domains such automotive, avionics, railway, and space are on the rise. This is fueled by the trend towards implementing an increasing number of product functionalities in software that ends up managing huge amounts of data and implementing complex artificial-intelligence functionalities [1], [11].

Manycores are able to satisfy, in a cost-efficient manner, the computing needs of embedded real-time industry [2], [10]. In this line, building as much as possible on manycore solutions deployed in the high-performance (mainstream) market [41], [32], contributes to further reduce costs and increase availability.

However, manycores bring several challenges for their adoption in the critical embedded market. One of those is deriving timing bounds to tasks' execution times as part of the overall timing validation and verification processes [28]. In particular, the network-on-chip (NoC) has been shown to be the main resource in which contention arises, and hence hampers deriving tight bounds to the timing of tasks [24].

For widely-used wormhole NoCs (wNoCs) [32], [41], several proposals show how to compute latency upperbounds to the different flows communicating on the manycore [26], [31] under some restrictions, e.g. deterministic routing. Unfortunately, WCET estimates computed with wNoCs are generally pessimistic when – as required to achieve composable estimates – no restrictions are imposed on when and where interference occurs in the wNoC. Interestingly, wNoCs offer several software-controllable parameters that allow to optimize (reduce) the worst-case contention delay (WCD) that packets crossing can suffer. These include mapping, routing, and allocation of weights (referred to as walloc) to arbitration policies in each router. NoC contention optimization solutions have been proposed for mapping [7], [48] and combining routing and mapping [46], [47]. Additionally, optimal allocation of weights to achieve fair bandwidth balancing have been also proposed for TDMA [36] and wNoCs [25]. In general, those solutions do not tackle all parameters at once, which leads to globally suboptimal solutions.

Overall, reducing WCD in NoCs is indeed a multidimensional problem and, to make things worse, strong dependences exists between the different parameters. For instance, the impact of routing in WCD is heavily affected by the mapping of tasks to cores.

Despite the inter-dependences among these parameters, to our knowledge no previous work proposes an integral solution to the problem of WCD reduction simultaneously optimizing mapping, routing and walloc.

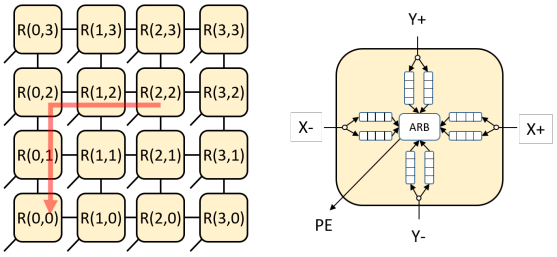
Contribution. In this paper, we cover this gap by proposing *NoCo*, a wNoC ILP- and stochastic-based optimization framework that minimizes WCD estimates (and hence WCET estimates) of applications running in the wNoC-connected manycores. In particular:

① We analyze the main wNoC parameters that cause variability in WCD (mapping, routing, and walloc) and how they relate each other. We propose a modeling approach that allows deriving the contribution of each wNoC parameter to WCD.

② We show that reducing WCD is a multidimensional problem that we decompose into a stochastic-based optimization and an ILP formulation. The former covers the optimization of the routing, whereas the latter optimizes mapping and walloc.

③ We show the effectiveness of NoCo compared to hand-made setups and other approaches that optimize a subset of the parameters. Our results confirm that our multidimensional optimization approach achieves performance guarantees that outperform the other ones evaluated. We also show that optimizing virtual-channel (VC) allocation provides a subset of the configurations obtained with walloc, so that optimizing walloc makes VC not to provide any additional advantage.

We focus on high-performance wormhole NoCs in which time-predictability is achieved by leveraging an optimal configuration of parameters. This includes features like arbitration and routing already configurable from software in existing real wNoC designs. Weight allocation, while to our knowledge it has not been implemented in commercial NoCs yet, it is widely used in high-performance routers for off-chip



(a) (b)

Fig. 1. Mesh basics. (a) Router coordinates in a 4x4 part of a mesh. (b) Ports and VCs competing for output PE port in a canonical 2D-mesh router.

wormhole networks [8]. Given that the implementation cost of weighted arbitration is quite low [25], they can be included with low cost in high-performance on-chip designs. Moreover, modifications required to implement weighted arbitration are local in contrast to hardware proposals that require global changes like, new signals among routers and nodes, different flow-control, global clocks or the like. We cover these specific real-time NoC designs in the related work.

The rest of this paper is organized as follows. Section II shows an approach that allows capturing the impact of every wNoC component on WCD. Section III presents our approach to optimize several wNoC components simultaneously. Section IV assesses the WCD/WCET estimate reduction and overheads of NoCo. Section V presents the most relevant related works. Finally, Section VI summarizes the main conclusions of this work.

II. Abstracting sources of jitter (WCD modeling)

We target standard 2D *NxM* wormhole mesh NoCs, see Figure 1(a), as they deliver high performance and are commercially available [2], [32], [41]. Each node comprises the router that serves as interface to the mesh and a *PE* (Processor element). Each router comprises 5 ports, see Figure 1(b), with routers at the edges having fewer ports. Input ports comprise a queue to store flits, or several if virtual channels are implemented, as shown in Figure 1(b). Main memory is attached to one of the spare ports of one of the routers in a corner. For arbitration, on round-robin as it is a locally fair policy that favors time predictability and it is easy to implement [15].

In our target system critical and non-critical applications run using disjoint resources, either in time or space [24], so that non-critical applications do not interfere with critical ones. For critical applications, communication occurs via main memory given the difficulties of timing analysis techniques to account for the impact of coherence management. This fits software execution models, e.g. avionics for which ARINC 653 [4] uses buffers for intra-partition communication that can be implemented with global (non-cacheable) variables. Inter-partition communication occurs also through buffers so that the sender writes the data, caches are flushed at partition boundaries, thus consolidating all data in memory, and then consumers in other partitions necessarily retrieve data from memory avoiding coherence concerns.

A. Introduction to WCD modeling

The WCD of the packets of an application, and hence the WCET of the application, is heavily affected by the NoC parameters. In particular the NoC has a two fold impact.

TABLE I
Definitions used in this paper.

Term	Definition
$R(x,y)$	Router with coordinates (x,y) in the NoC
F_i	Flow i: stream of flits traversing the same H-node route
R_i^j	For flow F_i is the router at hop j
$\overbar{FX_i^j}$	Set of flows targeting the same output port F_i targets at R_i^j
H_i	Number of hops in a flow F_i
P_i^j	No. of requests that may contend for the same R_i^j output port as F_i under the worst-case contention scenario
ER_i^j	Rate at which flits of flow F_i can be ejected from R_i^j in the absence of backpressure
D_i^j	Maximum time that a packet of F_i requires to go from the input port of R_i^j to its destination node
BW_i	Minimum bandwidth allocated to flow F_i at source node
$\overbar{BW_i^j}$	Minimum bandwidth allocated to flow F_i at R_i^j
BW_i^j	Progated worst-case ejection rate for F_i at R_i^j
NR_i^j	Number of queues that can potentially contend for an output port that F_i is targeting at R_i^j
$\omega(i,j)$	Function returning the index x of the worst possible destination flow F_x that starts at R_i^{j+1} and reaches the worst destination in terms of indirect blocking of packets of F_i

1) It affects the zero-load latency (zll), i.e. the latency experienced by a packet to traverse the network from source to destination in the absence of contention.
2) More importantly, it plays a key role on the contention delay, bounded by WCD, that the application can suffer.

Finding an optimal network configuration to minimize WCD/WCET is a multidimensional problem encompassing several inter-dependent parameters. In order to make the optimization problem tractable, abstractions are needed to capture the impact of different parameters in a common modeling framework. This is challenging since existing abstractions are heterogeneous and not intended to fit ILP models.

In this paper we propose a particular WCD-centric abstraction to address the main sources of jitter in a wNoC, namely: placement of tasks (threads) to cores, routing, and weight allocation (walloc).

Moreover, we also provide an abstraction for VCs that allows us to show that walloc provides a superset of the configurations obtained with VCs, so removing the need of having extra VCs to reduce contention if walloc is in place.

In order to derive the WCD for a flow F_i we build on the formulation in [26], see Equation 1.

Table I describes the terms used in this paper. The first multiplicand provides an upperbound to number of rounds each packet in F_i is stalled until all contending requests in router R_i^j are able to progress, which is $NR_i^j - 1$ for round robin.

The second multiplicand is the *indirect contention delay* each packet of F_i can suffer in each hop due to the worst possible destination flow $F_{\omega(i,j)}$. The worst destination of flow $F_{\omega(i,j)}$ is the one causing the highest contention to F_i in router R_i^j. It can be computed iterating all flows until they target their destination [26]. We also define $w(i,j)$ as a function that returns the index of the worst-destination flow.

$$\overline{WCD_i} = \sum_{j=1}^{H_i} \left(\underbrace{(NR_i^j - 1)}_{ContendingRequests} \times \underbrace{\prod_{m=1}^{H_{\omega(i,j)}} NR^m_{\omega(i,j)}}_{1/Bandwidth} \right) \quad (1)$$

In this paper, we refactor this expression to capture the impact of walloc in the WCD of a flow F_i. In an arbitrary network the time required to process a packet, i.e. its network traversal time, can be computed based on the network bandwidth (BW). Let BW_i be the bandwidth assigned to F_i, then its traversal time can be computed as $1/BW_i$. Note WCD is the result of adding the time the network requires to process the $NR_i^j - 1$ corresponding requests at each of the H_i hops traversed. Then, from Equation 1 we can derive the minimum bandwidth BW_i^j allocated to any packet at R_i^j targeting the same output port F_i targets as follows:

$$BW_i^j = \frac{1}{\prod_{m=1}^{H_{\omega(i,j)}} NR^m_{\omega(i,j)}} \quad (2)$$

Alternatively, and with the aim of simplifying ILP formulation, the minimum allocated bandwidth can be derived building on the concept of worst-case ejection rate (ER_i^j). ER_i^j is the worst ejection rate for flits of flow F_i in router R_i^j through the output port determined by the routing policy whenever the next router in the path (R_i^{j+1}) can accept an incoming packet. ER_i^j can be computed as $ER_i^j = 1/P_i^j$ where P_i^j equals NR_i^j when packets of F_i can be blocked due to contention and 0 otherwise[1]. The effect of back-pressure is covered by the propagated worst-case ejection rate that represents the minimum bandwidth allocated to F_i in router R_i^j. Let $\widehat{FX_i^j}$ be the set of flows that contend for the output port targeted by F_i in R_i^j the worst-case propagated ejection rate $\overline{BW_i^j}$ can be formulated as follows:

$$\overline{BW_i^j} = \min_{\forall F_x \in \widehat{FX}} \left(\prod_{k=j}^{H_w(x,j)} \frac{1}{P_x^k} \right) \quad (3)$$

In the formulation above when P_x^k is 0 the ejection rate and the corresponding allocated bandwidth are defined to be ∞, representing there is no contention due to backpressure and the packets of F_i can progress without contention to the next router. Note also that once in a given router R_i^j packets of F_i suffer from contention, P_x^k values cannot be equal to 0 in subsequent routers due to back-pressure effects.

Building on $\overline{BW_i^j}$, we can derive the WCD that a packet of flow F_i takes to reach its destination node from the input port of router R_i^j. To that end, we define D_i^j as the latency that a packet requires to go from R_i^j to destination. Overall, the recursive definition of D_i^j is as follows:

$$D_i^j = \frac{1}{\overline{BW_i^j}} + D_i^{j+1} \quad (4)$$

[1]Note that there are cases in which in spite of the routing does not prevent flows to contend with F_i no contention with other flows can occur due to the way the flows are mapped to the physical network.

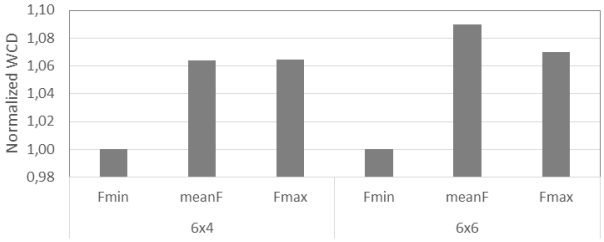


Fig. 2. Tightness of WCD bounds for a 6x4 and 6x6 wNoC

Note that WCD_i is then equivalent to D_i^1 and can be obtained recursively by adding the time to move from each router to the next one, where the time to move from one particular router R_i^j to R_i^{j+1} is upper-bounded by $1/\overline{BW_i^j}$ and equal to 0 when $\overline{BW_i^j}$ is ∞.

We have thoroughly validated WCD values computed using the expressions above by observing WCD values obtained for XY routing and round-robin arbitration match the ones obtained with formulations in [26] and [31] for the analyzed flows. Furthermore, we have assessed the accuracy of our model in upperbounding the actual contention caused in the wNoC by empirically reproducing a worst-congestion scenario using gNoCsim [3]. To that end, we simulate the traffic generated by a memory-intensive micro-kernel in which all tasks in the NoC send packets to memory sustainedly. We have performed this experiment for 2 different network setups. To ensure a steady congestion state is reached, we have taken measurements once at least 1,000 packets per node have been injected. We also repeat the measurement process until all nodes have sent at least 2 million requests. Figure 2 shows the comparison of the analytically computed values with the ones obtained with simulations. The comparison is performed for the the flows with the minimum and maximum WCD (referred to as *Fmin* and *Fmax*, respectively), and the average across all flows, referred to as *meanF*. As it can be seen in the figure our expressions provide a tight upperbound of the NoC contention.

B. Routing

In this work we consider only routing policies with minimal distance routes, that is those policies forward packets in each router in any direction that guarantees that the distance to the target decreases by one hop. For instance, XY routing meets this constraint.

Moreover, due to their suitability for critical real-time systems and their efficiency in terms of implementation, we further assume deterministic routing policies, so that a given packet can only be sent to a specific output port in a router, as dictated by the routing policy. For instance, recalling example in Figure 1(a), a packet sent from $R(2,2)$ to $R(0,0)$ with XY routing can only move in the X direction until $R(0,2)$, and then in the Y direction until $R(0,0)$. Note that with deterministic routing, the route to follow by a packet is independent of whether there is contention in the path. Such contention may delay progress, but not alter the route.

Routing determines for a packet stored in an input port queue of the router the other input ports can contend for a given output port. To simplify the routing algorithm implementation, in the routing policies considered, all flows in the wNoC share the same routing decisions/restrictions. This approach is also followed by the majority of on-chip routing

algorithms including XY, dimension-order-routing, segment-based routing [19], and derivatives.

Routing restrictions help determining the exact number of requests (P_i^j) that might contend at router R_i^j for the same output port as F_i in the worst-case situation. For instance, P_i^j values for a mesh with XY routing and assuming all-to-all communication are determined as: $P_i^j = 2$ if if the destination is X or $X-$ and $P_i^j = 4$ if the destination is Y or $Y-$ or the PE. For a particular routing policy P_i^j is fixed, with different values across routers. Hence, routing can be abstracted by replacing P_x^k in Equation 3 by the actual number of flows that can contend for the output port used by the routing algorithm at each router R_i^j. Therefore, by setting specific routing directions in each router, P_i^j changes for the output port of each router and new D_i^j values (per-core WCD) are obtained for each core.

Deadlock Avoidance. Routing algorithms in wormhole have to ensure deadlock freedom. Deadlock situations in wNoCs occur when packets are waiting on each other in a cycle. For instance, XY algorithm avoids deadlock situations by prohibiting certain turns. We prevent deadlocks by ensuring that there are not prohibited cycles in the generated routing. This can be alternatively achieved by impose further restrictions in the routing inputs of the model or using specific VCs to this end. This however, is orthogonal to our overall formulation just removing some routing options.

C. Weight Allocation (Arbitration)

Weighted meshes allow allocating heterogeneous bandwidth in the routers to the different flows to accommodate the different needs of different communication flows in the wNoC [27]. Weighted arbitration can be employed to achieve a globally-fair (homogeneous) bandwidth allocation across cores [25]. Conceptually, given a NoC with *NxM* nodes, globally-fair weighted meshes reduce the bandwidth for nodes whose allocated bandwidth is above $\frac{1}{N \times M}$ for a given destination node and increases it for those whose bandwidth is below. This is achieved by using, for instance, a larger arbitration window in the case of round-robin, so that a larger number of slots is given to some ports so that the overall bandwidth allocated to each core to the destination can be arbitrarily chosen.

So far we have assumed round-robin with homogeneous weights across flows, so that given P_i^j flows contending for an output port in a router, each one is allowed to eject a packet at a rate $ER_i^j = \frac{1}{P_i^j}$ whenever the next router in the path accepts incoming packets. Hence, the total ejection rate of the output port is $\frac{1}{P_i^j}$ for each of the P_i^j flows.

For instance, given $P_i^j = 3$ contending flows, the default round-robin arbitration policy uses arbitration windows with 3 slots, one of which is given to each flow. Hence, each flow has $1/3$ ejection rate. With weighted arbitration, we can set a window with an arbitrary number of slots (e.g. 5) and allocate them to flows as wanted. For instance, we could allocate 3 slots to flow 1 and 1 slot to each other flow, so that their respective ejection rates would be $3/5$, $1/5$ and $1/5$. Appropriate weights can be set up to modify the WCD of each core. For instance, authors in [25] have shown that homogeneous bandwidth can be achieved by properly allocating weights in routers, which

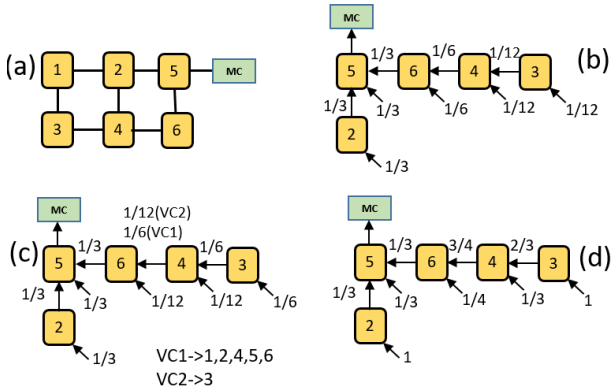


Fig. 3. Example of VCs and its equivalent walloc design.

homogenizes to some extent WCD values[2]. In general, weight allocation can be set as needed – with a granularity limited by the size of the arbitration window size – and abstracted by using appropriate P_i^j values for each flow in each router for the computation of the D_i^j values.

D. Virtual Channels

In canonical wormhole routers, virtual channels (VCs) are used to multiplex physical channels: an input queue resource per port is assigned to a VC. Dynamic VC allocation increases throughput of the wNoC, however, in the context of critical real-time, the dynamic allocation of VCs penalizes WCD [26]. Instead, static VC allocation can alleviate contention in a way that WCD can be reduced by providing isolation if the particular allocation of VCs allows reducing the overlapping between the routes followed by the different flows.

With VCs, two different arbitrations take place in the router, see Figure 1. A first arbitration determines the input port that is granted access to the output port. A second arbitration selects the VC that is granted access. For instance, given a router with 2 VCs that are statically allocated, with 1 flow in the former (VC1) and 3 in the latter (VC2), the arbiter grants access alternatively to each VC, and within the second VC in a round-robin fashion to each flow. Hence, the flow in VC1 has an ejection rate of $1/2$, whereas each of the other flows has an ejection rate of $1/6$. As shown, this formulation is identical to that of the weights for weighted meshes, with the difference that weights can be allocated as needed at much finer granularity.

We illustrate how VCs are subsumed by walloc with an example for a $XY - RR$ 3x2 mesh NoC, see Figure 3(a). Its representation with no VCs in the form of a tree to access the memory controller (MC) is shown in Figure 3(b), where values at the edges indicate the ejection rates for each port. Figure 3(c) shows the tree with 2 VCs where node 3 is allocated to VC2 and the rest of nodes are allocated to VC1. In the mesh without VCs, the bandwidth (BW) is evenly shared across input ports, which makes node 3 to have only 1/12 of the BW (see Figure 3(b)). Instead, when we use the particular VC allocation in Figure 3(c), BW is allocated in a different manner making node 3 enjoy 1/6 of the total BW. This occurs because BW allocation is modified in the links from 6 to 5 (6to5) and from 4 to 6 (4to6) since two VCs are multiplexed

[2]While weights can homogenize bandwidth, they cannot mitigate the communication cost caused by physical distance to destination (zll), so in general, WCD cannot be made fully homogeneous.

over the same physical link. In particular, 6to5 BW (1/3) and 4to6 (1/6) are shared between VC1 and VC2. In 6to5, 1/6 of the BW is allocated to VC1 and the other 1/6 is allocated to VC2. Thus, the node attached to router 6 gets 1/12 of the BW and the remaining 1/12 is allocated to the node attached to router 4. On the contrary, since node 3 uses VC2, its allocated BW is still the one coming from 6to5 (1/6). Finally, once BW per core is determined, it is trivial to set the weights that lead to that particular bandwidth allocation starting from the destination node and splitting it as needed. In this particular example, Figure 3(d) shows the walloc that leads to identical behavior to the case of 2 VCs on $XY - RR$.

E. Mapping

The WCET of parallel and single thread applications can be obtained using the formulation below. For parallel applications WCET is determined by the thread finishing the latest.

The WCET of an individual thread (application) T_i, $WCET_i$, is computed as shown in Equation 5.

$$WCET_i = ETI_i^j + Nreq_i \cdot D_i^j \tag{5}$$

The ETI_i^j figure corresponds to the classical notion of WCET, defining an upper-bound for the execution of T_i *in isolation*, under any possible execution scenario. In a NoC, however, the ETI_i^j bound depends on the node where T_i executes, and must be defined for all nodes j in the system (i.e. all different Cartesian distance values). Since ETI_i^j is independent of contention and we assume routing policies with minimal distance routes, some nodes will share the same ETI_i^j. Hence, deriving the worst-case execution time bounds in isolation of T_i just on a subset of the cores will suffice to represent all potential distances to the memory node. For instance, recalling the example in Figure 1(a), and assuming that memory is located in $R(0,0)$, we could run T_i in the cores at routers $R(0,0)$, $R(0,1)$, $R(0,2)$, $R(0,3)$, $R(1,3)$, $R(2,3)$ and $R(3,3)$, since, for instance, ETI_i^j will be identical at $R(0,2)$, $R(1,1)$ and $R(2,0)$ since all them have the same Cartesian distance to the destination.

Similarly to ETI_i^j, $Nreq_i$ captures the worst-case number of requests triggered by T_i under any possible execution conditions, with the difference that it does not depend on the execution node (i.e., it is constant across cores). The worst-case number of requests is an essential dimension to consider when bounding contention effects: $Nreq_i$ bounds can be derived either statically [22] or based on measurements [9]. Note that it is fundamental to conservatively consider worst-case execution time and number of requests separately as the corresponding scenarios (e.g., input data or execution path) do not necessarily match [9].

Finally, D_i^j is derived according to Equation 4. Ultimately, for a given thread-to-core mapping, D_i^j is constant for each node j (i.e., it is constant once T_i is mapped to a core).

Therefore, the effect of thread-to-core mapping on $WCET_i$ can be abstracted by using the corresponding ETI_i^j and D_i^j values precomputed for the core where the thread is mapped.

III. Formulation

We propose a hybrid NoC Optimization (NoCo) approach to solve the multidimensional problem of NoC parameter optimization. Our approach combines optimization algorithms and ILP formulation to reach its goals. The approach also comprises a less important post-processing module. In particular, building on the analysis Section II, the optimization algorithms of NoCo cover the variability of routing while mapping and walloc are optimized via an ILP formulation This is sketched in the Figure 4.

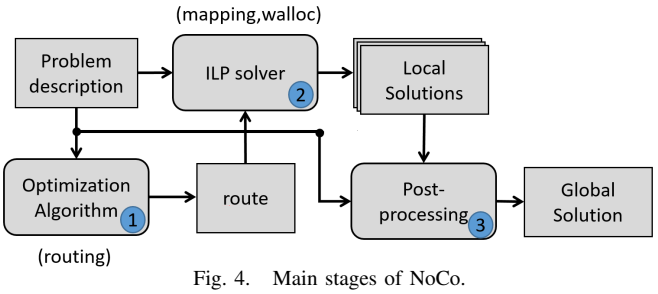


Fig. 4. Main stages of NoCo.

The overall goal of NoCo is to derive the WCET of the application factoring in on NoC contention.

Under a given routing, the zll for a given thread is constant given a specific core c while the WCD for the packets sent by that core (D^c) depends on the particular routing (r) followed by the packets of the other cores, walloc (w), and VCalloc (v) used. Formally stated, $D^c = f(r, w, v)$. In fact, as we show later, given a routing r, the WCD imposed by a given walloc w and VCalloc v can be obtained without using VC, since there exists a walloc w' that delivers the same WCD across cores. Hence, $D^c = f(r, w, v) = f'(r, w')$.

It is noted that, a holistic approach modeling also routing as an ILP variable, would cause the optimization problem to become quadratic as the correlation between bandwidth quotas assignment and routing cannot be modeled with linear constraints. In contrast to ILP or MIP (Mixed-Integer Programming) problems, quadratic optimization problems are generally NP-hard and state-of-the-art solvers are normally incapable of proving the optimality of any solution possibly found.

For the routing optimization, we opted for a stochastic approach instead of an heuristic-based one for a two-fold reason. First, heuristic-based solutions present the problem that in general it is is not possible to assess their quality, since they might be subject to local maxima. Furthermore, the use of routing-only heuristics to find a (local) solution can result in negative overall results when walloc and mapping optimization are applied. And second, as detailed later in this section, a stochastic-based solution allows exploring a restricted number of routes so with limited exploration time, while allowing to argue that the best evaluated route belongs to the top $X\%$ best routes.

A. Routing

For time predictability reasons, we stick to static (predictable) routing that must further avoid deadlocks. In particular, we explore XY, referred to as '0' in the following figures, and YX ('1') routing for each core. Combining both allows achieving good malleability in limiting the number of contenting flows P_i^j in the routers of a particular flow F_i. Figure 5 illustrates with an example how the proposed routing determination algorithm works. Figure 5(a) sketches the graphical convention we use to represent the routing selected by each node and the number of flows contending in each output port. If we apply XY for all nodes, $r_0 = (000000000)$, we obtain the routes in Figure 5(b) and the contention per

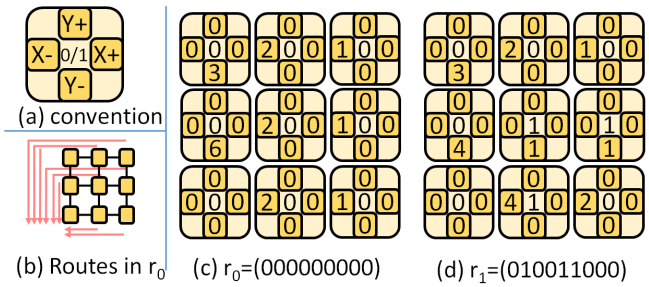


Fig. 5. Examples of different routings and pressure on different output ports

output port as shown in Figure 5(c). We can see that some output ports suffer high contention (up to 6 flows). A different arbitrary routing $r_1 = (000010010)$ presents lower maximum contention per output port, see Figure 5(d).

It follows that good routes are those that limit the number of contenting flows P_i^j in the routers used by of a particular flow F_i. However, this is a local (i.e. routing-only) optimization. When combined with mapping and walloc optimization, routes that create more contention – and hence are less optimal from the routing point of view – can result in reduced maxWCD/sumWCD. For instance, let us assume that under a particular routing one route suffer high contention while the rest suffers low contention. Further assume a second routing that much better balance contention. For an application in which one thread (task) is insensitive to NoC WCD, while the rest of the threads are, the former (less balanced) routing results in reduced WCD and hence WCET.

Hence, since a priori we cannot determine what a good route is, it would be hard to define a standard heuristic approach to address the problem (e.g. genetic algorithms, simulated annealing). Besides, those heuristics would not allow assessing how far a solution (routing) is from the optimal one.

In order to cover both issues, we use a stochastic approach based on a Monte-Carlo experiment. Basically, we produce static routings schemes by selecting randomly whether each node uses XY or YX routing. Hence, from the finite – but huge – population of all routings, a sample is selected using random sampling with replacement. This can be done with following the simple approach shown in algorithm 1: for every node in the NxM mesh we generate a random integer: if it is odd we assume XY routing (0) and YX (1) otherwise. We ensure resulting configurations are deadlock-free by filtering out the samples in which routing cycles are created [19].

Algorithm 1 Algorithm to generate random routings

```
1: procedure GEN_ROUTE_RANDOM(sd,iter,type, ncount)
2:     for (i = 0; i < NxM; i++;) do
3:         mapping[i] = random() mod 2;
```

With this approach we can probabilistically reason on the quality of a given routing. Let C be the probability that a sample of random routings contains at least one of the top X ($X \in [0..1]$) routings, i.e. fraction of routing from the entire population providing the best results for a given target metric (e.g. maximum WCD across cores). Let $\bar{C}$ the complementary of C, i.e. $1-C$. The probability that a single random routing is not in the top X of the population is $(1-X)$. The probability that all k mappings do not belong to the top X is, therefore, $(1-X)^k$. Hence, its complementary, C, is the probability that the best routing choice in the random sample belongs to the top X routings. Hence, $C = 1-(1-X)^k$.

TABLE II
$\bar{C}$ FOR DIFFERENT RANDOM SAMPLE SIZES.

	$\bar{C}$				
X	$\mathbf{10^{-2}}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-4}}$	$\mathbf{10^{-5}}$	$\mathbf{10^{-6}}$
$\mathbf{10^2}$	0.37	0.90	0.99	0.999	0.9999
$\mathbf{10^3}$	$4.3 \cdot 10^{-5}$	0.37	0.90	0.99	0.999
$\mathbf{10^4}$	$< 10^{-43}$	$4.5 \cdot 10^{-5}$	0.37	0.90	0.99
$\mathbf{10^5}$	$< 10^{-300}$	$< 10^{-43}$	$4.5 \cdot 10^{-5}$	0.37	0.90
$\mathbf{10^6}$	$< 10^{-300}$	$< 10^{-300}$	$< 10^{-43}$	$4.5 \cdot 10^{-5}$	0.37

As illustrated in Table II, the probability of not having, for instance, any routing within the top best 0.01% routings ($\bar{C} = 10^{-4}$) with a sample size of $100,000$ is of around $4.5 \cdot 10^{-5}$ (so 0.0045%). Thus, the probability of having at least one of those top $X = 0.0001$ routings is $C = 0.999955$ (above 99.99%). In general, we observe that a sample size of around $1,000$ random routings allows guaranteeing with very high confidence that at least one of the top 1% best routings is observed.

Once a routing is fixed, we can derive the WCD for every packet going from a given node to the memory. The route information is encoded in a *route*, see Figure 4, passed to the ILP model to optimize mapping and walloc.

B. Mapping and walloc

The worst-case contention delay (WCD) potentially suffered by a task τ_i upon each performed memory access, when executing on a weighted mesh NoC, is determined by the interrelation of several factors: the router τ_i is mapped to, the adopted routing configuration (with its inherent flow constraints) and the bandwidth distribution along the mesh [3]. The WCD has to be accounted for in the definition of the worst-case execution time (WCET) of all tasks. We present an ILP formulation for optimizing the WCET of a task by finding an optimal task mapping and bandwidth assignment, under a given routing configuration.

We consider a 2D mesh NoC comprising a set of routers $\mathcal{R} = \{R_0, \ldots, R_s\}$, with associated computational nodes $\mathcal{N} = \{N_0, \ldots, N_s\}$ (such that N_k is attached to R_k) and a set of tasks $\mathcal{T} = \{\tau_0, \ldots, \tau_n\}$, that need to be executed on the mesh NoC. Each task τ_i is characterized by the a number of performed memory accesses a_i, and an execution time bound computed in isolation, dependent on the node it executes on: we use the notation c_i^k to represent the timing bound of τ_i, when executed on node N_k (implicitly accounts for the routing policy). The WCET of a task τ_i when executed on router R_k, with a given bandwidth quota, is obtained by inflating the execution time in isolation with a worst-case delay penalty for each memory access a_i:

$$WCET_i^k = WCD^k * a_i + c_i^k \tag{6}$$

In turn, the worst-case delay WCD^k potentially suffered by τ_i when executed on node N_k is determined in particular by the routing policy in use, and the specific bandwidth assignment among nodes.

[3]In this subsection we refer to the walloc optimization problem as bandwidth distribution problem.

TABLE III
ILP MODEL NOTATION.

$\mathcal{R} = \{R_0, \ldots, R_s\}$	Set of routers in the mesh
$\mathcal{N} = \{N_0, \ldots, N_s\}$	Set of computational nodes in the mesh
$\mathcal{T} = \{\tau_0, \ldots, \tau_n\}$	Task set to be executed on the mesh
a_i	Number of memory accesses triggered by τ_i
c_i^k	Execution time in isolation of τ_i when executed on node N_k
$WCET_i^k$	Worst-case execution time of τ_i when executed on node N_k
WCD^k	Worst-case delay suffered by a task mapped to node N_k
$\mathcal{M} : \mathcal{T} \mapsto \mathbb{Z}$	Mapping of τ_i to indexes in the node set $\mathcal{N}$
$\mathcal{B} : \{\mathcal{R} \cup \mathcal{N}\} \mapsto \mathbb{R}$	Mapping of R_k and N_k to a bandwidth assignment
$WCET_i^{\mathcal{M},\mathcal{B}}$	WCET for task τ_i under mapping $\mathcal{M}$ and bandwidth allocation $\mathcal{B}$
$WCD^{\mathcal{M}(i),\mathcal{B}}$	WCD suffered by τ_i, under mapping $\mathcal{M}$ and bandwidth allocation $\mathcal{B}$.
$\mathcal{H} : \mathcal{R} \mapsto \mathcal{P}(\mathcal{R})$	List of routers (hops) a packet needs to traverse to reach a destination from a given source router
$\mathcal{L} : \mathcal{R} \mapsto \mathcal{P}(\mathcal{R})$	Map of active links in the mesh
$BWR_k^{\mathcal{B}}$	Bandwidth assigned to the output port of router R_k
$BWN_k^{\mathcal{B}}$	Bandwidth assigned to the output port of the computational node attached to router R_k

Given a fixed routing policy (configuration), our ILP model leverages on task mapping and bandwidth assignment to optimize the WCET of tasks.

B.1. Objective Function

The ILP formulation supports two objective functions representative of two typical metrics for assessing the performance of a wNoC. We are interested in finding the optimal task-to-node mapping and bandwidth assignment that minimizes:

- *SumWCD*. For independent workloads, reducing the addition of the WCD of all tasks/threads, helps optimizing resource usage and improving overall guaranteed performance. Reducing *SumWCD* ca be done under some constraints on the maximum WCD allowed per task. Overall this metric, which minimizes the WCD experienced by all requests of all threads in the workload, is particularly relevant to assess the global efficiency of our approach.
- *MaxWCD* is derived by computing the total WCD experienced by all requests for each thread, and minimizing the maximum WCD value across threads. For parallel applications, this metric provides information about the thread experiencing highest contention and hence potentially delaying the completion of the application.

We introduce $\mathcal{M}(i) : \mathcal{T} \mapsto \mathbb{Z}$ to define a mapping from tasks to computational nodes in the mesh, where $\mathcal{M}(i)$ returns the index k that identifies the node $N_k \in \mathcal{N}$ such that τ_i is mapped to N_k. Note that N_k is by definition attached to router R_k. Similarly, we define $\mathcal{B} : \{\mathcal{R} \cup \mathcal{N}\} \mapsto \mathbb{R}$, mapping from routers and nodes to a bandwidth quota, where $\mathcal{B}(R_k)$ returns router R_k bandwidth assignment and $\mathcal{B}(N_k)$ returns the same for node N_k. It is worth noting that $\mathcal{B}$ depends on the specific routing configuration.

A formulation for the WCET parametric on task mapping and bandwidth assignment would be as follows, where $WCD^{\mathcal{M}(i),\mathcal{B}}$ identifies the per-access WCD potentially suffered by τ_i, under mapping $\mathcal{M}$ and bandwidth allocation $\mathcal{B}$.

$$WCET_i^{\mathcal{M},\mathcal{B}} = WCD^{\mathcal{M}(i),\mathcal{B}} * a_i + c_i^{\mathcal{M}(i)} \tag{7}$$

For *maxWCD*, the ILP is meant to optimize the time required to execute the longest activity in the mesh.

$$\min_{\mathcal{M},\mathcal{B}} \max_{\tau_i \in \mathcal{T}} WCET_i^{\mathcal{M},\mathcal{B}} \tag{8}$$

For *sumWCD*, instead, ILP aims at minimizing the resources required to execute whole workload.

$$\min_{\mathcal{M},\mathcal{B}} \sum_{\tau_i \in \mathcal{T}} WCET_i^{\mathcal{M},\mathcal{B}} \tag{9}$$

B.2. Modeling WCD

Similarly to the execution time in isolation, WCD is affected by the task location in the mesh and the routing policy in use. However, while the set of C_i^k is an input variable to the ILP model, the WCD is indirectly a decision variable, as a function of bandwidth allocation. The part taken by the routing policy in the computation of the WCD have to be made explicit, in the same way as the mesh bandwidth assignment. The bandwidth allocated to each element in the mesh is modeled by an explicit decision variable: $BWR_k^{\mathcal{B}}$ defines the bandwidth allocated to the output port of router R_k under bandwidth assignment $\mathcal{B}$, whereas $BWN_k^{\mathcal{B}}$ stands for the bandwidth assigned to the computational node N_k, attached to R_k.

Routing is indeed relevant for determining both the packet route and the feasible bandwidth split rules. We defined two abstraction, $\mathcal{H}$ and $\mathcal{L}$ to capture these relevant aspects.

$\mathcal{H} : \mathcal{R} \mapsto \mathcal{P}(\mathcal{R})$ models the list of routers (hops) a packet needs to traverse to reach a destination from a given source router. This information is necessary to accumulate the WCD along the end-to-end flow.

$\mathcal{L} : \mathcal{R} \mapsto \mathcal{P}(\mathcal{R})$, instead, is a map of the active links in the mesh, in accordance to the routing rules. This information is fundamental to encode the rules for bandwidth allocation.

Given a routing policy defined by the pair $< \mathcal{H}, \mathcal{L} >$, we model bandwidth allocation rules and WCD bound as follows. First Eq. 10 models the fact that the bandwidth assigned to the output port of a router R_k is determined by the bandwidth in the node-local computational node BWN_k, and the cumulative bandwidth propagated by other routers connected to R_k through an active link.

$$BWR_k^{\mathcal{B}} = BWN_k^{\mathcal{B}} + \sum_{R_t \in \mathcal{L}(R_k)} BWR_t^{\mathcal{B}} \tag{10}$$

Having defined the bandwidth per router, it is possible to model the WCD for a router R_k as follows:

$$WCD_i^{\mathcal{M},\mathcal{B}} = \frac{1}{BWN_{\mathcal{M}(i)}^{\mathcal{B}}} + \sum_{R_t \in \mathcal{H}(\mathcal{M}(i))} \frac{1}{BWR_t^{\mathcal{B}}} \tag{11}$$

As defined in Eq. 11, the WCD per-access for a task mapped to router R_k is determined by the cumulative inverse bandwidth across all hops in the path from source to destination. The computed WCD can be used for the WCET computation in Eq. 7.

B.3. Modeling Constraints

The mesh topology and routing policy allow to derive a set of constraints to guide bandwidth allocation across routers

and task mapping. Some simple constraints can put on $\mathcal{B}$ by defining the domain space for the ILP variable:

- The WCD is always greater than zero.

$$WCD_i^{\mathcal{M},\mathcal{B}} > 0.0 \quad \forall i, \mathcal{M}, \mathcal{B}$$

- Allocated bandwidth per routers must be larger than zero.

$$BWR_k^{\mathcal{B}} > 0.0 \quad \forall k, \mathcal{B}$$

- The cumulative amount of bandwidth allocated to all computational nodes must be exactly one

$$\sum_{R_k \in \mathcal{R}} BWN_k^{\mathcal{B}} = 1$$

Similarly, several constraints can be defined on $\mathcal{M}$.

- Each router cannot be assigned to more than one thread

$$\mathcal{M}(i) \neq \mathcal{M}(j) \quad \forall i, j$$

- Each thread can only be assigned to one router

$$|\mathcal{M}(i)| = 1$$

B.4. Putting it all Together

NoCo optimizes routing with a stochastic approach that allows assessing the quality of the explored routes. The alternative approach of expressing the routing as a decision variable in the problem formulation has the notable drawback of breaking the linearity of the model. Modeling the routing as the combination of two new variables, representing the bandwidth distribution and the flow configuration, would turn the computation of both the router bandwidth quotas and, ultimately, the worst-case delay into a quadratic optimization problem. For each route, an ILP model optimizes mapping and walloc (bandwidth allocation) to minimize either *maxWCD* or *sumWCD*. In the next section we empirically assess the benefits of NoCo over other existing approaches.

IV. Experimental Evaluation

A. Manycore Processor Model

We conduct the evaluation of NoCo on an integrated simulation framework. It includes an enhanced version of SoCLib [39], a cycle-accurate multicore processor simulator, and gNoCSim [3], a cycle-accurate NoC simulator. We model a tile-based manycore [32], [41] where each tile comprises L1/L2 cache memories and a core that communicates with the rest of tiles and memory using a NoC router.

Processor cores implement an in-order pipeline with 32KB 4-way 16B/line IL1 and DL1 caches, where DL1 is write-through. The manycore also includes a unified distributed shared L2 cache memory, so that each core has a local partition of the L2 cache. To increase the predictability of the cache hierarchy the L2 cache is partitioned and each core is provided with a 64KB region with 64B per cache line. L2 partitioning does not only avoid inter-task interferences in the L2 but allows both isolating cache coherence between different critical and non-critical tasks [18] and/or to disable coherence support to avoid further NoC interferences.

Memory requests access main memory through the NoC. Each memory request operates at a granularity equal or smaller to a L2 cache line, thus transferring up to 16B (128 bits) per NoC packet, which also has 16 control bits. Overall, packets with up to 144 bits are sent in a single flit through a 144-bit link width, which allows all NoC packets affecting our tasks to have one-flit. Despite one-flit packets are preferred to reduce contention, our model is also able to deal with other packet lengths by computing worst-case ejection rates considering packets of the maximum length.

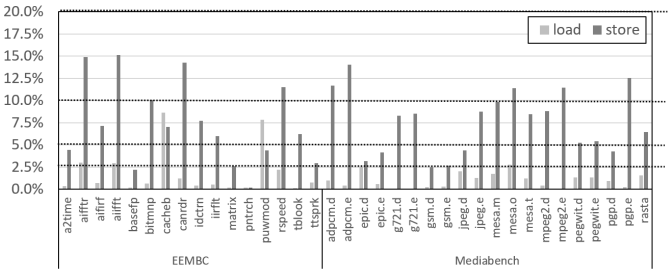


Fig. 6. Percentage of load and store in each benchmark

Note that routers are connected through 2 links, each one sending data in one direction. Hence, whereas memory accesses may experience contention, memory responses cannot since, at every router only one input port (the one used for responses) contends for the corresponding output port (the one to move the response to destination).

B. Workloads

In the real-time domain, manycores are exploited by speeding up single (multithreaded) applications that use several cores or consolidating multiple single-task applications, which increases overall throughput. The latter is more frequent in domains with several software partitions (e.g. IMA [44] in avionics). In order to cover both scenarios, we design two types of workloads, namely with independent single-task applications and with a single parallel application.

In the case of parallel applications, we assume those supported by the ADA programming language [33], where, essentially, a single-threaded phase is followed by a parallel phase where all threads are spawned simultaneously and they synchronize upon completion with a barrier.

To capture the degree of load that different applications would put on the NoC, we have generated several benchmarks, named A, B, ..., H that put specific load on the NoC. In order to use representative values, we have analyzed the load imposed by two different reference benchmark suites:

- MediaBench [17] is a well-known benchmark suite comprising multimedia and communication applications relevant for many critical real-time systems, especially for autonomous navigation and driving systems.
- EEMBC Autobench [29] benchmark suite includes automotive applications relevant for critical real-time systems.

As shown in Figure 6, the particular load imposed by the different benchmarks for both load and store operations varies significantly, being up to 15% for each type of application. We create different benchmarks (A, B, ..., H) to impose access frequencies that include the range of those benchmark suites. In particular, we consider load and store access frequencies between 2.5% and 40% so that scenarios evaluated are relevant for those benchmark suites and for further stressful conditions where NoC contention could have a higher impact. The particular details for those benchmarks are shown in Table IV.

From this set of benchmarks, we have generated workloads with NxN benchmarks for each mesh size analyzed. Some workloads are homogeneous, thus having all benchmarks of

TABLE IV
GENERATED APPLICATIONS AND THE PERCENTAGE OF LOCAL, LOAD, AND STORE OPERATIONS THEY HAVE. CORE OPERATIONS ARE THOSE OPERATION NOT USING THE NOC, I.E. CORE OPERATIONS AND MEMORY OPERATIONS HITTING IN LOCAL CACHES.

Benchmarks	A	B	C	D	E	F	G	H
% Load	10	10	40	20	2.5	2.5	10	5
% Store	10	40	10	20	2.5	10	2.5	5
% Local	80	50	50	60	95	87.5	87.5	90

the same type (so 8 workloads in total), whereas others (4 workloads) are heterogeneous by choosing the NxN benchmarks randomly (with replacement) out of our set of 8.

C. Reference techniques

As reference approaches to compare NoCo against, we use:

- $XY-RR$ deploys predictable XY routing with standard predictable round-robin arbitration in each router.
- $XY-WRR$ is analogous to $XY-RR$ except that we modify weights to balance the bandwidth across all nodes [25]. This makes that the WCD of all nodes accessing memory is more homogeneous than in $XY-RR$, effectively mitigating mapping as a source of variability.

In the rest of this section we refer to our technique as $rILP(m,w)$ since NoCo optimizes routing, r, first using the stochastic approach in Section II-B with samples of size 10 for 3x3 experiments and 100 for 4x4 experiments, and mapping and walloc using the ILP approach, $ILP(m,w)$, presented in Section III. Results are shown in the form of WCET reduction w.r.t. the $XY-RR$ case. For instance, the maxWCET improvement of $ILP(m,w)$ is obtained as:

$$1-\frac{maxWCET_{ILP(m,w)}}{maxWCET_{XY-RR}} \tag{12}$$

D. Incremental evaluation

In order to assess the benefits of optimizing each individual parameter of a NoC (mapping, routing, and walloc), we present the results obtained with our approach as it incrementally optimizes them. In particular we compare these setups:

(a) $XY-WRR-ILP(m)$. Both routing (XY) and weights (WRR) are fixed. Therefore, NoCo only optimizes mapping, i.e. it explores the different mappings of tasks to cores ($ILP(m)$).
(b) $XY-ILP(m,w)$. Only routing is fixed (XY) and NoCo explores (i.e optimizes) both mapping and weight allocation at the arbitration level ($ILP(m,w)$).
(c) $rILP(m,w)$. All three NoC parameters are optimized by NoCo: routing, mapping and walloc.

With this incremental approach we can derive the benefits of optimizing each parameter: (a) gives the benefits of optimizing mapping; (b)-(a) the benefits of optimizing walloc; and finally (c)-(b) the benefits of optimizing routing.

Figure 7 shows the impact of applying the optimization of the different parameters incrementally. In particular, we evaluate the 4 heterogeneous 9-task workloads, which we refer to as MIX1 to MIX4 on a 3x3 mesh NoC. Note that the chart shows the maxWCET reduction that each technique obtains over the baseline $XY-RR$.

Compared to the baseline $XY-RR$, we can see that $XY-WRR$ [25] obtains similar results (-1%) for three of the workloads, while for one workload it is 16% better.

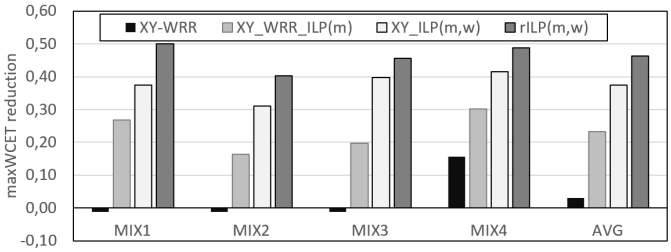


Fig. 7. Effect of incremental optimizations: mapping, walloc, and routing.

Effect of mapping. Compared to the baseline $XY-RR$, $XY-WRR-ILP(m)$ obtains maxWCD reductions of 23% on average, showing the benefits of optimizing task mapping.

Effect of mapping and walloc. The combined effect of mapping and walloc $XY-ILP(m,w)$ results in further reductions of maxWCD across all workloads. On average improvements 37%, so that the benefit of optimizing only $walloc$ is 14 percentage points.

Full optimization. When NoCo optimizes all three NoC parameters it produces the tightest WCET estimates, with maxWCET reductions of 46% on average, so that the benefit of routing optimization is 9 percentage points.

Overall can see that results are consistent across all workloads ranging from 40% to 50%. These results show the benefits of simultaneously optimizing routing, mapping, and walloc and how this provides the best WCET reduction results.

We evaluate $rILP(m,w)$ in the rest of this section without further breaking down experiments incrementally.

E. Optimal routing

In this section we compare the random routing selection to exploring all possible routings. While the latter is not feasible in general due to its huge execution time overheads, for the small set of experiments we evaluate it provides evidence on the benefits of the stochastic approach to optimize routing.

Figure 8 shows the maxWCET obtained with random routing selection normalized so that 100% corresponds to the lowest maxWCET and 0% to the highest maxWCET. Thus, we show how close to the optimal is the best solution so far.

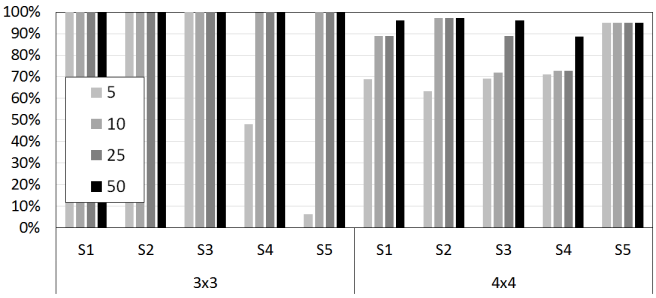
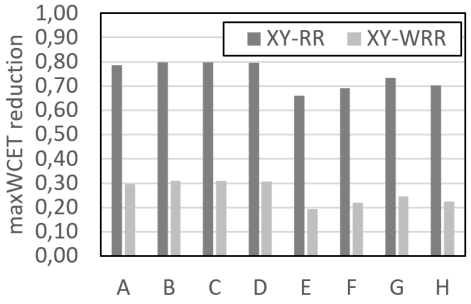
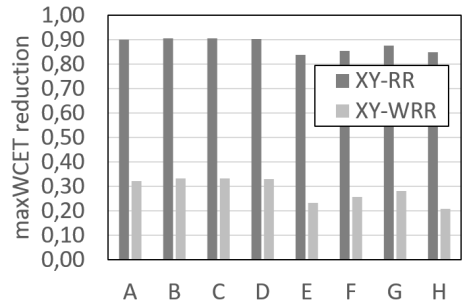


Fig. 8. Normalized maxWCET of random routing w.r.t the best routing

We have performed two experiments for 3x3 and 4x4 respectively. In both cases, we take 5 samples, each of which with just 5, 10, 25 and 50 of all possible routings, i.e. 512 for 3x3 and 65,536 for 4x4. Across all five samples we can see that the maxWCET results obtained with random routing are very close to those obtained with the best routing with samples of 50 observations. In the case of 3x3, sample sizes of 10 already find optimal results. For 4x4, samples for size 10 get on average a solution 85.2% optimal. With samples of 50, the average is 94.6% and the worst case 88.7%. With

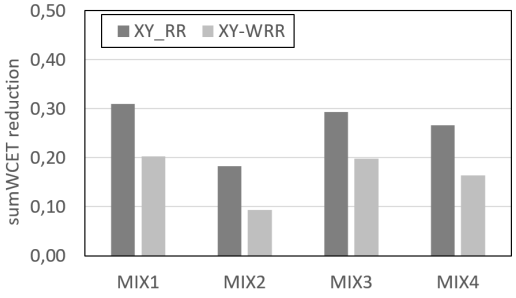


(a) 9 threads

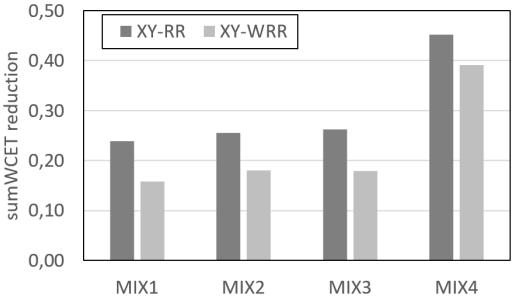


(b) 16 threads

Fig. 9. rILP(m,w) maxWCET results for 9- and 16-thread applications



(a) 9 threads



(b) 16 threads

Fig. 10. rILP(m,w) sumWCET results for 9- and 16-task workloads

samples of 100 (not shown in the plot), the average is 98.6% and with samples of up to 330 (0.5% of the population) the optimal solution is found in the 5 samples.

F. Homogeneous Workloads

For homogeneous workloads, the optimization goal we set for NoCo is reducing the WCET of the thread with longest WCET (maxWCET). For parallel applications using coarse-grain parallelism, the thread with lowest performance is usually the one determining the WCET of the application and thus, optimizing maxWCET minimizes the overall WCET of the parallel applications.

It is also worth mentioning that for homogeneous workloads thread mapping plays no role since all threads are identical. Hence, gains come from walloc and routing optimization only.

Figure 9(a) and Figure 9(b) compare the WCET reduction of the full NoCo optimization, i.e. $rILP(m,w)$, against the reference $XY-RR$ and $XY-WRR$ designs for 9-thread and 16-thread applications, respectively. Those applications we map to a 3x3 and 4x4 mesh NoCs respectively.

For 9-thread workloads, $rILP(m,w)$ achieves average 74% WCET reductions w.r.t. $XY-RR$ and 26% w.r.t. $XY-WRR$. The rationale behind these results is as follows:

- WCD values for $XY-RR$ are highly heterogeneous and hence, the WCD experienced by the thread at the core with highest contention is far higher than that of most of the other threads.
- By using WRR, WCD values become more homogeneous, thus significantly decreasing the opportunities for optimization in the context of homogeneous workloads. Yet, even in this case, $rILP(m,w)$ decreases maxWCET significantly (26% on average).

Results across workloads show that those with higher access frequencies obtain higher benefits with NoCo, since the impact of WCD on their WCET is higher. Still, we observe that decreasing NoC requirements by 4x only decreases gains from 26% to 22%, thus showing the importance of optimizing NoC configuration to improve performance.

For 16-thread applications maxWCET reduction follows the same trend, though improvements are more noticeable. In particular, $rILP(m,w)$ decreases maxWCET by 88% and 29% on average w.r.t. $XY-RR$ and $XY-WRR$ respectively, with increased gains occurring consistently across all workloads.

G. Heterogeneous Workloads

For heterogeneous workloads, NoCo focuses in reducing sumWCET, hence actually reducing the overall impact – and wasted resources due to contention. While in the experiments in this section, NoCo optimizes all three NoC parameters to reduce sumWCET with unrestricted per-task WCET, our formulation supports setting specific bounds to the WCET for some tasks. Unlike homogeneous workloads, for heterogeneous ones, (task) mapping plays a role in optimizing performance – as it was analyzed in Section IV-D.

Figure 10 shows the sumWCET reduction of NoCo with respect to $XY-RR$ and $XY-WRR$ designs for 9-task workloads (Figure 10(a)) and 16-task workloads (Figure 10(b)).

For the 9-task workloads, $rILP(m,w)$ we observe pretty consistent improvements. In particular, with respect to $XY-WRR$ improvements are similar to those obtained for homogeneous workloads ranging from 17% to 22% (19% on average). We also see that for the heterogeneous workloads, the results of RR are not as bad as for the homogeneous. Yet NoCo improves $XY-RR$ by 30% on average. Results for the 16-task workloads support that $rILP(m,w)$ achieve consistent reduction w.r.t. the other two techniques.

H. Other Metrics

In previous sections we have used two optimization criteria sumWCET and maxWCET. NoCo also supports other criteria such as, for example, limiting the maximum WCET/WCD of tasks. This is better illustrated with an example that uses the homogenous 9-thread workload A that we run in a 3x3 network under a fixed mapping. In a first experiment we run NoCo minimizing sumWCET. This produces the WCET shown in the upper part (1) of Table V. In a second experiment

we run NoCo, still minimizing sumWCET, but also enforcing that τ_8 cannot exceed 11.56 (million cycles). The result of this experiment is shown in the lower part (2) of Table V. As it can be seen, the WCET of τ_8 is indeed 11.55, which is achieved by maintaining its WCD below 3.3. As a side effect, we observe a small increase in sumWCET, from 87.9 to 88.2.

TABLE V
EFFECT OF LIMITING THE WCET OF ONE THREAD. ET STANDS FOR EXECUTION TIME IN ISOLATION. TIME SHOW IN MILLION CYCLES.

		τ_0	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8
1	ET	4.99	5.81	6.63	5.81	6.63	7.44	6.63	7.44	8.26
	WCD	2.27	2.89	3.57	2.56	2.97	3.83	2.89	3.28	3.98
	WCET	7.26	8.70	10.20	8.37	9.59	11.28	9.51	10.72	12.24
2	ET	4.99	5.81	6.63	5.81	6.63	7.44	6.63	7.44	8.26
	WCD	2.55	2.93	3.80	2.56	3.18	3.86	3.07	3.29	3.29
	WCET	7.54	8.74	10.43	8.36	9.80	11.30	9.69	10.73	**11.55**

V. RELATED WORK

We have broadly classified the most relevant related works into: (1) NoC designs for hard-real time systems, and (2) optimization approaches to minimize NoC contention.

Real time NoC designs. *Contention-free NoCs* have been traditionally considered the best fit to hard-real time applications. Contention free communication can be achieved by using time-division multiplexing [12], [20], [35], time-triggered architectures [23], and other ad-hoc wormhole-based designs like SurfNoC [43] or PhaseNoC [30]. However, as the number of cores included in the processor increases time-division multiplexing approaches loose competitiveness since differences between worst-case performance and average performance increase with the number of cores [28]. Furthermore, customized NoCs will naturally find difficulties in being adopted by industry [40] since their implementation incur high non-recurrent costs. *Priority-based flit-level preemption NoCs* based on customized wormhole NoCs in which different priorities are assigned to the existing flows in the NoC have been proposed. The initial approaches [6], [16], [37] are based on assigning a different virtual-channel to each of the flows in the NoC such that the impact of NoC contention is accounted for in the schedulability analysis. Enhanced hardware designs have been proposed in this context to improve performance of both, low-criticality and high-criticality traffic [13]. More recent approaches have shown that virtual channels can be reduced by ensuring a different priority (virtual channel) is assigned to the flows that actually share one or more links [21]. Virtual-channel requirements can be further reduced if the analysis includes the effect of contention of flows sharing priorities [38]. More recently, authors in [45] have shown that the model proposed in [37] and later enhanced [16] has some deficiencies. Based on the findings in [45], authors in [14] have proposed a tight alternative model for priority-preemptive real-time networks. *Best-effort wormhole NoCs* can also be employed in the context of hard-real time systems by deriving latency upperbounds as described in [31]. However, as shown in [26] and [42] these bounds can be pessimistic when time-composability properties have to be preserved. To mitigate this problem authors in [25] proposed introducing packetization and weighted arbitration to improve the latency bounds provided by best-effort wormhole NoCs.

Wormhole NoCs optimization approaches. In [5] integer linear programming models of heterogeneous multi-cores are used to find the optimal task layout that guarantees execution time. In this approach the internals of the communication infrastructure are not exposed to the solver. Since the adoption of NoCs as the primary interconnection architecture for multi/many-cores many works have targeted the problem of task mapping [34]. Authors in [7] proposed an ILP formulation for a contention-aware application mapping algorithm in tile-based NoC using meshes with XY routing to minimize inter-tile network contention being able to achieve a significant reduction of packet latency. In [47] the scheduling and mapping of tasks is combined to minimize packet latency and increase predictability in the context of meshes with flexible routing decisions. To that end, communications have to follow a regular access pattern. A similar approach is the one in [48] that in the context of 2D mesh with XY routing that uses a constraint solver to find the mapping where contention is minimized. Once the mapping is fixes communication bursts are mapped onto frames that are time-multiplexed.

Positioning of this paper. Our work targets providing real-time guarantees in the context of high-performance wormhole NoCs thus, fitting in the area of best-effort NoCs. We use an ILP-based optimization approach to allow achieving improved performance guarantees so its spirit is similar to the one of the optimization approaches presented above. However, our approach targets fine-grain communications representing memory accesses whose behavior is not explicitly handled by the application or the OS. Also, unlike other proposals that focus on a subset of NoC parameters our optimization algorithm deals with the most relevant user-controllable parameters like routing, thread allocation, and weight allocation at once to leverage the performance provided by wormhole NoCs.

VI. CONCLUSIONS

We have presented NoCo a framework that leverages the configuration potentials of wormhole NoCs via a hybrid stochastic/ILP approach. NoCo finds the best routing configurations with a stochastic approach and applies ILP to the generated routing policies to find the optimal mapping and weights allocation for each of them. We show that NoCo achieves significant improvements over other optimization strategies that focus in just a subset of the NoC parameters. NoCo outperforms reference XY-RR and XY-WRR wNoC designs for both heterogeneous and homogeneous workloads, and also in the context of parallel and single-thread workloads.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness under grant TIN2015-65316-P and the HiPEAC Network of Excellence. It also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (agreement No. 772773). Carles Hernández is jointly supported by the MINECO and FEDER funds through grant TIN2014-60404-JIN. Jaume Abella has been partially supported by the Spanish Ministry of Economy and Competitiveness under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717. Enrico Mezzetti has been partially supported by the Spanish Ministry of Economy and Competitiveness under Juan de la Cierva-Incorporación postdoctoral fellowship number IJCI-2016-27396.

REFERENCES

[1] Intel GO Automated Driving Solution Product Brief. https://www.intel.es/content/dam/www/public/us/en/documents/platform-briefs/go-automated-accelerated-product-brief.pdf.

[2] *Kalray MPPA 256 Many-Core Processor, http://www.kalray.eu/products/mppa-manycore,*.

[3] *NanoC: http://www.nanoc-project.eu*.

[4] ARINC Inc. *ARINC Specification 653: Avionics Application Software Standard Standard Interface, Part 1 and 4, Subset Services*, June 2012.

[5] A. Bender. MILP based task mapping for heterogeneous multiprocessor systems. In *Design Automation Conference, 1996, with EURO-VHDL '96 and Exhibition, Proceedings EURO-DAC '96, European*, pages 190–197, Sep 1996.

[6] A. Burns et al. A wormhole NoC protocol for mixed criticality systems. In *2014 IEEE Real-Time Systems Symposium*, pages 184–195, Dec 2014.

[7] C.-L. Chou and R. Marculescu. Contention-aware application mapping for network-on-chip communication architectures. In *2008 IEEE International Conference on Computer Design*, pages 164–169, Oct 2008.

[8] D. Crupnicoff et al. *Deploying Quality of Service and Congestion Control in InfiniBand-based Data Center Networks*. Mellanox Technologies, 2005.

[9] D. Dasari et al. Response time analysis of cots-based multicores considering the contention on the shared memory bus. In *IEEE TrustCom*, 2011.

[10] R. Ginosar et al. RC64: High performance rad-hard manycore. In *2016 IEEE Aerospace Conference*, pages 1–9, March 2016.

[11] M. Girone. Computing Challenges at the Large Hadron Collider (LHC). *Keynote at the European Network on High Performance and Embedded Architecture and Compilation (HiPEAC) Conference 2018*, 2018.

[12] K. Goossens et al. Aethereal network on chip: concepts, architectures, and implementations. *IEEE Design Test of Computers*, 22(5):414–421, Sept 2005.

[13] L. S. Indrusiak et al. Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 47–56, July 2015.

[14] L. S. Indrusiak et al. Buffer-aware bounds to multi-point progressive blocking in priority-preemptive nocs. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 219–224, March 2018.

[15] J. Jalle et al. Deconstructing bus access control policies for real-time multicores. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 31–38, June 2013.

[16] H. Kashif and H. Patel. Buffer space allocation for real-time priority-aware networks. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–12, April 2016.

[17] C. Lee et al. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of 30th Annual International Symposium on Microarchitecture*, pages 330–335, Dec 1997.

[18] M. Lodde and J. Flich. An NoC and cache hierarchy substrate to address effective virtualization and fault-tolerance. In *2013 Seventh IEEE/ACM International Symposium on Networks-on-Chip (NoCS), Tempe, AZ, USA, April 21-24, 2013*, pages 1–8, 2013.

[19] A. Mejia et al. Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori. In *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, pages 10 pp.–, April 2006.

[20] M. Millberg et al. The nostrum backbone-a communication protocol stack for networks on chip. In *17th International Conference on VLSI Design. Proceedings.*, pages 693–696, 2004.

[21] B. Nikolic et al. Are virtual channels the bottleneck of priority-aware wormhole-switched noc-based many-cores? In *21st International Conference on Real-Time Networks and Systems, RTNS 2013, Sophia Antipolis, France, October 17-18, 2013*, pages 13–22, 2013.

[22] J. Nowotsch et al. Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement. In *ECRTS*, 2014.

[23] R. Obermaisser et al. The time-triggered system-on-a-chip architecture. In *2008 IEEE International Symposium on Industrial Electronics*, pages 1941–1947, June 2008.

[24] M. Panic et al. Parallel many-core avionics systems. In *2014 International Conference on Embedded Software (EMSOFT)*, pages 1–10, Oct 2014.

[25] M. Panic et al. Improving performance guarantees in wormhole mesh NoC designs. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1485–1488, March 2016.

[26] M. Panic et al. Modeling high-performance wormhole NoCs for critical real-time embedded systems. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–12, April 2016.

[27] H. Park and K. Choi. Position-based weighted round-robin arbitration for equality of service in many-core network-on-chips. In *Proceedings of the Fifth International Workshop on Network on Chip Architectures*, NoCArc '12, pages 51–56, New York, NY, USA, 2012. ACM.

[28] M. Paulitsch et al. Mixed-criticality embedded systems - A balance ensuring partitioning and performance. In *2015 Euromicro Conference on Digital System Design, DSD 2015, Madeira, Portugal, August 26-28, 2015*, pages 453–461, 2015.

[29] J. Poovey. *Characterization of the EEMBC Benchmark Suite*. North Carolina State University, 2007.

[30] A. Psarras et al. PhaseNoC: TDM scheduling at the virtual-channel level for efficient network traffic isolation. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1090–1095, March 2015.

[31] D. Rahmati et al. Computing accurate performance bounds for best effort networks-on-chip. *IEEE Transactions on Computers*, 62(3):452–467, March 2013.

[32] S. Ramos and T. Hoefler. Capability models for manycore memory systems: A case-study with Xeon Phi KNL. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 297–306, May 2017.

[33] S. Royuela et al. OpenMP tasking model for ada: Safety and correctness. In J. Blieberger and M. Bader, editors, *Reliable Software Technologies – Ada-Europe 2017*, pages 184–200, Cham, 2017. Springer International Publishing.

[34] P. K. Sahu and S. Chattopadhyay. A survey on application mapping strategies for network-on-chip design. *Journal of Systems Architecture*, 59(1):60 – 76, 2013.

[35] M. Schoeberl et al. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pages 152–160, May 2012.

[36] M. Shekhar et al. Network-on-chip aware scheduling of hard-real-time tasks. In *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pages 141–150, June 2014.

[37] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, pages 161–170, April 2008.

[38] Z. Shi and A. Burns. Real-time communication analysis with a priority share policy in on-chip networks. pages 3–12, July 2009.

[39] SoCLib. -, 2003-2012. http://www.soclib.fr/trac/dev.

[40] J. Sparsø. Design of networks-on-chip for real-time multi-processor systems-on-chip. In *2012 12th International Conference on Application of Concurrency to System Design*, pages 1–5, June 2012.

[41] Tilera. *TILE-Gx Processors Family http://www.tilera.com/products/TILE-Gx.php*.

[42] S. Tobuschat and R. Ernst. Real-time communication analysis for networks-on-chip with backpressure. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 590–595, March 2017.

[43] Wassel et al. SurfNoC: A low latency and provably non-interfering approach to secure networks-on-chip. *SIGARCH Comput. Archit. News*, 41(3):583–594, June 2013.

[44] C. Watkins and R. Walter. Transitioning from federated avionics architectures to Integrated Modular Avionics. *In Proceedings of 26th Digital Avionics Systems Conference. DASC '07*, 2007.

[45] Q. Xiong et al. Real-time analysis for wormhole NoC: Revisited and revised. In *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, pages 75–80, May 2016.

[46] L. Yang et al. Task mapping on SMART NoC: Contention matters, not the distance. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2017.

[47] H. Yu et al. Communication-aware application mapping and scheduling for NoC-based MPSoCs. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 3232–3235, May 2010.

[48] C. Zimmer and F. Mueller. Low contention mapping of real-time tasks onto TilePro 64 core processors. In *2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium*, pages 131–140, April 2012.