Improving Performance Guarantees in Wormhole Mesh NoC Designs

Miloš Panić*,†, Carles Hernandez†, Jaume Abella†, Antoni Roca Perez*, Eduardo Quiñones†, Francisco J. Cazorla†,‡

*Universitat Politècnica de Catalunya †Barcelona Supercomputing Center ‡Spanish National Research Council (IIIA-CSIC)

***Abstract*—Wormhole-based mesh Networks-on-Chip (wNoC) are deployed in high-performance many-core processors due to their physical scalability and low-cost. Delivering tight and time composable Worst-Case Execution Time (WCET) estimates for applications as needed in safety-critical real-time embedded systems is challenged by wNoCs due to their distributed nature. We propose a bandwidth control mechanism for wNoCs that enables the computation of tight time-composable WCET estimates with low average performance degradation and high scalability. Our evaluation with the EEMBC automotive suite and an industrial real-time parallel avionics application confirms so.**

I. Introduction

Critical Real Time Embedded Systems (CRTES) industry is gradually shifting towards multi- and many-core processors to satisfy the guaranteed performance needs of complex safety-related functions. This transitions challenges the derivation of time-composable Worst-Case Execution Time (WCET) estimates, i.e. bounds to the execution times of tasks that are independent of the load that co-running tasks put on shared resources. Time-composable WCET estimates simplify incremental certification [9] by allowing each system component to be subject to formal timing certification in isolation and independently from other components.

From an end-user perspective, the deployment of manycores in CRTES requires following properties:
– *UserReq1*: Manycores should facilitate deriving tight WCETs so that high guaranteed performance is provided;
– *UserReq2*: Manycores must facilitate deriving time composable WCETs to enable incremental certification;
– *UserReq3*: Manycores should also provide high average performance for some applications;
– *UserReq4*: Manycores for real-time should use technology as close as possible to COTS (high-performance) technology to ease their adoption. The low manycore demand of safety-critical real-time systems, w.r.t. the mainstream market, calls for reducing the need for customized real-time technology.

Network-on-Chip (NoC) is one of the manycore shared resources with the highest impact on average performance and WCET. Hence, the four user requirements on manycore designs are to be captured and fulfilled by NoC designs. We consider wormhole mesh NoC (wNoC) as a candidate NoC solution as it is widely accepted in the high-performance market due to its physical scalability and low cost [28][23].

The high-performance requirements imposed by some applications (*UserReq3*) require wNoCs to provide high throughput. This requirement is fulfilled since wNoCs are designed for high-performance systems. Meanwhile, *UserReq2* for real-time applications requires time-composable worst-case traversal time (WCTT), i.e. WCTT not affected by the load contender tasks put on the wNoC. wNoCs can meet this by using time-analyzable arbitration policies [17][10] and applying the model in [21].

Contribution. We show that current wNoCs fail to achieve tight WCTT (*UserReq1*), which negates their benefits. In particular we show that (i) WCTT values derived for current wNoCs poorly scale with network size – even for small networks; and (ii) the WCTT derived for a task depends on the maximum allowed packet size and poorly scales with it. Further, current wNoCs do not necessarily impose a limit on the packet size and leave that to the protocol on top of the network (e.g. AMBA [1]).

We propose a new time-composable wNoC design relying on concepts developed for high-performance wNoCs, hence achieving *UserReq4*. Our design focuses on controlling the network bandwidth (the main factor affecting *WCTT*) to provide a fair guaranteed bandwidth distribution across the different communication flows. Bandwidth control is exercised at two levels. At local level, we ensure fairness by providing a *WCTT-aware Packetization* (WaP) that makes real-time guarantees independent of contenders packet size. At global level, we provide fairness across contenders by performing a *WCTT-aware Weighted* (WaW) round-robin arbitration.

We evaluate $WaW + WaP$ on a 64-core manycore architecture with cores accessing memory controllers through a wNoC. We use EEMBC [20] autobench and an avionics real-time parallel application provided by Honeywell [16]. We show that our design significantly decreases WCET estimates for the parallel application by a factor of $4.8\times$ to $9.5\times$ depending on the number of flits per packet. For single-threaded applications WCET decreases by $230\times$ on average across all cores and by $1.4\times$ for the 25% best set of cores for the baseline NoC.

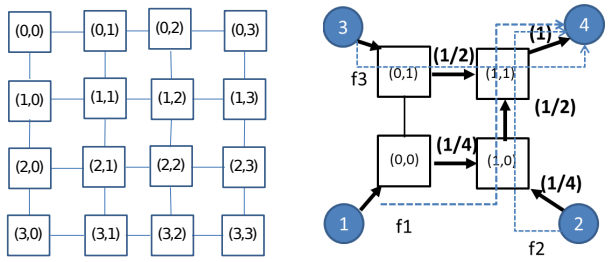
II. Wormhole-based mesh NoCs

Deriving WCET estimates in manycores relies on bounding access times to shared hardware resources [19][6]. In the case of NoCs this translates into i) bounded WCTT such that every request sent to the NoC has a service time, i.e. traversal time, boundable at analysis; and ii) time-composable WCTT such that the bound to the traversal time derived for the request of a task does not depend on the load put by other co-running tasks on the NoC. Low WCTT translates into tighter WCET estimates, which allows increasing the guaranteed performance that the manycore chip can provide.

A. Assumptions

We assume a canonical 2D-mesh [5] with wormhole switching and XY routing policies (Figure 1(a)). The need for time-composable WCTT prevents making assumptions about the number and load of crossing flows. Time-composable WCET estimates provide a drastic reduction of development costs as

The research leading to these results is funded by the European Union Seventh Framework Programme under grant agreement no. 287519 (parMERASA) and by the Ministry of Science and Technology of Spain under contract TIN2012-34557. Miloš Panić is funded by the Spanish Ministry of Education under the FPU grant FPU12/05966. Carles Hernández is jointly funded by the Spanish Ministry of Economy and Competitiveness and FEDER funds through grant TIN2014-60404-JIN. Jaume Abella is partially supported by the Ministry of Economy and Competitiveness under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717.



(a) (b)

Fig. 1. (a) Router coordinates in a 4x4-Mesh. (b) Unfair bandwidth allocation in wormhole.

each subsystem can be independently developed and certified and incrementally integrated. These benefits pay off the increase in WCET caused to achieve time composability. Instead, we assume the worst-case of the wNoC state and load:

– (1) Every node in the network is able to send and receive packets to/from any other node in the network.

– (2) Every time we inject a packet in the NoC, any possible contending flow is also sending packets creating a worst possible contention scenario, i.e. for a packet of a given flow at every hop all possible contenders (i.e. all possible flows partially sharing the path) are also requesting the same output port.

– (3) Packets contending for an output port are arbitrated using a time-analyzable policy – round-robin in our case [10], which is already used in some existing mesh wNoCs [28][23].

– (4) Maximum allowed packet size in the network is known. We assume that packets of contending requests have maximum size when deriving WCTT bounds.

– (5) Finally, it is also required assuming that the network is congested by the time packets are injected in the network.

B. Factors impacting WCTT estimates

There are two main aspects affecting real-time guarantees that we address with our design:

Message size impact on arbitration slot duration. In wNoCs only the header flit of a packet is arbitrated. This implies that *the time that requests in a given router wait to be arbitrated depends on the size of the particular requests contending for the same output port*. Hence, deriving time-composable WCTT values requires considering that all possible requests (i.e. the number of router ports minus one) can contend for the same output port and the size of all requests is the maximum allowed size. However, some wNoCs do not impose any limit on packet size, enabling undefined-length requests [1]. Even with the maximum packet size limited, different lengths of packets penalize real-time guarantees, since we have to consider that contending requests have the maximum size.

Unfair bandwidth allocation. In a network where all flows may contend for the same resource, WCTT mainly depends on the flow's allocated bandwidth. The latency in a congested system can be approximated as $1/(bandwidth)$ [5]. Despite round-robin arbitration ensures a fair distribution of resources when it is used in a centralized way, *round-robin fails to fairly share the bandwidth in distributed networks*. For example, when round-robin is used in an on-chip bus, it distributes the bandwidth amongst the cores accessing the bus fairly. However, when a request passes through several chained routers to reach a given node, the bandwidth allocated to this request is not the same as the one allocated to a closer or farther request. In the evaluation section we show how the unfair bandwidth allocation translates into bad (high) WCTT values.

III. Flit-Homogeneous Guarantees in Meshes

We present a new wNoC design that performs a flit-level fair distribution of guaranteed bandwidth to achieve time-composable and tight WCTT. Our proposal requires minimum modifications to regular mesh designs in the packet generation (*local fairness*) and in the packet arbitration (*global fairness*).

WaP. Packet length has high impact on the maximum contention a request can suffer. If the wNoC is not carefully designed this could lead to an unbounded WCTT. For wNoCs the arbitration slot duration is equal to the packet size, the larger the packet is, the longer the time slot is. To avoid having arbitration slots of different duration we use WaP that forces all packets to have the size of the smallest packet in the system. This is achieved by slicing a request into one or more minimum size packets at the network interface (NIC).

When a request (Req_i) arrives at the NIC a regular packetization scheme creates a single packet that is injected in the network. With WaP the request payload is sliced in minimum sized packets and header info is replicated. WaP improves NoC WCTT as the size of contending packets is bounded to the minimum size packet. For instance, with a regular packetization scheme, the worst-case latency for a S-flit packet for reaching an output port to which 4 different input ports are contending is $3*L+S$ where L is the maximum allowed size of packets in the network. Instead, with WaP, for a minimum packet size of m, the worst-case latency is $3*m+m$. Note that maximum allowed packet size in the network (L) is much larger than minimum size packets (m) that commonly consist of one-flit.

WaP penalizes the effective bandwidth due to the overhead of the required routing and control information (that can be significant in a manycore). In Section IV we evaluate WaP in terms of both average and worst-case performance.

WaW. WaW relies on a weighted round-robin arbitration scheme [18] to enable a globally fair link bandwidth distribution that balances the WCTT off all nodes in the NoC. Weighted round-robin uses weights to assign the rotating priorities to contending input ports. WaW uses arbitration weights per router input port that balance WCTT in all nodes of the router. It set weights by accounting for the number of contending flows coming through a given input port and the total number of flows traversing the requested output port. These numbers are determined by the routing algorithm [5].

Let $I_{dir_i}(i,j)$ be the number of communication flows traversing the dir_i input port of router $R(i,j)$ – dir_i can be any of the possible mesh router port directions $X+$,$X-$, $Y+$, $Y-$, or PME. Let $O_{dir_o}(i,j)$ be the number of flows traversing the dir_o output port of $R(i,j)$. WaW per-input/output port pair arbitration weights $W(I_{dir_i}, O_{dir_o})$ can be computed for any of the possible input/output port combinations at $R(i,j)$ using the following equations:

$$I_{X+} = x \qquad O_{X+} = x+1$$
$$I_{X-} = N-x \qquad O_{X-} = N-x+1$$
$$I_{Y+} = N*y \qquad O_{Y+} = N*(y+1)$$
$$I_{Y-} = N*(M-y-1) \qquad O_{Y-} = N*(M-y)$$
$$I_{PME} = 1 \qquad O_{PME} = N*M-1$$

N and M are the horizontal and vertical dimensions of the network, respectively, while x and y stand for the horizontal and vertical coordinates of the node under analysis. For the $X+$ input port the number of flows coming through it corresponds to the x coordinate i.e. the number of nodes that precede the actual node in the same row. Note that with XY routing, packets in the Y direction cannot be forwarded to the X direction.

TABLE I
ARBITRATION WEIGHTS FOR A 2X2-MESH ROUTER R(1,1) IN A REGULAR MESH AND WITH WaW

	Regular Mesh	Weighted Mesh
W(PME,X-)	1	1
W(PME,Y-)	0.5	0.5
W(X-,PME)	0.5	0.33
W(X-,Y-)	0.5	0.5
W(Y-,PME)	0.5	0.66

Therefore, the flows accessing an X port are only the ones in the same row. On the contrary, flows crossing Y-direction ports may be originated at any of the preceding nodes in any row. Per direction router weights are derived using Equation 1.

$$W(I_{dir_i}, O_{dir_o}) = I_{dir_i}/O_{dir_o} \quad (1)$$

Let us illustrate how WaW works with the example from Figure 1(b). Let us consider all flows with destination node 4. At $R(1,1)$ only $X+$ and $Y+$ input ports can access the PME output port. $O_{PME} = 3$ as the flows originated at the 3 remaining nodes access node 4 using O_{PME}. For the input ports we have $I_{X+} = 1$ and $I_{Y+} = 2$. We consider that in this example $N = 2$ and $M = 2$ so $I_{Y+} = |2 * (1)| = 2$ and $I_{X+} = x = 1$. Table I shows $R(1,1)$ weights required to perform the weighted arbitration in the 2x2 mesh NoC and compares them with the default weights of the round-robin arbitration. The weight values range from 0 to 1 and represent the bandwidth that is allocated to a given input/output pair. For example, for the input ports requesting the PME output port the weighted arbitration assigns 1/3 of the bandwidth to the flows coming from X- and 2/3 of the bandwidth to the flows from Y-. Note that X- only serves one flow from node 3 to node 4 while Y- serves 2 flows (from nodes 1 and 2 to node 4). Instead, round-robin arbitration assigns always the same bandwidth (0.5) to any of the 2 input ports requesting a given output port, regardless of the number of potential flows using these input ports.

***WaW* implementation.** XY routing allows precomputing the weights and assigning them to input ports statically, as needed for WCET estimation. In our implementation, input port weight is measured as the number of flits it can transmit to an output port. When several input ports contend for an output port, the input port with the largest flit count wins, and decrements its flit count by one. If more than one contender has the largest flit count, a conventional round robin policy is used to arbitrate amongst them. Instead, when no input ports demand an output port, each input port flit count is incremented (if it is not larger than its weight). When an input port is the unique candidate to access an output port, its flit count is unaltered.

Hardware modifications. In order to increase compliance with COTS *wNoC* designs, WaW and WaP incur minimum local changes. Those changes can be implemented in regular wNoCs which could provide a feature to enable/disable them depending on the average and guaranteed requirements of the wNoC. This departs from other designs that might require changes in buffering, switches architecture, synchronization, etc., that would decrease the chance of adoption of our proposal.

NICs are already equipped with the logic to perform packetization of processor requests. Hence, WaP only requires the size of packets to be parametrizable from the software. Meanwhile WaW requires per-input port counters (no more complex than the ones required for regular round-robin arbitration) and an additional arbitration policy. Our results – obtained from the NoC area decomposition given in [24] – show that the area increase incurred in the NoC is below 5%.

TABLE II
WCTT VALUES FOR DIFFERENT MESH SIZES FOR 1-FLIT PACKETS.

	Regular Mesh			WaW + WaP		
NxM	max	mean	min	max	mean	min
2x2	14	10	6	11	9	8
3x3	123	39.16	9	32	24	17
4x4	1071	145.68	9	64	45	31
5x5	8895	568.14	9	108	72	49
6x6	72447	2375.85	9	163	105	71
7x7	584703	10632.53	9	230	144	97
8x8	4698111	50516.79	9	310	189	127

IV. EVALUATION

We use a cycle-accurate simulator based on SoCLib [3] with gNoCSim [2] integrated. We model a 64-core mesh-based processor (routers range from $R(0,0)$ to $R(7,7)$). In our manycore, load (and write-miss) requests comprise a one-flit message from the core to memory. Given that cache line size is 64-bytes and we need 16-bits for control data (512+16 bits), memory answers with 4-flit messages over 132-bit wide links. Evicted line requests require a 4-flit message and a one-flit answer. $Waw + WaP$ adds control data to each flit, therefore requiring an extra flit, so 5 instead of 4 (512+5*16 bits over a 132-bit wide channel), leading to 25% overhead.

WCTT. Table II shows average, max, and min WCTT values for the regular wNoC and WaW+WaP across several network sizes. While regular mesh designs obtain always the lowest WCTT values (for the nodes that are directly attached to destination) our proposal achieves significantly better WCTT values for the majority of the network flows (as shown by the average WCTT results). For instance, for the 64-node NoC the minimum WCTT with regular meshes is 9 and with WaW+WaP is 127 cycles, while the maximum value decreases from above 4 million cycles to 310 (a decrease of 4 orders of magnitude). On average the WCTT for the original NoC is above 50,000 cycles (largely above our design, 189).

WCET estimates for EEMBC. Our simulation architecture supports the *WCET computation mode* [17], in which at analysis time, requests accessing the NoC are artificially delayed by an *upper bound delay* (UBD). During operation, WCET computation mode is disabled and NoC requests suffer only actual delays, which are safely upper-bounded by UBD.

In Table III each cell represents a node of a $8x8$ *wNoC*. All nodes communicate to the memory connected to the top-left node $R(0,0)$. Each cell shows the WCET of WaW+WaP normalized w.r.t. a regular *wNoC*. In particular we show the average reduction across all (single-threaded) EEMBC Automotive benchmarks. Values above 1 show that WaW+WaP provides higher WCET estimates than a regular *wNoC* and vice versa. We observe that WCET values for nodes close to $R(0,0)$ are slightly higher than for the regular wNoC. In particular 11 nodes present WCET values worse than the ones provided by a regular *wNoC* with a maximum slowdown of up to $1.5\times$ for the best situated node. However, on the other 53 nodes, average WCET estimates are significantly higher (worse) with the regular wNoC than with WaW+WaP. In some cases, as shown in Table III, the difference is 3-4 orders of magnitude, i.e. the WCET obtained with WaW+WaP is only 0.002 of that with the regular wNoC.

WCET estimates for Parallel Applications. We also evaluate $WaW+WaP$ using 3D path planning (3DPP), an industrial avionics parallel application provided by Honeywell [16]. 3DPP uses 16 cores to guide an aircraft through the obstacle map represented as a 3D matrix. In the $8x8$ wNoC we run 3DPP under four different placements (see Figure 2(b)).

With focus on P0, Figure 2(a) shows the WCET estimates

TABLE III
NORMALIZED WCET PER CORE OF EEMBC WITH WaW+WaP

Y-axis position \ X-axis position	0	1	2	3	4	5	6	7
0	1.4841	1.4841	1.4920	1.4387	1.3046	1.0850	0.8131	0.7292
1	1.3609	1.3806	1.2843	1.0899	0.8262	0.5575	0.3427	0.3260
2	1.2454	1.0856	0.8441	0.5777	0.3553	0.2027	0.1112	0.1226
3	0.9855	0.6078	0.3739	0.2123	0.1150	0.0609	0.0321	0.0428
4	0.6024	0.2304	0.1219	0.0634	0.0328	0.0169	0.0088	0.0145
5	0.2779	0.0692	0.0345	0.0174	0.0089	0.0046	0.0024	0.0049
6	0.1063	0.0189	0.0093	0.0046	0.0024	0.0012	0.0004	0.0016
7	0.0528	0.0067	0.0033	0.0016	0.0008	0.0004	0.0002	0.0008

for the regular and $WaW + WaP$ wNoC considering that the maximum packet size in the network is 1, 4 and 8 flits (labeled L1, L4 and L8 respectively). We observe the significant impact of $WaW + WaP$. Overall, it outperforms the regular wNoC for all packet sizes considered, with improvements ranging from 1.4X for L1 to 3.9x for L8.

For the L1 setup Figure 2(b) shows the impact of placement of the application. $WaW + WaP$ benefits are two-fold. It achieves lower WCET estimates (from 1.4x to 7x) than the regular wNoC and leads to smaller variability across placements (around 20% in our setup compared to over 6x with the regular NoC). This is of paramount importance in real-time systems to control the impact of placement, which has been shown as a first-order factor in the WCET [14].

Average performance. We have as well evaluated $WaW + WaP$ and regular wNoC in terms of average performance. Results show that $WaW + WaP$ incurs negligible average performance degradation (less than 1%) for both single-threaded and parallel applications. The origin of the degradation resides in the overhead introduced by packetization that is minimized as it only affects those packets having more than one flit.

V. RELATED WORK

Customized NoCs for real-time such as TDMA-based or time-triggered ones will find difficulties in being adopted by the real-time industry [27] since their implementation incurs high non-recurrent costs. This is the case for [25], [7], [15], [13].

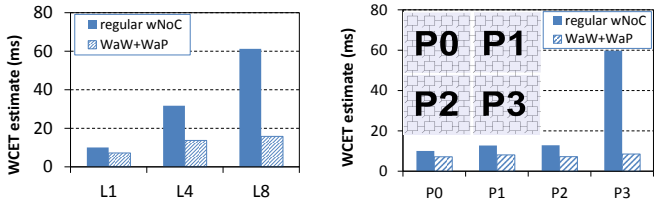
In best-effort wNoCs the use of virtual channel prioritization has been proposed as an effective way to provide tight latency bounds [26] and [22]. The same logic applies to [11], where authors provide bandwidth guarantees for GS connections per port. However provided guarantees require a detailed knowledge of the applications/tasks that will run in the final system and thus, fail to satisfy incremental certification requirements.

In [12], [21] authors provide realistic bounds for wNoCs without using flit-level virtual channel preemption. The model in [21] requires knowing all communication flows integrated in the system to derive safe upper-bounds, making those bounds not time-composable. Interference-free NoC designs using wormhole-based NoC designs have been recently proposed in [4] and [8]. While [4] shows lower best-effort traffic degradation than [8] by smartly multiplexing virtual channels, the degradation of best-effort traffic performance is significant.

We follow a different approach to fulfill hard-real time requirements by deriving time-composable WCTT bounds in wNoCs without sacrificing average performance. Further, we address the scalability problems of latency bounds in wNoCs by proposing a mesh design that significantly improves default mesh WCTT values with low hardware complexity.

VI. CONCLUSIONS

The use of wormhole-based NoCs in the context of CRTES applications complicates the timing analysis of applications,



(a) Regular wNoC vs. WaW+WaP (b) Impact of Allocation

Fig. 2. WCET estimates for the 16-core parallel avionics application

making the WCET estimates of those applications rapidly increase with the network size. The latency bounds achieved by our design are scalable. Our proposal enables a fair sharing of the available bandwidth across the different flows in the network. This makes time-composable WCET estimates less affected by the core count in the manycore. Our results with benchmarks and a real application confirm that the proposed mesh achieves tight and uniform scalable WCET values with negligible average performance degradation. Furthermore, hardware modifications required for the proposed design w.r.t. regular mesh designs are few, easing its adoption.

REFERENCES

[1] *ARM AMBA 3 AXI Specification (available at http://www.arm.com/products/solutions/axi_spec.html).*
[2] *NanoC: NaNoC design platform. http://www.nanoc-project.eu.*
[3] Soclib, http://www.soclib.fr/trac/dev, 2012.
[4] A.Psarras, et al. Phase-noc: Tdm scheduling at the virtual-channel level for efficient network traffic isolation. DATE 2015.
[5] J. Duato, et al. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann, 2002.
[6] GENESYS. http://www.genesys-platform.eu.
[7] K. Goossens, et al. Aethereal network on chip: concepts, architectures, and implementations. *Design Test of Computers, IEEE*, 2005.
[8] H. M. G. 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.
[9] R. Inc. RTCA DO-297 integrated modular avionics (IMA) development guidance and certification considerations. 2005.
[10] J. Jalle, et al. Deconstructing bus access control policies for real-time multicores. In *SIES*, June 2013.
[11] T. Kranich and M. Berekovic. Noc switch with credit based guaranteed service support qualified for GALS systems. In *DSD*, 2010.
[12] S. Lee. Real-time wormhole channels. *Journal Of Parallel And Distributed Computing*, 63:299–311, 2003.
[13] M. Millberg, et al. The nostrum backbone-a communication protocol stack for networks on chip. In *IEEE VLSI Design*, 2004.
[14] J. Mische and T. Ungerer. Guaranteed service independent of the task placement in nocs with torus topology. In *RTNS*, 2014.
[15] R. Obermaisser, et al. The time-triggered system-on-a-chip architecture. In *ISIE*, 2008.
[16] M. Panic, et al. Parallel many-core avionics systems. *EMSOFT*, 2014.
[17] M. Paolieri, et al. Hardware support for WCET analysis of Hard Real-Time Multicore Systems. In *ISCA*, 2009.
[18] H. Park and K. Choi. Position-based weighted round-robin arbitration for equality of service in many-core network-on-chips. NoCArc, 2012.
[19] parMERASA. *EU-FP7 Project:http://www.parmerasa.eu/.*
[20] J. Poovey. *Characterization of the EEMBC Benchmark Suite*. North Carolina State University, 2007.
[21] D. Rahmati, et al. Computing accurate performance bounds for best effort networks-on-chip. *IEEE Transactions on Computers*, 62(3), 2013.
[22] E. A. Rambo and R. Ernst. Worst-case communication time analysis of networks-on-chip with shared virtual channels. DATE, 2015.
[23] J. Rattner. *Single-chip Cloud Computer: An experimental many-core processor from Intel Labs*.
[24] A. Roca. *Floorplan-Aware High Performance NoC Design*. PhD thesis, Universitat Politecnica de Valencia, 2012.
[25] M. Schoeberl, et al. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *NoCS*, 2012.
[26] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *NoCS*, 2008.
[27] J. Sparsoe. Design of networks-on-chip for real-time multi-processor systems-on-chip. In *Application of Concurrency to System Design (ACSD), 2012 12th International Conference on*, pages 1–5, 2012.
[28] Tilera. *TILE-Gx Family http://www.tilera.com/products/TILE-Gx.php*.