Designing Regular Network-on-Chip Topologies under Technology, Architecture and Software Constraints

F.Gilabert[‡], D.Ludovici[§], S.Medardoni[†], D.Bertozzi[†], L.Benini[††], G.N.Gaydadjiev[§]
[†] ENDIF, University of Ferrara, 44100 Ferrara, Italy.
[††] DEIS, University of Bologna, 40136 Bologna, Italy.
[‡] Dept. of Computer Engineering, Universidad Politecnica de Valencia, Spain.
[§] Computer Engineering Lab., Delft University of Technology, The Netherlands.

***Abstract*—Regular multi-core processors are appearing in the embedded system market as high performance software programmable solutions. The use of regular interconnect fabrics for them allows fast design time, ease of routing, predictability of electrical parameters and good scalability. k-ary n-mesh topologies are candidate solutions for these systems, borrowed from the domain of off-chip interconnection networks. However, the on-chip integration has to deal with unique challenges at different levels of abstraction. From a technology viewpoint, interconnect reverse scaling causes critical paths to go across global links. Poor interconnect performance might also impact IP core speed depending on the synchronization mechanism at the interface. Finally, this might also conflict with the requirements that communication libraries employed in the MPSoC domain pose on the underlying interconnect fabric. This paper provides a comprehensive overview of these topics, by characterizing physical feasibility of representative k-ary n-mesh topologies and by providing silicon-aware system-level performance figures.**

I. Introduction

The execution of many multimedia and signal processing functions has been historically accelerated by means of specialized processing engines [1]. With the advent of multi-processor system-on-chip (MPSoC) technology, performance of hardware accelerators is becoming accessible by combining multiple programmable processor tiles within a multicore system [2]. In addition, the performance of latest application specific integrated processors (ASIPs) [4] together with the high availability of transistors is making the design of custom hard-wired logic always less convenient (time-to-market, respin risks). The underlying principle is that efficient computation can be achieved while only marginally impacting programmability and/or configurability, and architectures can be devised that address the computation requirements of an entire application domain [3].

In this context, tile-based architectures cope effectively with the productivity gap, in that they provide parallelism through the replication of many identical blocks placed each in a tile of a regular array fabric [3], [6], [7]. This approach makes performance scalability more a matter of instantiation and connectivity capability rather than architecture complexity.

Perhaps the most daunting challenge to make MPSoC technology mainstream is to realize the enormous bandwidth capacities and stringent latency requirements when interconnecting a large number of processing cores. This task is on burden of the global intrachip communication infrastructure. Networks-on-chip (NoCs) are generally believed to be the long term solution to the communication scalability issue [8].

Topology selection is a NoC design issue which needs to be addressed in the early design stages and which has deep implications both on final system performance and on physical network feasibility. NoC architectures can be designed with both regular and custom topologies. The primary advantages of a regular NoC architecture are topology reuse, reduced design time, ease of routing, better control of electrical parameters and hence less design respins and a higher degree of performance predictability. The 2D mesh is currently the most popular regular topology used for on-chip networks in tile-based architectures, because it perfectly matches the 2D silicon surface. Unfortunately, 2D meshes show very poor scalability properties in terms of diameter, average minimal hop count and bisection bandwidth.

In contrast, topologies with more than 2 dimensions are attractive for a number of reasons. First, increasing the number of dimensions in a mesh results in higher bandwidth and reduced latency. Second, the number of dimensions can be traded-off with the number of cores per switch, thus giving rise to concentrated topologies saving network components and trading bandwidth for latency. Third, wiring on a chip comes at a lower cost with respect to off-chip interconnections. However, wiring is also the challenging aspect of these topologies, since their mapping on a bidimensional plane involves the existence of wires with different lengths. Depending on the physical design technique, the more complex connectivity pattern may impact performance, area and power in different ways, such as a decreased operating frequency or a higher link latency.

The objective of this paper is to assess performance of k-ary n-mesh topologies while considering design constraints posed by real-life HW/SW MPSoC platforms. This makes the analysis more insightful and trustworthy than traditional abstract exploration frameworks based on pencil-and-paper floorplanning considerations. These latter often ignore the presence of non-routable hard IP blocks, the asymmetric tile size, the use of link pipelining to sustain network speed or the dependence of switch critical path on its radix. The relentless scaling of silicon technology to the nanoscale regime is making the interconnect delay issue even more critical and is causing the network critical path to move from the logic to global network links. By leveraging a backend synthesis flow for regular NoC architectures, we characterize the mapping efficiency of a given topology on the silicon layout, targeting a 65nm technology node.

We extract physical parameters from the physical synthesis

This work was supported by the GALAXY European Project (FP7-ICT-214364), by the Hipeac Network of Excellence (Interconnect Cluster), by the Spanish MEC under Grant TIN2006- 15516-C04-01 and by CONSOLIDER–INGENIO 2010 under Grant CSD2006-00046.

978-0-7695-3575-3/09 $25.00 © 2009 IEEE
DOI 10.1109/CISIS.2009.30

and expose them to the system-level simulation tool, thus coming up with silicon-aware performance figures. At this level, network performance is usually characterized by means of synthetic traffic patterns (such as uniform or hot-spot), in the best case reflecting average communication bandwidth of real-life applications. This paper aims to go a step further, by considering the requirements that a recently proposed middleware library for MPSoC communication poses on the underlying interconnect fabric. In essence, we consider network traffic generated by synchronization mechanisms implemented in software. Finally, we provide a model of the chip I/O interface, thus capturing the implications of I/O performance on that of specific k-ary n-mesh topologies.

II. Previous work

Although widely used across a number of network on chip (NoC) designs [3], [11], the 2D-mesh NoC topology lacks of scalability and tends to concentrate traffic in the center nodes [10]. This has motivated works in the open literature that come up with optimized NoC topologies while keeping regularity properties as much as possible. A novel interconnect topology called spidergon was proposed in [12], where each core is connected to the clockwise, counterclockwise and diagonal node. A traditional wormhole-routed mesh augmented by a hierarchical ring interconnect for routing global traffic is illustrated in [13]. NOVA is a hybrid interconnect topology targeted at an FPGA, and is compared in [14] with star, torus and hypercube topologies. Gilabert et al. propose in [15] to use high-dimensional topologies, using different metal layers to soft long link delay and trading-off dimensions with the number of cores per router. The work in [10] proposes a concentrated mesh architecture with replicated subnetworks and express channels.

Topology exploration is an active research area due to the large scale of on-chip networks and to the feasibility challenges posed by nanoscale technologies. An effort to compare mesh and torus topologies under different routing algorithms and traffic models with respect to their performance and power consumption is described in [17]. Theoretical uniform traffic based on the request/reply paradigm is used to assess ring, 2D-mesh, spidergon and unbuffered crossbar topologies in [18]. [19] claims that from an energy standpoint, high-dimensional tori should never be selected over hierarchical or express cubes.

As technology scales to the nanometer regime, topology analysis and exploration needs to be performed with tools that account for the effects of nanoscale physics, largely impacting final performance and even feasibility of many NoC topologies. A general guideline driving network-on-chip (NoC) design under severe technology constraints consists of silicon-aware decision-making at each hierarchical level [21]. This is likely to result in less design re-spins and in faster timing closure. In this direction, new tools are emerging that guide designers towards a subset of most suitable candidates for on-chip network designs while considering the complex tradeoffs between applications, architectures and technologies [22], [23].

Our previous work in [24] presents silicon-aware topology analysis for a network with 16 nodes. A transaction level simulation environment is presented that explores the implementation space of k-ary n-mesh topologies and provides guidelines and constraints for the physical synthesis. This paper significantly extends the work in [24], while keeping the same layout-aware approach. First, the emphasis here is not on capturing the sensitivity of system performance to physical parameters, but rather on the assessment of such parameters for selected k-ary n-mesh topologies when designed for maximum performance. Second, the analysis of topology relative performance and area is extended to 64 node networks as well, thus providing a scalability analysis. Third, we point out the key impact of specific physical design techniques on the performance-cost trade-off of topologies under test.

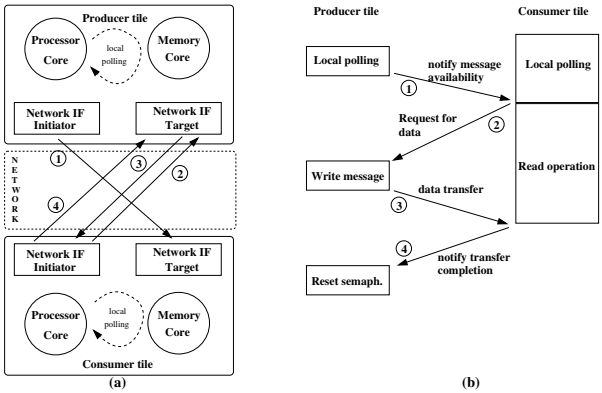

Fig. 1. Tile abstraction and mapping of producer-consumer communication handshake on network transactions.

III. Topology exploration framework

A. NoC architecture

Our realistic topology exploration framework utilizes the the xpipesLite NoC architecture [5]. The switching fabric implements a 2-cycle-latency (one for switch operation and one for traversing the output link), output-queued wormhole-switched router supporting round-robin arbitration on each output port. The implemented flow-control scheme is stall/go [25].

The switch is parameterizable in the number of its inputs and outputs, its link width as well as in the size of the output buffering. For this work, 6-flit buffers are assumed and the link (and flit) width is set to 32 bits.

The network interface (NI) is designed as a bridge between an OCP [1] interface and the NoC switching fabric. Its purposes are the synchronization between OCP and network timing, (de-) packetization, the computation of routing information (stored in a Look-Up Table, LUT) and flit buffering to improve performance. The NI performs clock domain crossing, however in order to keep the architecture simple the ratio between network and core clock frequencies needs to be an integer divider.

[1]Open Core Protocol – standard end-to-end communication protocol

	16 tiles			64 tiles		
Topology	4-ary 2-mesh	2-ary 4-mesh	2-ary 2-mesh	8-ary 2-mesh	2-ary 6-mesh	2-ary 4-mesh
Max Arity	6	6	10	6	8	10
Total Switches	16	16	8	64	64	16
Tiles x Switch	1	1	4	1	1	4
Total Ports	80	96	40	352	384	192
Bisection Cut	4	8	2	8	32	8
Ideal Diameter	6	4	3	14	6	4

TABLE I
Topologies under test

Both switches and network interfaces were originally modeled in SystemC as synthesizable RTL-equivalent models. However, conducting system level performance analysis with RTL simulation would imply unaffordable simulation times and resources. For this reason, in [16] we presented transaction-level (TL) models abstracting all relevant mechanisms of the xpipesLite architecture (retiming, buffering, arbitration, flow control, switching, synchronization), including injection and ejection interfaces. We proved an accuracy of TL simulation within 0.03% of RTL simulation while achieving one order of magnitude faster simulation speeds. A 256 core system with 16 millions of OCP read burst transactions can be simulated in a couple of hours.

Interestingly, the TL simulator can backannotate silicon-dependent parameters from the physical synthesis such as link latency and maximum operating frequency of NoC building blocks, thus making silicon-aware decision making viable even at the highest layers of the design hierarchy. Such parameters were extracted from post-layout analysis for 16 node topologies and projected for 64 node ones.

B. Topologies under test

The target of our analysis is a *tile based architecture* where each tile is assumed to include at least *one processor and one local memory core*. Therefore, the asymmetric tile size needs to be accounted for when laying out the topology: this puts traditional assumptions on mesh and hypercube wiring in discussion.

Meshes can be referred to as k-ary n-meshes. The topology has k^n routers in a regular n-dimensional grid with k switches in each dimension and links between nearest neighbors. Moreover, the n-hypercube topology is a particular case of a mesh where k is always 2. Also, each switch can have one or more tiles attached.

In this paper we devise topologies for two system scales, namely topologies for 16 tiles and topologies for 64 tiles. In the 16 tiles category, we analyze a 4-ary 2-mesh (referred to as 2D-mesh from now on) with one tile per switch, a 2-ary 4-mesh (4-hypercube) with one tile per switch, and a 2-ary 2-mesh with 4 tiles per switch. The 4-hypercube is a representative topology for those ones featuring a number of dimensions higher than 2, while the 2-ary 2-mesh illustrates the properties of $concentrated$ topologies, connecting more nodes to the same switch.

In the 64 tiles category, we analyze a 8-ary 2-mesh (also referred to as 2D-mesh) with one tile per switch, a 2-ary 6-mesh (6-hypercube) with one tile per switch and a 2-ary 4-mesh with 4 tiles per switch. These topologies were chosen with the same criteria as for the 16 node systems.

Table I shows some representative data for the studied topologies. Please note that even with 1 tile per switch, 2 switch input and 2 switch output ports are required to connect the tile, since it includes an initiator and a target NI, each with one input and one output port to the switch for receiving/sending data. The initiator NI uses the output port to send out packets and the input port to receive packets carrying read response data. The target NI uses the input port to receive packets carrying write data or read requests for the connected target. Read responses are packetized and sent out through the output port.

C. Backend synthesis flow

The practical feasibility of topologies under test was explored by means of a semi-automated design flow spanning from RTL description to layout-level verification. This enables us to explore and validate topologies down to the placement and routing steps, thus accounting for the effects of nanoscale technologies. The flow has been conceived for the physical synthesis of 2D-meshes and multi-dimension regular topologies. We use industrial tools for placement-aware logic synthesis and for place-&-route on an STMicroelectronics $65nm$ SVT technology optimized for low-power.

The first step of our backend synthesis flow is placement-aware logic synthesis through Synopsys Physical Compiler, applied to switch and network interface modules in isolation. This tool keeps optimizing the gate level netlist based on the expected placement and the wire loads it implies. The final resulting netlist considers placement-related effects and makes performance estimated at this level more trustworthy. Post-synthesis max. frequency however represents a theoretical upper bound, since the critical path is computed inside the modules and actual routing of switch-to-switch links will then cause a further unpredictable performance drop.

Floorplanning and place&route are performed with the Cadence SoC Encounter tool. Computation tiles are replaced by non-routable hard obstructions of size $2mm$ x $1mm$. At first, hard black boxes are manually placed on the floorplan. Fences are then defined to limit the area where the cells of each network-on-chip module can be placed. Subsequently, the tool automatically places cells without trespassing the fences. Fence size is devised based on the report of the Physical Compiler on floorplan cell area of each module, while fence position depends on the switch placement strategy. Our choice was to aim at uniform latency across wiring dimensions. As an example, the placement strategy for a 2-ary 4-mesh topology with 4 tiles per switch is illustrated in Fig. 2.

Subsequent steps include clock tree synthesis and power supply network insertion. Each IP core is assumed to be an independent clock domain with its own clock tree. In this work, we assume all IP cores to work at the same frequency, which depends on the network speed and on the divider applied to the frequency-ratioed clock domain crossing mechanism at the network interface. After the power nets have been routed, the tool begins to route the logic wires. After an initial mapping, search and repair loops are executed to fix any violations. As a final step, post-routing optimizations are performed, including crosstalk and antenna effect minimization. Finally, a signoff procedure can be run by using Synopsys PrimeTime to accurately validate the timing properties of the design.

D. Communication semantics

In our work, a transaction-level simulator of the xpipesLite NoC architecture is used for system-level performance analysis. The clock cycle accuracy with respect to RTL simulation is proved in [16], which also demonstrates its superior simulation speed. This section recalls only the details of network traffic generation.

Our approach is to project network traffic based on the latest advances in communication middleware for MPSoCs and to assess its performance with an on-chip network as the communication backbone. We derive from the queue-based library in [9] the guidelines for producer-consumer interaction. That library is suitable for a number of MPSoC architectures,

	16 tile			64 tile				
Topology	4-ary 2-mesh	2-ary 4-mesh	2-ary 2-mesh	8-ary 2-mesh	2-ary 6-mesh	2-ary 6-mesh High-Speed	2-ary 4-mesh	2-ary 4-mesh Reduced
Max. switch arity	6	6	10	6	8	8	12	12
Post-synthesis freq.	1 Ghz	1 Ghz	850 Mhz	1 Ghz	900 Ghz	900 Ghz	790 Mhz	790 Mhz
Post-layout.	786 MHz	640 Mhz	600 Mhz	786 Mhz	640 Mhz	786 Mhz	500 Mhz	500 Mhz
Core speed (max. 500)	393 MHz	320 Mhz	300 Mhz	393 Mhz	320 Mhz	393 Mhz	250 Mhz	500 Mhz
Cell Area	$949k\ \mu m^2$	$1108k\ \mu m^2$	$733k\ \mu m^2$	$4461k\ \mu m^2$	$7356k\ \mu m^2$	$22784k\ \mu m^2$	$2610k\ \mu m^2$	$2611k\ \mu m^2$
Power	0.67 W	0.64 W	0.32 W	—	—	—	—	—
Latency on top dimensions								
Dimension 3	—	—	—	1	1	2	1	1
Dimension 4	—	—	—	1	1	2	2	2
Dimension 5	—	—	—	1	2	2	—	—
Dimension 6	—	—	—	1	3	3	—	—

TABLE II
PHYSICAL PARAMETERS OF TOPOLOGIES UNDER TEST

including the tile-based MPSoC scenario addressed in this paper. Therefore, we built an abstraction layer on top of our TL simulator, which models the behavior of a processor tile and of its HW/SW communication support. In essence, the tile architecture consists of a processor core and a local memory core, as illustrated in Fig.1(a). Both cores are connected to the network through a network interface initiator and target respectively. We assume that the two network interfaces can be used in parallel. While the processor is reading/writing from/to other tiles, the processor core of other tiles can read/write from/to the tile local memory. We assume producer-consumer communication between tiles based on the handshake in Fig.1(b). The producer checks local semaphores indicating whether there are previous pending messages for the target destination. If not, it writes communication data to the local tile memory and notifies data availability to the consumer by unblocking a remote semaphore. The consumer was meanwhile performing local polling on that semaphore. The producer is then free to carry out other computation or communication activities to other consumer tiles. The consumer then reads computation data from the producer tile, and sends a notification upon completion. This allows the producer to send another message to this specific consumer. The implementation of this communication protocol involves 4 network transactions: notification of data availability, read request, actual data transfer and notification of transfer completion. The producer local polling is performed in order to avoid congesting the network in case the consumer is slow in absorbing its input messages. The consumer local polling allows the consumer to synchronize data transfer operations from multiple producers. This avoids the collision of multiple packets in the network from the producers to the same consumer, since this latter operates all transfers once at a time. Under these working conditions, concentrated architectures trading bandwidth for latency become attractive. However, physical implementation effects might put this picture in discussion.

IV. PHYSICAL SYNTHESIS

Link latency and maximum achievable frequency are key parameters to determine performance, area and power of each topology. However, they can only be quantified by post-layout analysis. This motivates our bottom-up approach to topology exploration. Due to synthesis time constraints, real physical parameter values were obtained only for 16 tile systems, while those for 64 tile systems were extrapolated based on the synthesis experience on the smaller systems and on ad-hoc scalability experiments.

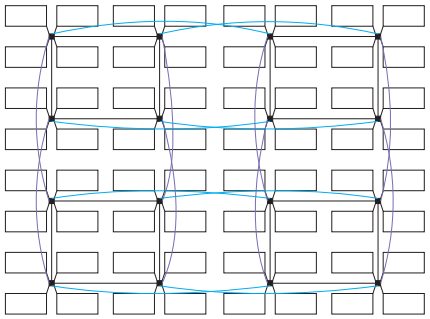

Fig. 2. Floorplan of a 2-ary 4-mesh with 4 tiles per switch.

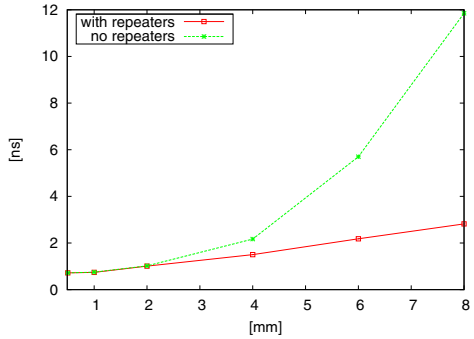

Fig. 3. Switch-to-switch distance and associated critical path.

A. 16 tile networks

Network building blocks have been synthesized in isolation for maximum performance. Post-synthesis achievable frequencies are reported in Table II - 3rd row. They only account for timing paths in network logic and ignore those going through switch-to-switch links. We always found the critical paths to be in the switches and never in the network interfaces, and this explains why the network speed closely reflects the maximum switch radix of each topology.

When post-layout speed is considered, we observe that inter-switch wiring has caused a significant performance drop for all topologies, depending on the wiring intricacy of each of them. As reported in Table II - 4th row, the more complex connectivity pattern of 2-ary 4-mesh results into a larger frequency drop than the 2D mesh. The 2-ary 2-mesh pays its lower number of switching resources with a larger switch-to-switch separation, and hence with a severe degradation of network performance due to link delay.

Since frequency-ratioed clock domain crossing is implemented in xpipesLite network interface, network speed affects

IP core speed. For this latter, a maximum value of 500 MHz is assumed in the context of multi-core embedded microprocessors. In spite of the post-layout speed drop, IP cores cannot sustain the network speed just at the same and therefore a divider of 2 is applied (Table II - 5th row).

The total larger number of switch I/O ports used by the 4-hypercube well motivates its larger area footprint than the 2D mesh. Cell (floorplan) area is considered in Table II - 6th row, while chip floorplan area is not reported since no specific optimizations were applied to it. Although the 2-ary 2-mesh has half the number of switches, its area is not halved as well, due to the fact that those fewer switches have a larger radix.

B. 64 tile networks

For 64 tile networks, we had to assume a different physical design technique than for smaller scale systems. In fact, should switch-to-switch link delay still impact overall network speed, this latter would become unacceptably low. For this reason, link pipelining becomes mandatory. While it might turn out to be expensive for 16 tile systems, for larger networks it can break long timing paths across on-chip interconnects and sustain network speed even in presence of long links.

The number of pipeline stages depends on the link length on the layout. As a first approximation, layouts for 64 node topologies can be obtained in a modular way by replicating those of 16 node topologies 4 times. So, for instance, the layout of a 2-ary 6-mesh can be derived from that of the 4-hypercube while from a 2-ary 2-mesh it is possible to draw the layout of a 2-ary 4-mesh. Since in the 64 tile systems the switch radix is larger, the main inaccuracy may only regard the size of some routing channels. The 2D mesh is an exception to this scaling policy, in that its extension to 64 tiles is straightforward and retains the switch radix.

From the layouts, we were able to project link lengths for each topology dimension and, consequently, link latencies when link pipelining is applied. Table II reports latency results for the top dimensions, since the lower ones always feature 1 cycle latency. This way, the same post-layout frequency of the originating 16 tile topologies could be retained. The only exception regards the 2-ary 4-mesh, where the maximum frequency is determined by the large switch radix and not by link delay anymore.

The mapping function between wirelength and link delay has been experimentally derived. We placed two switches in the layout at increasing distance, and measured the post-routing critical path going through the switch-to-switch link as a function of the distance. The resulting curve is illustrated in Fig.3. We have two physical design options. Repeater stages can be used in link segments to speed up signal propagation, resulting in an almost linear increase of critical path delay. This technique also allows to limit the number of link pipeline stages. In contrast, some previous works advocate the use of unrepeated global wires to avoid an exponential increase in area and power. As can be observed from Fig.3, the consequence would be a steep increase of link latency, and hence of retiming stages. We leave the exploration of the most power-efficient solution for previous work, and consider the combined use of repeater and retiming stages in this work.

In order to point out the key role of physical design techniques in determining system performance, we consider two topology variants featuring the same connectivity pattern of presented topologies but different physical parameters.

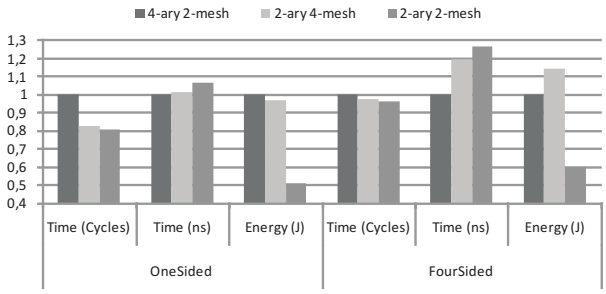

Fig. 4. Normalized execution time for 16 tile topologies.

The *high-speed* version of the 2-ary 6-mesh topology illustrated in Table II (7th column) differs from the reference 2-ary 6-mesh only for the higher post-layout operating frequency, which was made equal to that of the 2D mesh. This was allowed by an aggressive use of link pipelining, applied also to the links in the third and forth dimensions. Since each pipeline stage is not just a retiming stage but also a flow control stage, it needs to have a 2 slot buffer, and this leads to the significant area overhead illustrated in Table II with respect to the baseline 2-ary 6-mesh. The same design technique could not be applied to the 2-ary 4-mesh, since the high switch radix poses an upper bound to the performance optimization achievable by link pipelining.

On the other hand, maximum speed of the 2-ary 4-mesh is so low that a modification of the frequency ratio at the network interface might become convenient. In fact, if we change the ratio from 2 to 1, we enable IP cores to run at the same speed of the network. So, the network slowdown might be offset by speeding up the IP cores. We refer to this topology with NI divider set to 1:1 as the *reduced* 2-ary 4-mesh.

V. TOPOLOGY PERFORMANCE RESULTS

In order to simplify topology analysis, we assumed a workload distribution between the tiles which de-emphasizes the role of the topology mapping algorithm. In fact, we consider a parallel benchmark consisting of one or more producer tasks, a scalable number of worker tasks and 1 or more consumer tasks. Every task is assumed to be mapped on a different hardware tile. The producer task(s) reads in data units from the I/O interface of the chip and distributes it to the worker tasks. There are no constraints on which worker tile has to process a given data unit. Output data from each worker tile is then collected by one or more consumer tiles, which write them back to the I/O interface. All communications follow the queue based semantics illustrated in Subsection III-D. The following assumptions were made on the I/O interface. A maximum of 8 I/O ports is assumed for 64 tile systems, each one used for input or for output. This number was reduced to 2 I/O ports for 16 tile systems. Such ports are accessed through sidewall tiles. The mapping of producer(s) and consumer(s) tasks is therefore constrained to these tiles. This I/O architecture is compliant with that of commercial embedded microprocessors, such as [3]. We set latency for access to the off-chip I/O devices as a function of their frequency. We considered 20 cycles at 500 MHz and 15 cycles at 350 MHz.

While insensitive to worker tile mapping on the topology, our benchmark is still sensitive to I/O tile mapping on the chip periphery. For this reason, two scenarios are considered:

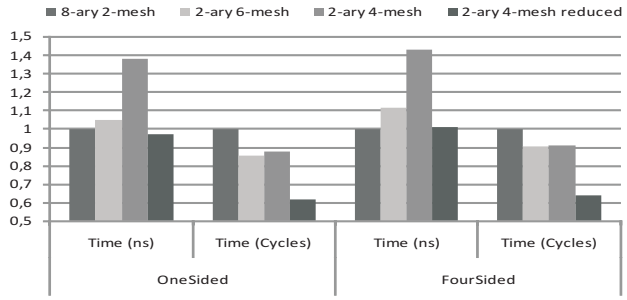

Fig. 5. Normalized execution time for 64 tile topologies.

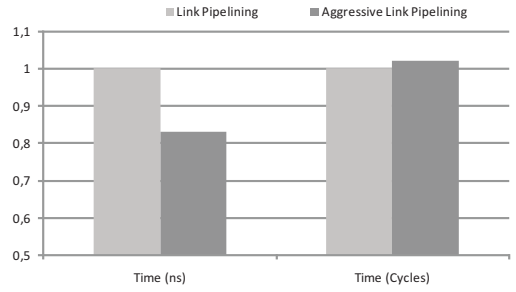

Fig. 6. Relative performance comparison between 2-ary 6-mesh and its high-speed variant leveraging aggressive link pipelining.

$OneSided$: all the I/O tiles are placed on the same side of the chip. This mapping has a high probability of I/O streams collision.
$FourSided$: I/O tiles are spread across the four sides of the chip. For 64 tile systems, at least one input and one output is placed at each side. For 16 tile systems, I/O tiles are placed at opposite sides to balance the average number of hops to input and output tiles for all workers.

A. 16 tile

Figure 4 shows performance of 16 tile topologies in clock cycles and elapsed time. The left side shows results for $OneSided$ mapping, while the right side shows results for $FourSided$ mapping. In $OneSided$, the hypercube (2-ary 4-mesh) reduces total number of cycles by 27.4%. In this mapping, inputs and outputs are placed at the top of the chip. Therefore, path length is very irregular, as tiles located at the top of the chip can reach input and output through a shorter path than the tiles located at the bottom. This makes topologies with higher network diameter more sensitive to this effect. Moreover, the probability of collision between I/O streams is quite high, thus penalizing topologies with fewer dimensions. The concentrated hypercube (2-ary 2-mesh) reduces cycles only by 1.6% over the hypercube, despite its lower diameter. The main reason for this lies in the chip I/O: as the bottleneck introduced by the topology is alleviated, the external I/O bottleneck arises. So, the maximum improvement that can be achieved in the 16 tile system is bounded by I/O speed. On the other hand, $FourSided$ mapping (see Figure 4) reduces I/O streams collision probability while providing homogeneous path length, thus decreasing performance differences among topologies.

Unfortunately, these results become irrelevant when considering the real operating frequency of each topology. While in $OneSided$ mapping the reduction of cycles of the 4-hypercube barely compensates for its lower frequency, in $Foursided$ mapping it is not enough, and the 2D mesh turns out to be the best topology overall.

Finally, Figure 4 shows the energy consumed by each topology. These numbers are measured on the post-layout netlists illustrated in Section IV. While the 4-hypercube consumes almost the same or even more energy than the 2D mesh depending on the I/O tile mapping, the concentrated hypercube shows superior energy saving properties (from 40 to 50% less than the 2D mesh).

B. 64 tile

Fig.5 shows performance of 64 tile topologies for both mappings in cycles and real elapsed time. The projected link latencies from Table II are considered for each topology. The trend observed in 16 tile systems with respect to the I/O tile mapping seems to be confirmed also in 64 tile systems. With this system scale, the impact of the different operating frequencies of the topologies is even more apparent. For instance, performance in cycles of both non-reduced hypercubes seems quite similar, but when frequency is taken into account, performance results are very different.

Again, the 2D mesh outperforms both the non-reduced hypercubes, due mainly to two reasons. The first one is of course the decreased operating frequency of the hypercubes, which is the price to pay for the more intricate physical routing (6-hypercube) or for the longer links used to interconnect few network resources sparse all around the chip (4-hypercube). Performance improvements in cycles are not such to offset the lower operating speed. However, one might expect a larger cycle reduction from an hypercube, which conflicts with the real 10% reduction measured in our simulations. This is due to the fact that the systems under test are I/O constrained, since computation tiles spend around 50% of their time waiting to send data to the consumer tile. This constraint introduces an upper bound to topology-related performance optimization. Removal of the I/O bottleneck has to be considered as mandatory to achieve performance differentiation between topologies.

An interesting effect can be observed when looking at the results of both 2-ary 4-mesh topologies. In the reduced one, clock domain ratio is set to 1, so NoC and tiles work at the same frequency, allowing to increase tile speed to 500 MHz. In this case, although the number of cycles is greatly reduced (29% less cycles than the 2D mesh), the low network frequency compensates for that, so performance is very similar. However, the reduced 2-ary 4-mesh requires 4 times less switches than the 2D mesh, half the number of ports and works at half the frequency, so power requirements are going to be lower, while preserving performance.

Finally, the effects of an aggressive utilization of re-timing stages over performance was analyzed for the 2-ary 6-mesh. Figure 6 shows the performance of 2-ary 6-mesh and of its high-speed variant in both cycles and elapsed time. As can be observed, the high-speed 2-ary 6-mesh slightly increases the number of cycles. However, the increased operating frequency is high enough to reduce elapsed time by 27% over the baseline topology. This might cause the high-speed 2-ary 6-mesh to become competitive with the 2D mesh performance-wise, but a significant area and power overhead needs to be taken into account, associated mainly with the additional

buffers used in retiming stages.

VI. Conclusions

This paper takes a bottom-up approach to the assessment of k-ary n-mesh topologies for regular tile-based architectures. Scalability of presented results to 64 tile architectures is considered. The paper considers a number or real-life issues: physical constraints of nanoscale technologies (post-layout performance, area and power results are given), different physical design techniques, the role of the chip I/O, the communication semantics of middleware for MPSoCs and the role of I/O tile mapping.

We found that the intricate wiring of multi-dimension topologies or the long wires required by concentrated k-ary n-meshes can be changed into 2 different kinds of performance overhead by means of proper design techniques:

- operating frequency reduction. This is likely to be the technique of choice for small scale systems. In this case, in spite of a lower number of execution cycles, multi-dimension topologies loose in terms of real execution time due to lower working frequency. Nonetheless, reducing the number of dimensions and connecting more cores to the same switch represent a way to trade performance for power and area;
- increase of link latency. An aggressive utilization of retiming stages allows to sustain operating frequency while increasing network latency. The switch delay associated with its radix poses an upper bound to the effectiveness of this technique. Finally, a significant area and power overhead is to be expected, since retiming stages need to be also flow control stages.

Overall, we found the 2D mesh to still outperform the hypercubes even in 64 tile systems. This is counterintuitive, since hypercubes should scale better in principle. The motivation lies in the mismatch between multi-dimension topologies and the 2D silicon surface (whatever the kind of performance overhead it is changed into) and in the chip I/O bottleneck, which prevents an aggressive performance speed-up at least in clock cycles. Removal of this bottleneck is mandatory to achieve significant performance differentiation between topologies.

These considerations are architecture-specific to some extent. The xpipesLite NoC for instance uses frequency-ratioed clock domain crossing at the network interface. So, we found the opportunity to take profit of the low speed of concentrated hypercubes to change the frequency divider at the network interface and to have the cores running faster. We therefore got a topology that achieves the same performance of the 2D mesh while consuming much less hardware resources and power. The availability of more complex synchronization techniques such as asynchronous FIFOs at network interfaces may extend this benefit to other topologies as well.

References

[1] STn8811A12 Mobile Multimedia Application Processor, available online: http://www.st.com

[2] SPEAr Plus600 dual processor cores, available online: http://www.st.com

[3] TILE64 PROCESSOR FAMILY, available online: http://www.tilera.com/pdf/ProBrief_Tile64_Web.pdf

[4] Tensilica LX configurable processor, Tensilica Inc., http://www.tensilica.com

[5] S.Stergiou et al., "Xpipes Lite: a Synthesis Oriented Design Library for Networks on Chips", DAC, pp.559-564, 2005.

[6] S. W. Keckler et al., "A wire-delay scalable microprocessor architecture for high performance systems", ISSCC'03, Feb. 2003, pp. 168-169.

[7] Z. Yu et al., "An asynchronous array of simple processors for DSP applications", in ISSCC'06, Feb. 2006, pp. 428-429.

[8] Benini, L., De Micheli G., "Networks on Chips: a New SoC Paradigm", IEEE Computer 35(1), pp.70-78, 2002.

[9] Dalla Torre, A., Ruggiero, M., Acquaviva, A., Benini, L., "MP-Queue: an Efficient Communication Library for Embedded Streaming Multimedia Platform", IEEE Workshop on Embedded Systems for Real-Time Multimedia, 2007.

[10] Balfour, J., Dally, W.J., "Design Tradeoffs for Tiled CMP On-Chip Networks", ACM International Conference on Supercomputing, 2006.

[11] S. Vangal et al., "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS", ISSCC 2007, pp.98-589, 2007.

[12] Coppola, M.; Locatelli, R.; Maruccia, G.; Pieralisi, L.; Scandurra, A., "Spidergon: a novel on-chip communication network", International Symposium on System-on-Chip, 2004, pp.16-18, 2004.

[13] Bourduas, S.; Zilic, Z., "Latency Reduction of Global Traffic in Wormhole-Routed Meshes Using Hierarchical Rings for Global Routing", IEEE International Conf. on Application-Specific Systems, Architectures and Processors, pp.302-307, 2007.

[14] Martinez Vallina, Fernando; Jachimiec, Nathan; Saniie, Jafar; " NOVA interconnect for dynamically reconfigurable NoC systems", IEEE International Conf. on Electro/Information Technology, 2007, pp.546-550, 2007.

[15] Gilabert, F., Gomez, M.E., Lopez, P.J., "Performance Analysis of Multidimensional Topologies for NoC", ACACES 2007, poster session with proceedings at the Summer School.

[16] Medardoni, S., Gilabert, F., Bertozzi, D., Gomez, M.E., Lopez, P.J., "Towards an Implementation-Aware Transaction-Level Modeling of On-Chip Networks for Fast and Accurate Topology Exploration", INA Workshop, presented at the HiPEAC08 conference, 2008.

[17] Mirza-Aghatabar, M.; Koohi, S.; Hessabi, S.; Pedram, M., "An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models", Euromicro Conference on Digital System Design Architectures, Methods and Tools, Page(s):19-26, 2007.

[18] Luciano Bononi, Nicola Concer, Miltos Grammatikakis, Marcello Coppola, Riccardo Locatelli, "NoC Topologies Exploration based on Mapping and Simulation Models", Euromicro Conference on Digital System Design Architectures, Page(s):543-546, 2007.

[19] Wang, H., Peh, L.S., Malik, S., "A Technology-Aware and Energy Oriented Topology Exploration for On-Chip Networks", Design Automation and Test in Europe, Vol.II, pp.1238 - 1243, 2005.

[20] Pavlidis, V.F.; Friedman, E.G., "3-D Topologies for Networks-on-Chip", International SOC Conference, Page(s):285 - 288, 2006.

[21] I.Hatirnaz, S.Badel, N.Pazos, Y.Leblebici, S.Murali, D.Atienza, G.De Micheli, "Early Wire Characterization for Predictable Network-on-Chip Global Interconnects", SLIP'07, pp.57-64, 2007.

[22] Murali, S., De Micheli, G., "SUNMAP: a Tool for Automatic Topology Selection and Generation for NoCs", Proc. of the Design Automation Conference, 2004, pp.914-914.

[23] Soteriou, V., Eisley, N., Wang, H., Li, B., Peh, L.S., "Polaris: a System-Level Roadmapping Toolchain for On-Chip Interconnection Networks", IEEE Trans. on VLSI 15(8), 2007, pp.855-868.

[24] F.Gilabert, S.Medardoni, D.Bertozzi, L.Benini, M.E.Gomez, P.Lopez, J.Duato, "Exploring High-Dimensional Topologies for NoC Design Through an Integrated Analysis and Synthesis Framework", Proc. of the 2nd Int. Symposium on Networks-on-Chip, to appear in 2008.

[25] A.Pullini et al., "Fault tolerance overhead in network-on-chip flow control schemes", SBCCI, pp.224-229, 2005.