Hierarchical Network-on-Chip for Embedded Many-Core Architectures

Alexandre GUERRE, Nicolas VENTROUX, Raphaël DAVID
CEA, LIST,
Embedded Computing Laboratory
Gif-sur-Yvette, F-91191, France;
Email: firstname.name@cea.fr

Alain MERIGOT
Institut d'Electronique Fondamentale
Université Paris Sud
Orsay, F-91405, France;
Email: alain.merigot@u-psud.fr

***Abstract*—The need for computing power drastically increases and one good solution is to use many-core architectures. Besides, complex embedded applications become data-dependent and their execution time depends on their input data. For this reason, on-line task and data allocation is needed to optimize the architecture efficiency. Moreover, communications are a complex problem in many-core architectures. Because of dynamic allocation, communication paths and network loads become unpredictable, which must be handled by the network. This paper proposes an evaluation of different network topologies in terms of performance and area for many-core architectures. It concludes that hierarchical networks are the best trade-off. In particular, the MultiCross topology is 10 times more efficient than the mesh topology.**

I. Introduction

In response to an ever increasing demand for computing power, the performance of embedded system architectures have improved constantly over the years. Today, multi-core processors have become the mainstream in embedded systems [1]. In this context, according to ITRS [2], a 32% yearly increase in the number of cores will be necessary to keep up with the applications' needs. ITRS assesses that in 2012, the number of cores in a chip, will exceed 100 cores. In order to guarantee a sufficient energy and area efficiency, each part of the architecture must be optimized. In this paper, we will focus on communication structures adapted to shared-memory many-core architectures, also named Network-on-chips (NoCs) [3]. Processors are interconnected with local and shared memory banks.

In addition, these many-core architectures must support the execution of dynamic computation-intensive applications. Algorithms have become highly data-dependent and their execution times depend on their input data. Therefore, to optimize the overall execution time, a dynamic data and task allocation is necessary. This dynamic optimization and the increase in the number of processors in a chip generate important communication constraints. Indeed, an off-line task partitioning imposes controlled and known communications, whereas an on-line dynamic task allocation generates unpredictable communications, i.e. network loads, message destinations and sources, or message sizes. This can have an impact on contentions and the efficient bandwidth of the network. For these reasons, this dynamic context must be taken into account in the interconnection design.

In the industrial world, multi-domain many-core architectures already exist and use different kinds of NoCs. HL-256 from Plurality [4] uses a butterfly multistage network, TILE64 from Tilera [5] prefers multiple meshes, whereas Larabee from Intel [6] implements multiple rings to interconnect their processors. Actually, a NoC is characterized by many parameters and its design space is very large. However, all these parameters have to be accurately chosen to optimize the architecture design. Virtual channels [7] or buffer sizes [8] are some examples of dimensioning parameters. In this paper, we will focus on the network topology, which has a very important impact on performances, especially with a dynamic and unpredictable context. To correctly size the architecture with a dynamic task allocation, it is essential to use a regular network to homogenize performances. In addition, the topology has to support and limit contentions whatever are communications.

Some other network topology comparisons have been published. In [9], Salminen et al. list many existing networks and detail their whole parameters (the topology, the size), as well as evaluation criteria. However, no performance or area comparisons are done and their study cannot be entirely used to design new many-core architectures. In [10], Tutsch and Malek make an accurate comparison but only between a mesh and a butterfly multistage topology. An other approach, proposed by Murali and De Micheli [11], uses a tool to define a topology according to application needs. From an application graph, processors are organized around different topologies and tasks are statically allocated on them. The average latency, the area and the power consumption are used as exploration parameters. Unfortunately, this tool cannot define a multi-domain architecture, as well as NoCs in a dynamic execution context. Finally, no existing study offers a large comparison between many NoC topologies for multi-domain, dynamic applications and shared-memory many-core architectures.

In this paper, 8 different NoC topologies are compared. This includes 5 well-known basic interconnection topologies named multibus, ring, multistage, mesh and torus. Hierarchical networks are also studied and 3 of them are proposed: CrossTorus, CrossRing and Multicross. Performance comparisons will evaluate their abilities to support network loads, whereas area comparisons with ASIC synthesis will measure

their area efficiency. These comparisons will explore NoC topologies for a many-core architecture in a multi-domain context.

The remainder of this article is organized as follows: section II presents the simulation environment for the performance comparison and the ASIC synthesis tool for the area comparison. Section III describes all studied networks. Then, section IV presents an area and performance comparison. An area efficiency analysis of these 8 basic and hierarchical networks completes this section. Section V explores the different configurations of hierarchical networks in order to find the best ones. Finally, section VI concludes on the best network topology for many-core shared-memory architectures and dynamic applications.

II. Simulation and Synthesis Environment

The comparison between our network topologies is done with two main criteria in order to study their area efficiency: the performance and the area. This section presents the environment that is used for these evaluations.

A. Simulation Environment

To evaluate NoC performances, a simulation environment is used to implement these 8 different topologies. This environment performs approximate-timed TLM SystemC communications to provide accurate and fast simulations [12]. This framework implements each network, traffic generator and memory. The configuration of each network is described in the next section. We consider only shared-memory architectures. Thus, each communication is a request to a memory. Access times and access conflict timings are computed into memories. Traffic generators send only blocking read request with a fixed data size.

1) Traffic Modeling: We propose to implement two kinds of traffic into our traffic generators, in order to represent two different applications. The first one is a uniform traffic that matches with an unpredictable load in the network. A "locality rate" parameter introduces a locality notion for hierarchical networks. This corresponds to the intra-cluster communication rate. The second one is a normal distribution traffic that mainly represents local communications. It stands for a "smart" application mapping on processors through the network. The normal distribution variance gives the main distance that can be reached by messages, whereas the mean represents the closest memory from the traffic generator. Both traffics are representative of a dynamic multi-domain application. The message sending frequency can be dynamically modified. The frequency controls the network load during the simulation. When the message creation frequency is higher than the sending rate, messages are buffered in an infinite FIFO memory.

2) Network Modeling: The main part of this simulator is the network modeling, because it gives the principal latency information. All basic networks use the same framework to estimate the time spent in it. This framework is described in a single SystemC module. The network behavior can be described as follows. First of all, a request is sent to the network and is stored in a list of pending requests. If it is the first one in the list, it sets off an event and wakes up the main thread of the network. This thread processes pending requests, calculates the path taken by the request in the network and computes a penalty in case of contention. Then, the request is sent to the destination. When the response comes back, a wait function is launched with the computed communication time as argument. Finally, the response is sent back to the initiator. In the response, the initiator has some information like the number of crossed routers or the time spent in the different modules of the MPSoC platform.

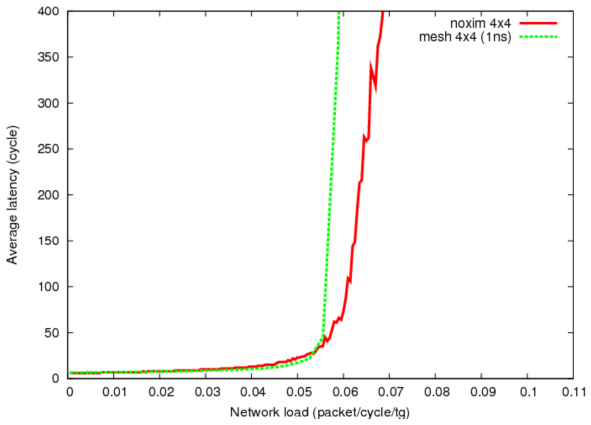


Fig. 1. Comparison between Noxim and our approach. (tg: traffic generator)

The difference between each NoC remains on the path decision and the contention calculation, which provides information on packet collisions. When a request is sent, an entry point ID is given to the request to know the position of the initiator. When the request enters in the network, this one begins with the path calculation. Depending on the routing type and this ID, the network builds a list of virtual routers and links. The time used by a request to enter and exit the router is associated with each element in the list. The network also calculates all contentions with other requests already in the network. The contention calculation is based on the comparison between the path and the timing of the new request, and requests already in the network. A proportional penalty is calculated with the latency of the network, according to the request size and the topology characteristics. For example, if two requests are routed on two different buses in a ring network, they will cross the same router without an added contention.

To validate the accuracy of our approach, some comparisons have been done with the cycle-accurate Noxim simulator [13]. For the experiment, random traffics on a 4x4 mesh, with XY routing using a wormhole technique, are used. Figure 1 shows that our approach obtains an error inferior to 9% until the network overload. After this saturation, our approach has more pessimistic results than the Noxim simulator but this has no impact on this study.

This framework has the particularity to support combinations of basic networks. As a result it allows the simulation

of a wide range of hierarchical network. In order to build a hierarchical network, a bridge must be placed between the different basic network parts to perform a correct interface. This bridge updates the requests and manages the number of incoming requests at the same time. Updating a message corresponds to changing the entry point ID of the request when it crosses the interface between the NoCs. When a request enters in another network, its entry point ID becomes the bridge ID. A semaphore is needed in the bridge because a TLM channel cannot block multiple requests. This case occurs when two traffic generators send requests that need to transit to another network through the same bridge. If these two requests reach the bridge at the same time, they will access concurrently to the same channel. So the bridge has to manage these multiple requests to have a correct estimation of the time spent waiting to cross this interface. This management is done by a semaphore. An approximate-timed TLM communication can transit through the channel only if it can take a semaphore token. This technique allows to block the initiator thread of the second request and to continue the simulation. In addition it allows a correct timing simulation when exploring hierarchical networks. Indeed, timing is incremented by the SystemC kernel during the waiting phase and no estimation is done. With this bridge all networks can be combined in a hierarchical way to evaluate their benefits and their drawbacks.

B. Synthesis Environment

All basic networks are described in RTL and synthesized using design compiler [14] from SYNOPSYS. For the mesh and the torus network, the VHDL codes come from the NoCem project [15]. The ring and multistage network are based on modified NoCem router. All synthesis are made with the TSMC 40nm library [16]. For a hierarchical network, its estimated area is obtained by adding areas of the basic networks composing it. As this estimation technique is based on real synthesis results, it allows realistic comparisons.

Combining these two environments offers to find the best trade off between performance and area by calculating the accurate area efficiency of each network. The next section describes in detail the configuration and parameters of compared networks.

III. Network Descriptions

For the evaluation, 8 different NoC topologies are studied. This includes 5 well-known basic interconnection topologies named multibus, ring, multistage, mesh and torus. These networks have been chosen because they are representative of the network design space [9]. Hierarchical networks are also evaluated and 3 of them are proposed: CrossTorus, CrossRing and Multicross. We chose to compare hierarchic networks because they present a regular topology and potentially limit the number of contention like in hierarchical bus. First of all, we are going to present the basic implemented networks, then we will describe hierarchical ones.

The multibus is implemented as shown in figure 2-a. In this network, an initiator drives a unique bus. The network

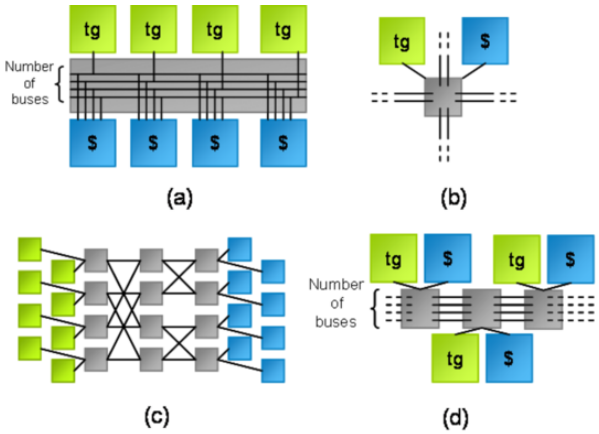


Fig. 2. (a) Multibus description. (b) Single part of a mesh or torus network. (c) Multistage description. (d) Part of a ring network.

owns several buses which can be shared or not. So changing the number of buses modifies the bandwidth. It is possible to consider only one initiator per bus. In that case, the multibus is considered as a "fully connected interconnect". For the rest of the paper, a multibus will refer to a fully connected network, i.e. initiators are not connected between them.

The mesh and the torus have the same configuration. One initiator and one target are linked together with a router as shown in figure 2-b. The number of columns and rows of the router matrix can be changed. An XY routing is implemented. These networks use a wormhole technique to transfer packets without virtual channels.

The multistage is an indirect fully connected network (Fig.2-c). It is divided into different stages which are composed of 4-input-output routers. These routers are linked with a butterfly topology. It also uses a wormhole technique to transfer packets without virtual channels. The multistage is used in a specific way, where all initiators are on the same side and all targets are on the other side. This arrangement simplifies the use of this network and it corresponds exactly to shared-memory architectures.

Figure 2-d presents a part of a ring network. In it, one target and one initiator are bounded to a router. A message has to cross every router when it transits through a ring. Each initiator can connect to only one ring. The number of rings will influence the bandwidth. Each ring is bi-directional. As for the mesh and torus, it uses a wormhole technique to transfer packets without virtual channels.

In the case of clustered architectures, each cluster receives a multibus as an inside-cluster network. To allow extra cluster communications, initiator ports and slave ports can be used and reserved on the inside-cluster multibus. We consider a link between the two network levels like the combining of an input and an output of the cluster. For each link, a inter-cluster network is implemented and bridges are placed between networks. And so, if we consider two links between two hierarchical levels, this corresponds to two inter-cluster

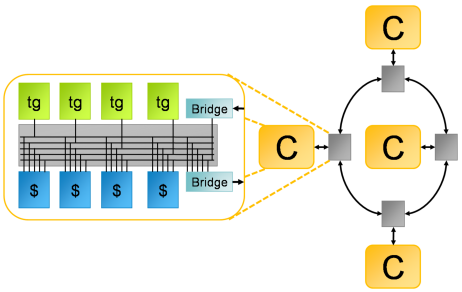


Fig. 3. Cluster view in a 4 clusters 1 link CrossRing.

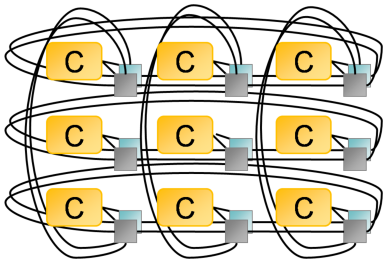


Fig. 4. CrossTorus with 9 clusters and 2 links.

networks. This duplication allows to increase the bandwith and limits contentions between cluster requests. The difference between hierarchical networks presented in the following paragraphs is about the inter-cluster network.

The CrossRing is a hierarchical network with a ring as inter-cluster network. The inter-cluster ring parameters can be defined as they could be in the non hierarchical case. Figure 3 presents a cluster view of a CrossRing network with one link between the two network levels.

The CrossTorus links the clusters with a torus inter-cluster network. Figure 4 shows a 9-cluster CrossTorus network with two links between the two network levels. As for the CrossRing, the inter-cluster network parameters can be selected. The CrossRing and the CrossTorus are two-level hierarchical networks.

The MultiCross implements a particular configuration where 4 clusters are connected around a ring router like on figure 5. This network offers fast links between a group of 4 clusters without going through the inter-cluster network. It limits the number of nodes in the inter-cluster network and therefore reduces the area. The MultiCross is a 3-level network. The first one corresponds to a multibus in each cluster. The second one connects 4 clusters in a group. The last level interconnects cluster groups. Nonetheless, the MultiCross is implemented by only two networks. Indeed, the second level network lies in each ring router. The next section details results from the area and performance evaluation.

IV. Network Evaluation

The network evaluation consists of a performance and an area comparison. In addition, an area efficiency analysis completes our comparative study. The evaluation platform is

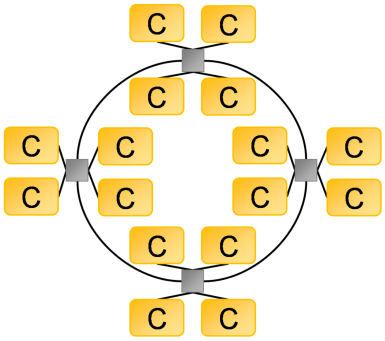


Fig. 5. MultiCross with 16 clusters and 1 link.

composed of 256 processors, represented by traffic generators, and 256 memories. Even if their number is fixed for each architecture, the processor clusterization and the number of links between hierarchical levels can be modified. To simplify the exploration space of hierarchical NoCs, only one configuration for each topology is explored. We consider 16 clusters of 16 processors for the MultiRing and the MultiTorus topologies, whereas we focus on 32 clusters of 8 processors for the MultiCross. Each cluster owns 2 links with the upper hierarchical level. These configurations are explored in the next section.

A. Performance Comparison

In this paragraph, we will determine the best performance network under different traffics. We consider the global latency of a message as the time between the message creation and the answer reception in the traffic generator. All considered latencies on next figures are the mean of 20 simulations with a minimun of 5000 requests by traffic generators. 1000 warm-up requests are generated at the beginning of each simulation. The simulation time depends on the packet injection rate. All routing algorithms are Deadlock free. All unfinished requests are not recorded and not taken into account in the calculation. The mean is computed from all traffic generator results. In our study, we only consider the MPSoC working area, which is before the network saturation. A network has a better performance if it can handle more network loads before overloading. For this performance comparison, we use 3 different traffics: a uniform traffic, a uniform traffic with a percentage of requests that stay in each cluster, and a normal distribution traffic.

As already explained, the use of a uniform traffic allows to simulate an unpredictable network load. Figure 6(a) shows the NoC average latency as a function of the network load. Because the multibus offers a direct and uniform link to each memory, it reaches the best performance. Actually, the more the distance with all memories is heterogeneous and important, the less the topology is high-performance with a uniform distribution. Thus, for instance, the torus is better than the mesh. The multistage owns a uniform 7-hop memory access, but its links between routers are too long to make this topology interesting. Besides, as depicted in this figure, the ring topology cannot reach good results with unpredictable network loads. Hierarchical networks are not better than a

torus. Their performance is explained by the fact that inter-cluster communications are penalized by the number of limited inter-cluster external links. Within hierarchical networks, the MultiTorus obtains the best performance, since the torus is well adapted to uniform communications.

If we now consider a uniform traffic with a percentage of requests that stay in each cluster, the performance of hierarchical networks increases as shown in figures 6(b) and 6(c). This parameter changes the repartition of the network load between local network and intra-cluster network. Non-hierarchical networks are not affected by this parameter because they are not clustered. Hierarchical networks become better than the mesh with 50% of intra-cluster communications and better than the torus with 75%. With a rate of 75% of intra-cluster communications, the MultiTorus becomes even better than the multibus. This traffic only shows how the locality affects the hierarchical networks and improves their performance. If this rate increases enough, they all become better than the multibus, which is however high performance oriented. This behavior can be explained by the fact that the multibus inside the cluster has better performance than the flat multibus. Indeed, the performance of the multibus is constrained by its size. Moreover, if communications stay inside a cluster, there is a smaller chance that contentions occur between communications.

To correctly estimate the impact of the locality, the comparison has to be done with a different traffic. We choose a traffic under a normal law to represent this phenomenon. We choose the deviance of normal law at 20. This represents the number of memories around the transmitter that receive the majority of the requests. It makes around 80% of memory access locality for a 256 traffic generator and 256 memory architecture. Figure 7 shows the average latency as a function of the network load with a normal distribution and a locality of 80%. As expected, the multibus is still not affected by the traffic type and keeps its high performance. Indeed, all memories are at the same distance for all traffic generators. On the contrary, a topology, with a non-uniform memory access, like the mesh is positively impacted, but this increase is limited by the topology borders. With this traffic, the MultiCross and the MultiTorus have better performance than the torus. It confirms that hierarchical networks are better with local traffics. Hierarchical networks remain less performant than multibus because the deviance is bigger than the cluster size and many communications go out of the clusters.

The local traffic could be representative of a real application behavior executed with a “smart” task allocation. In this case, hierarchical networks offer good performances and remain good candidates for a many-core architecture. Nonetheless, other criteria have to be verified. The next subsection continues the comparison in term of silicon area.

B. Area Comparison

This subsection focuses on the area, because it is an important parameter for the design cost of embedded systems. As already mentioned, the platform is composed of 256 traffic generators and 256 memories. All area results are obtained, as explained in the second section, by the synthesis of flat networks and by adding the area of different parts of hierarchical networks.

The Table I shows the area of the different basic networks depending on the number of inputs/outputs of the entire

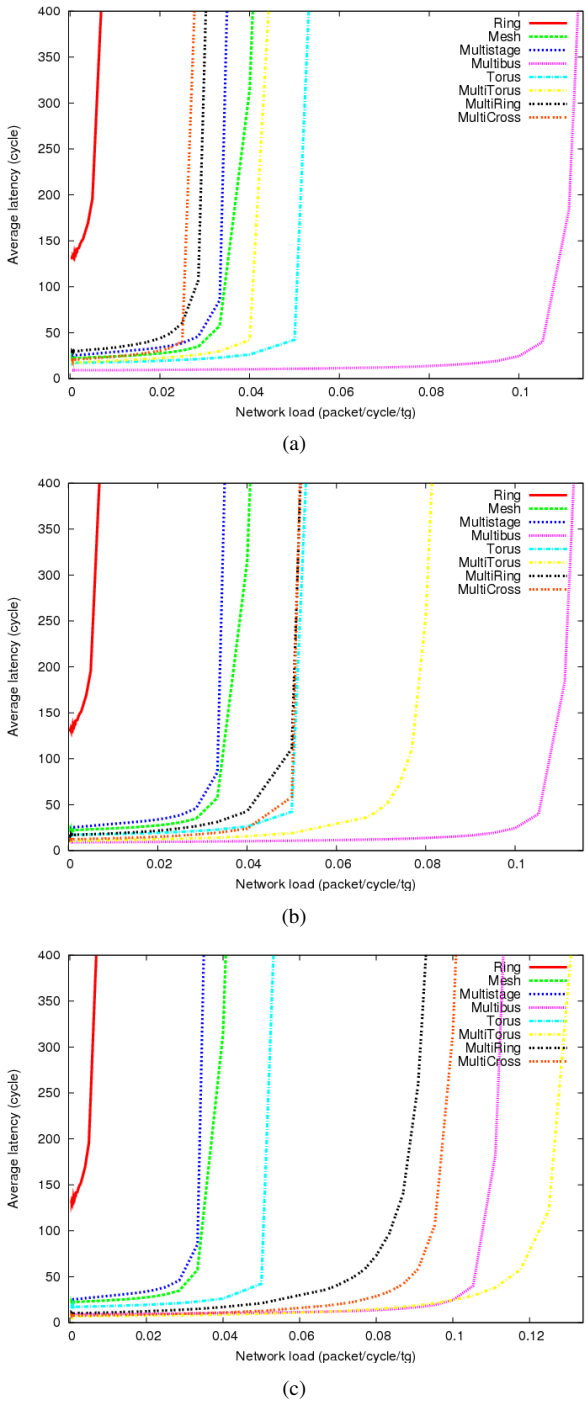


Fig. 6. (a) Average latency of all networks as a function of network load with a uniform distribution. (b) Average latency of all networks as a function of network load with a uniform distribution and 50% of inside-cluster requests for clustered networks. (c) Average latency of all networks as a function of network load with a uniform distribution with 75% of inside-cluster requests for clustered networks.

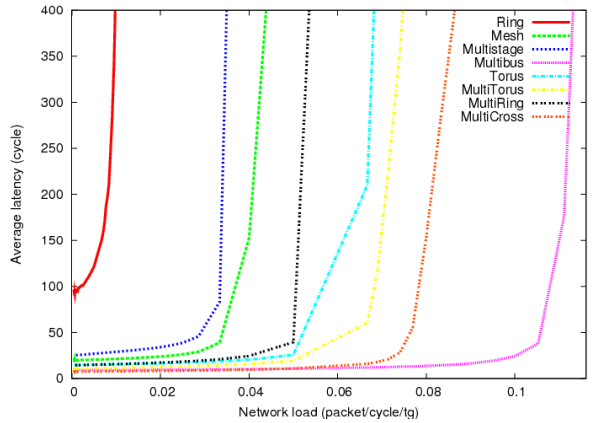


Fig. 7. Average latency of all networks as a function of network load with a normal distribution and 80% of locality.

TABLE I
AREA ESTIMATION TABLE IN mm^2 BASED ON TSMC 40NM LIBRARY. (X MEANS THAT NO RESULTS CAN BE PRODUCED, BECAUSE OF THE MULTISTAGE CONFIGURATION)

↓ Networks/Sizes →	8	16	32	64	256
Multibus	0,0193	0,0706	0,247	0,905	17,19
Mesh	0,11	0,235	0,46	0,913	4,24
Torus	0,139	0,258	0,524	1,017	4,337
Ring	0,106	0,196	0,38	0,698	3,030
Multistage	X	0,3	X	1,698	9,595

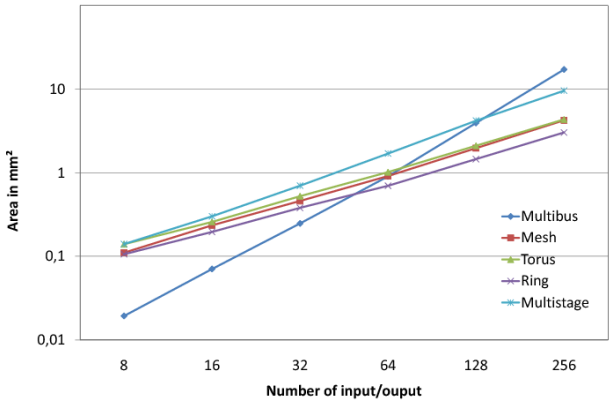


Fig. 8. Evolution of network area in mm^2 depending on the number of input.

network. These results are also used to build area estimation for hierarchical networks. The symbol "X" means that the multistage network configuration was not achievable. Indeed, with a 4-input-output router, the number of inputs has to be a power of 4. Figure 8 presents the area result table in a graphical view for a better comprehension. It is important to mention that the mesh, the torus and the ring have a linear rising that depends on the network size N. On the contrary, the multibus has an evolution in N^2 and the multistage in $N * log(N)$.

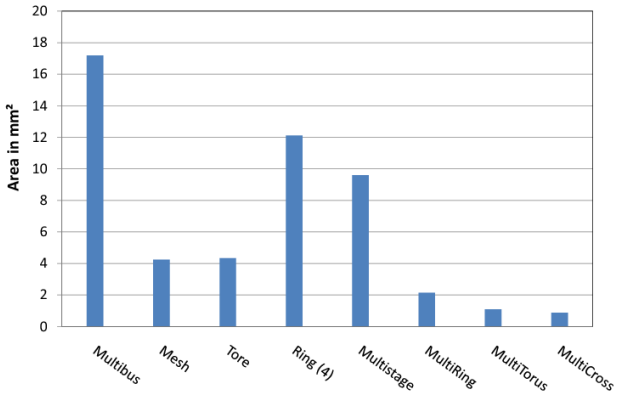


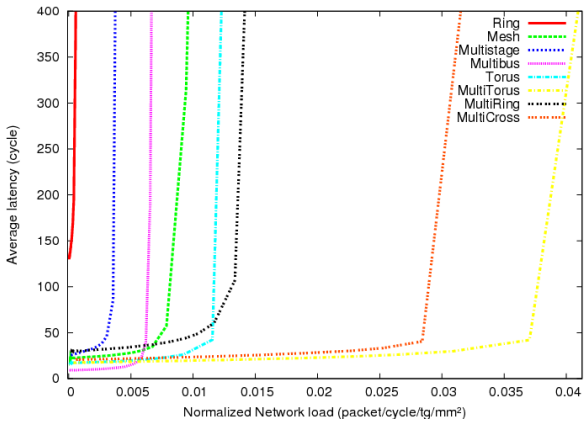
Fig. 9. Network area in mm^2 for 256 inputs/outputs in TSMC 40nm.

Figure 9 references the area of each basic and hierarchical network for a fixed size of 256 inputs/outputs. The multibus is the biggest network, followed by the 4-parallel ring and the multistage. The size of a network depends on the number of routers and the size of each router. The multibus can be seen as a single router with 256 inputs and 256 outputs, which explains its big size. The ring and the multistage also have a big silicon area because of too many routers. Hierarchical networks in these configurations are smaller than basic networks. Indeed, they use a multibus inside the cluster, which is the smallest network for reduced number of inputs/outputs. In addition, they limit the number of routers on the inter-cluster network compared to a non-hierarchical network. For all these configurations, the MultiCross is the smallest network, this is because it uses the crossbar inside routers as a second hierarchical level, and takes advantage of the router natural function. This decreases the area by slightly augmenting the size of each router instead of multiplying them. The biggest difference is between the multibus and the MultiCross topologies. Indeed, the multibus is 17 times bigger than the MultiCross. To be able to combine these two comparisons, the next subsection presents the area efficiency as a criterion to conclude this evaluation.

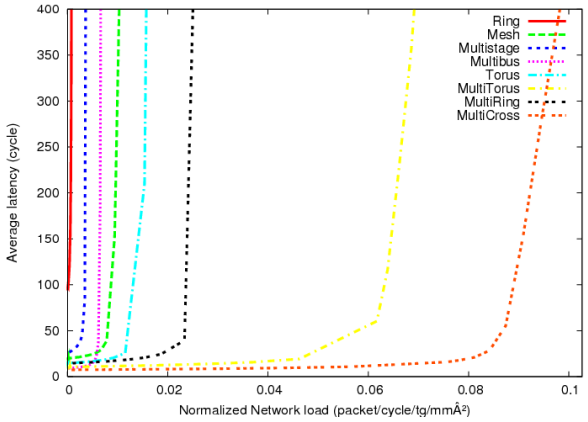
C. Area efficiency comparison

The last two subsections show an area and performance comparison. But for embedded systems, it is important to consider these two criteria together and the area efficiency. This subsection presents the area efficiency of all presented networks depending on the traffic. The area efficiency is not an easy criterion to represent for network. The performance is compared on the curve latency/network load. So, to introduce the area mesurement, we propose to normalize the network load by the size of the network. The new curves allow to compare normalized performance and so the area efficiency.

Figure 10(a) shows the area normalized network load with a uniform traffic. In general, hierarchical networks are more area efficient than non-hierarchical networks, since they can reach a



(a)



(b)

Fig. 10. (a) Average latency of all networks as a function of area normalized network load with a uniform distribution. (b) Average latency of all networks as a function of area normalized network load with a normal distribution and 80% of locality.

high performance with a smaller area. The multibus topology is the most performant, but with its important generated area, this topology becomes one of the less area efficient. This explains why the mesh topology is more used in embedded architectures than the multibus topology. The MultiTorus is 5 times more efficient than the mesh. Moreover, with an unpredictable load, it is also the best hierarchical network, with 33% more area efficient than the MultiRing and 25% than the MultiCross. The difference of area efficiency between hierarchical networks with a uniform traffic is explained by their inter-cluster networks. For example, the torus has a better area efficiency than the ring. Thus, the MultiTorus is more area efficient than the MultiRing.

As depicted in figure 10(b), the MultiCross remains the most area efficient with a normal distribution and 80% of locality. It is 50% more efficient than the MultiTorus, 10 times more efficient than the mesh and 7 times more than a torus topology. Its high area efficiency is due to its structure, which is designed for intra-cluster and neighboring-cluster communications.

This evaluation shows that hierarchical networks are more area efficient than non-hierarchical networks for the three kinds of traffics. The MultiCross remains the best tradeoff if a "smart" dynamic allocation is considered on a many-core architecture.

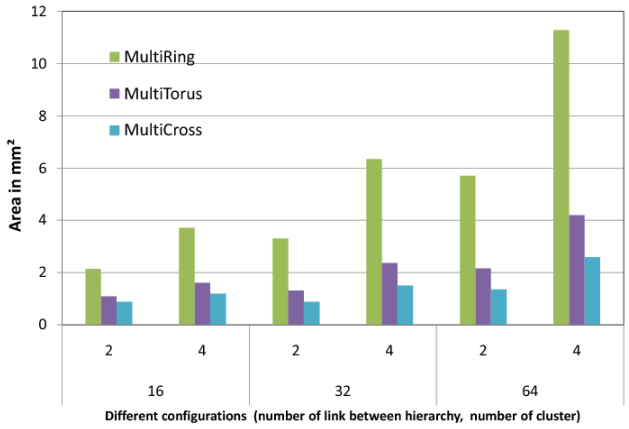


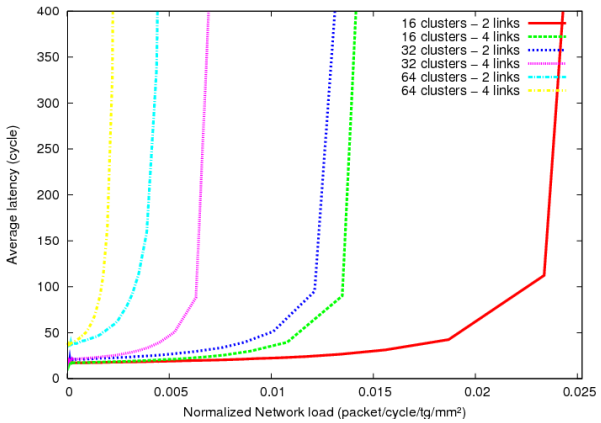
Fig. 11. Network area in mm^2 for 256 inputs/outputs depending on different hierarchical network configurations in TSMC 40nm depending on the number of links (2 and 4), and the number of clusters (16,32,64).

V. Hierarchical network exploration

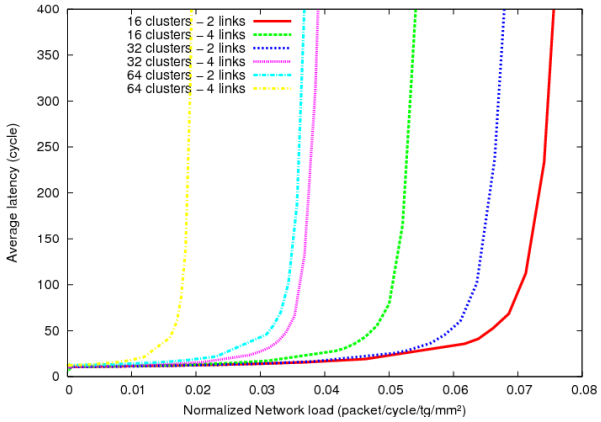
In the previous section, we made an arbitrary choice on the configuration of hierarchical networks. This section explores different configurations in order to determine the best one. The problem is to correctly size the number of processors into the cluster and the number of links between them. Links limit the number of simultaneous messages. In addition, the more the number of inside cluster processors increases, the more the number of simultaneous inter-cluster communications are. We choose to duplicate the network for each link, although this choice has a direct impact on the final area. We consider 2 or 4 links, and 16, 32 or 64 clusters. This means that the number of processors into a cluster is respectively equal to 16, 8 or 4. Area results are depicted in figure 11.

We still consider the area efficiency as the main comparison criterion. Figures 12(a), 12(b), 12(c) present the performance normalized by the area with a uniform distribution and 50% of inside-cluster communications. The MultiRing has the best area efficiency for the 16-cluster configuration. This is due to the high performance of the ring topology for small network sizes. However, duplicating the inter-cluster network does not offer enough performance compared to the area increase. The same conclusion can be derived with the MultiTorus. On the contrary, the MultiCross divides by 4 the number of routers. Therefore, it is possible to have more clusters without having a big area penalty. Like the MultiRing and the MultiTorus, the performance benefit from duplicating the inter-cluster network is not enough in comparison with the area increase.

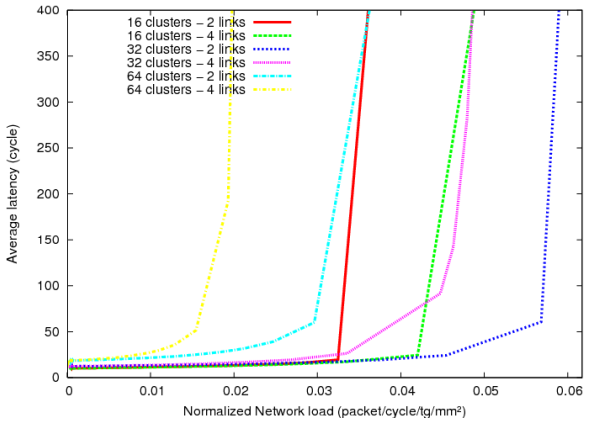
The chosen configurations for the previous comparisons are the most efficient and this exploration shows that it is necessary to correctly choose the number of clusters and links to obtain a good efficiency. If the architecture size changes, these configurations will not necessary be the best.



(a)



(b)



(c)

Fig. 12. (a) Average latency of a MultiRing as a function of area normalized network load with a uniform distribution and 50% of intra-cluster communications. (b) Average latency of a MultiTorus as a function of area normalized network load with a uniform distribution and 50% of intra-cluster communications.(c) Average latency of MultiCross as a function of area normalized network load with a uniform distribution and 50% of intra-cluster communications.

VI. Conclusion

The computational need increases with the development of new embedded applications. Besides, complex embedded applications become data-dependent and their execution time depends on their input data. For this reason, on-line task and data allocation is needed to optimize the architecture efficiency. This article deals with the problem of interconnecting hundreds of processors and memories in order to obtain an architecture for a multi-domain context. It focuses more specifically on the topology of the network-on-chip. The evaluation was done in terms of silicon area, performance and area efficiency. The first conclusion is that hierarchical networks present an area efficient solution. Moreover, the mesh, which is today a popular topology, has a bad area efficiency compared to hierarchical networks. The evaluation shows that the MultiCross has the smallest area and the MultiTorus is the most performant in hierarchical networks. The MultiCross presents the best area efficiency with local traffics, since it allows inexpensive communications between 4 clusters without disturbing other communications in the inter-cluster ring. The MultiCross is the best topology for a many-core architecture with a "smart" task and data allocation. The next step is to evaluate the power consumption which is also a criterion for embedded systems.

References

[1] W. Wolf, A. Jerraya, and G. Martin, "Multiprocessor System-on-Chip (MPSoC) Technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008.

[2] Semiconductor Industry Association, "International Technology Roadmap for Semiconductors," 2008.

[3] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th annual Design Automation Conference*, 2001, pp. 684–689.

[4] N. Bayer and R. Ginosar, "High Flow-Rate Synchronizer/Scheduler Apparatus And Method For Multiprocessors," *United States Patent*, no. 5,202,987, 1993.

[5] Tilera, "http://www.tilera.com/."

[6] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, P. Dubey, S. Junkins, A. Lake, R. Cavin, R. Espasa, E. Grochowski, T. Juan, M. Abrash, J. Sugerman, and P. Hanrahan, "Larrabee: A Many-Core x86 Architecture for Visual Computing," *IEEE Micro*, vol. 29, no. 1, pp. 10–21, Jan.-Feb. 2009.

[7] W. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, Mar 1992.

[8] J. Hu, U. Y. Ogras, and R. Marculescu, "System-Level Buffer Allocation for Application-Specific Networks-on-Chip Router Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2919–2933, Dec. 2006.

[9] E. Salminen, A. Kulmala, and T. Hamalainen, "On network-on-chip comparison," in *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, August 2007, pp. 503–510.

[10] D. Tutsch and M. Malek, "Comparison of network-on-chip topologies for multicore systems considering multicast and local traffic," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009, pp. 1–9.

[11] S. Murali and G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st Design Automation Conference*, 2004, pp. 914–919.

[12] A. Guerre, N. Ventroux, R. David, and A. Merigot, "Approximate-Timed Transactional Level Modeling for MPSoC Exploration: A Network-on-Chip Case Study," in *Proceedings of the 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, 2009, pp. 390–397.

[13] M. Palesi, D. Patti, and F. Fazzino, "Noxim." [Online]. Available: http://noxim.sourceforge.net

[14] SYNOPSYS, "http://www.synopsys.com/."

[15] G. Schelle and D. Grunwald, "Onchip Interconnect Exploration for Multicore Processors Utilizing FPGAs," in *Proceedings of the 2nd Workshop on Architecture Research using FPGA Platforms*, 2004.

[16] TSMC, "http://www.tsmc.com/."