Assessing Fat-Tree Topologies for Regular Network-on-Chip Design under Nanoscale Technology Constraints

D. Ludovici[§], F. Gilabert[‡], S. Medardoni[†], C. Gómez[‡],
M.E. Gómez[‡], P. López[‡], G.N. Gaydadjiev[§], D. Bertozzi[†]
[†] ENDIF, University of Ferrara, 44100 Ferrara, Italy.
[‡] Dept. of Computer Engineering, Universidad Politecnica de Valencia, Spain.
[§] Computer Engineering Lab., Delft University of Technology, The Netherlands.

Abstract

Most of past evaluations of fat-trees for on-chip interconnection networks rely on oversimplifying or even irrealistic architecture and traffic pattern assumptions, and very few layout analyses are available to relieve practical feasibility concerns in nanoscale technologies. This work aims at providing an in-depth assessment of physical synthesis efficiency of fat-trees and at extrapolating silicon-aware performance figures to back-annotate in the system-level performance analysis. A 2D mesh is used as a reference architecture for comparison, and a 65 nm technology is targeted by our study. Finally, in an attempt to mitigate the implementation cost of k-ary n-tree topologies, we also review an alternative unidirectional multi-stage interconnection network which is able to simplify the fat-tree architecture and to minimally impact performance.

1. Introduction

Networks-on-chip (NoCs) closely resemble the interconnect architecture of high-performance parallel computing systems [2]. For this reason, the interconnection topologies used in the early NoC prototypes can be traced back to the field of parallel computing. In particular, NoC architectures aiming at low latency communication, performance scalability and flexible routing selected fat-trees as their reference topology. The switch for the butterfly fat-tree network of [16] or the SPIN micronetwork [8] are examples thereof.

However, other topologies have found wider application in common NoC design practice so far, namely 2D meshes and even folded tori [3,4]. In fact, technology scaling to the nanoscale regime brings physical design issues to the forefront, such as the reverse scaling of interconnects. In this context, 2D mesh and torus topologies exhibit a grid-based regular structure which is intuitively considered to be matched to the 2D chip layout. In contrast, the higher wiring irregularity and the larger switch radix of most fat-tree configurations raise some skepticism about their practical feasibility. Moreover, instead of aiming strictly for speed, designers increasingly need to consider energy consumption constraints, and fat-trees are expected to pay the increased connectivity they provide with a significant area and power cost.

In spite of these concerns, constant attention has been devoted to tree-based topologies in the NoC community, proving their superior performance with respect to 2D meshes under different kinds of synthetic traffic patterns [1, 13]. However, these analysis frameworks are not able to be fully convincing and to impact NoC design practice in many senses. First, they often rely on abstract network simulators which cannot model the behaviour of any real architecture and sometimes make irrealistic assumptions, such as packet drop or TCP-compliant network transport protocols for on-chip communication. Second, most works really miss an in-depth physical analysis of layout feasibility and efficiency. Even when area synthesis results are provided, the impact of wiring congestion and interconnect delay on network performance is only assessed by means of analytical models. Also, the effectiveness of advanced design techniques such as clock or power gating or link pipelining is ignored.

This work aims at overcoming some limitations of previous fat-tree topology evaluations for NoCs, by primarily investigating the layout feasibility and the implications of physical mapping efficiency on system-level performance figures. This bottom-up approach to topology evaluation and selection reflects the design paradigm shift pushed by nanoscale technology, i.e., silicon-aware decision making at each level of the design hiearchy. The objective to perform a comprehensive layout-to-system level assessment of a fat-tree and its comparison with a 2D mesh forced us to necessarily restrict our exploration to a reasonable 16 core system. However, scalability insights into the connectivity of 64 core systems are provided as well.

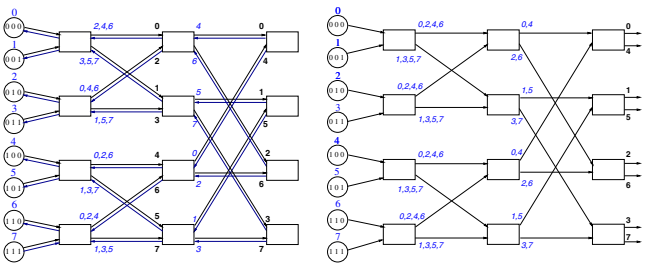
The fat-tree we consider in this paper is the commonly used k-ary n-tree, as it is defined in [12]. It allows to infer the topology by structuring multiple switches of constant size in a regular pattern and in a more compact layout. In spite of these properties, k-ary n-trees cannot avoid the trade-off between the increased connectivity they provide and the higher resource cost for it.

As a consequence, this paper also reviews an architecture optimization of fat-trees that aims at simplifying the switch architecture and saving network resources, while minimally impacting the high performance of these topologies. In practice, the proposed unidirectional multi-stage interconnection network (MIN) reduces the complexity of the downward phase, resulting in faster and more compact switches.

Network size	**16 Cores**			
Topology	4-ary 2-mesh	2-ary 4-tree	Unidir 2-ary 4-tree	Unidir 4-ary 2-tree
Switch radix	5	4	2	4
Total switches	16	32	32	8
Total Ports	64	112	64	32
Diameter	6	6	3	1
Bisect. Cut	8	16	8	8
Network size	**64 Cores**			
Topology	8-ary 2-mesh	2-ary 6-tree	unidir 2-ary 6-tree	unidir 4-ary 3-tree
Switch radix	5	4	2	4
Total switches	64	192	192	48
Total Ports	288	704	384	192
Diameter	14	10	5	2
Bisect. Cut	16	64	32	32

Table 1. Network topologies under test.

978-3-9810801-5-5/DATE09 © 2009 EDAA



(a) (b)

Figure 1. (a)A 2-ary 3-tree topology. (b)A RUFT derived from a 2-ary 3-tree.

2. Topologies

Table 1 shows some representative data for the studied networks. For each topology, half of the cores are processor cores and half represent their private memory cores. k**-ary** n**-meshes** are considered as the baseline topologies: a 4-ary 2-mesh for 16-core systems and an 8-ary 2-mesh in the 64 cores category. In all cases, we refer to them as **2D meshes**.

This paper focuses on on a specific implementation of **Fat-trees (FT)**: the k-ary n-trees (see Fig.1(a)), a parametric family of regular multistage topologies. k-ary n-trees are implemented by using n stages of identical switches, where k is the number of links of a switch that connect to the previous or to the next stage (i.e., the switch radix is $2k$). All the switches have the same number of ascending and descending links.

For fat-tree routing, this paper uses the deterministic routing algorithm presented in [9]. During the ascending phase, consecutive destinations are shuffled among the different ascending links of the switches, as in Figure 1(a): each ascending port is labeled in italics with the destination cores that are reachable through it. Also, Figure 1(a) shows in bold how destinations are distributed in the descending phase. As can be seen, each descending link is only used by a single destination. We analyze a 2-ary 4-tree for a 16-core system, and a 2-ary 6-tree for 64 cores.

Reduced Unidirectional Fat-Tree (RUFT) is a topology resulting from the simplification of a k-ary n-tree when using the deterministic routing algorithm above. RUFT was first introduced in [10] as an attractive conceptual topology scheme, without any implementation analysis. When using this deterministic routing algorithm, the whole descending phase can be reduced to a single long link that connects the output ports of the switches of the last stage with the input port of the corresponding destinations. In this way, switches become unidirectional and all packets must reach the last stage of the network (Figure 1(b)). Although the use of long links may compromise the feasibility of this topology, all the hardware resources related to the descending phase are reduced to these long links, simplifying the switch architecture. The resulting topology resembles an unidirectional butterfly, with a permutation of the reachable destinations from the last stage.

When evaluating RUFT, we consider two different topologies for each network size. The first ones are the unidirectional networks resulting from the simplification of standard k-ary n-tree fat-trees, that is, we analyze an unidirectional 2-ary 4-tree in the 16 cores category, while we analyze an unidirectional 2-ary 6-tree in the 64 cores category. These unidirectional networks have the same number of switches of the original fat-trees but, as can be seen in Table 1, the switch radix is reduced to one half.

This led us to consider an alternative RUFT implementation, denoted as **S(implified)-RUFT** hereafter, trading switch radix (which we make equal to that of the fat-tree) for the switch count. Thus, we define an unidirectional 4-ary 2-tree for the 16-core system and a 4-ary 3-tree for 64 cores. This way, the total number of switches is lower than that in the original fat-tree, as reported in Table 1.

Since our work considers topologies with a maximum switch radix of 5, the same switch count reduction could not be made in the original bidirectional fat-tree since this would have required 8x8 switches. The operating frequency slowdown would not be acceptable [7].

2.1 Floorplan design

This section discusses the criteria for floorplan design of the topologies under test. The xpipes-Lite NoC architecture has been used as an experimental NoC platform [17].

Processor and memory cores are replaced by non-routable hard obstructions of size $1mm$ x $1mm$. At first, we manually place the hard black boxes 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 Cadence SoC Encounter tool automatically places NoC cells without trespassing the fences. Fence size is set based on the report of the placement-aware logic synthesis.

The 2D mesh floorplan is straightforward (see Fig.2(a)) due to its regular grid structure matching the 2D silicon surface.

Things are more complex for 2-ary 4-tree FT (Fig.2(d)). The topology consists of 4 switch stages with 8 switches each. Our floorplanning strategy was to minimize wirelength between consecutive switch stages. For this reason, cores are clustered in groups of four and the connected switches (of the first and second stage) are placed in the middle of each cluster. The third switch stage is split into 2 subgroups and placed between the upper and lower clusters. Each subgroup serves its relative counterpart from the first and second stage. The last switch stage is located in the center of the chip. The presented layout exhibits equalized wirelengths between the second, the third and the last stage of switches.

The 2-ary 4-tree RUFT (Fig.2(b)) is a novel unidirectional fat-tree which has never been laid out before. This time, a switch belonging to the last stage is directly connected to the network interface of a core. This link is viewed in [10] as the intuitive weakpoint of the layout of this topology. To go around this problem, our floorplanning directive in this case is to minimize the wirelength of this critical set of links. Thus, switches from the last stage are positioned in the middle of each 4-core cluster. Obviously, also the first stage has to be close to the appropriate cores. Therefore, it is placed above and below the middle of the chip between two neighboring clusters, so to equalize the link length and keep the delay as homogeneous as possible on the wires of the first stage. As the third stage has to be connected to the last one and to the second one, two groups of switches belonging to the third barrier are placed at the left and at the right of the chip center. This also achieves an easy connection with the second stage, which is positioned in the center of the chip. An interesting property of the presented floorplan is that the link length is kept almost constant on a stage-by-stage basis.

Finally, the floorplan for a 4-ary 2-tree S-RUFT is illustrated in Fig.2(c). Although the number of switch stages is small (just 2), this is a challenging topology from a physical layout viewpoint. The problem stems from the fact that each switch of the first stage is directly connected to all the switches of the second stage. Moreover, a second stage switch is connected to network interfaces of cores, since this is again a unidirectional topology. Following the same floorplanning strategy of the 2-ary 4-tree RUFT, a switch from the last stage has to be placed in the middle of a 4-core cluster.

Unfortunately, in this case the switch has to be interconnected also to all the switches of the first stage, which should be necessarily placed in the center of the chip to equalize link length. This results in very long links going from each cluster to the switches at the center of the layout. As we will show later on, this dramatically impacts the achievable post-routing performance of this topology.

The 2D mesh is easily scalable to 64 cores, resulting in an 8-ary 2-mesh. Also the floorplan of the fat-tree (which becomes a 2-ary 6-tree FT) can be easily scaled with our strategy. In fact, a 64 node topology can be viewed as built up by four clusters of 16 cores with two additional switch stages connecting them to each other. The four clusters can be internally placed as previously described.

For the 2-ary 6-tree RUFT, again 4 clusters of 16 cores are formed. However, the main issue is that unlike the usual fat-tree, the center of each cluster is now occupied by the second stage of switches and not the fourth one (i.e., the last in the 16-core system). Therefore, the fourth stage being scattered on the edges of the chip, the connection with the additional switch stages is not equalized, leading to links of uneven lengths. A better floorplan could be obtained by customizing it for a 64-core system, which is not as intuitive as for a 16-core system and falls outside the scope of this paper.

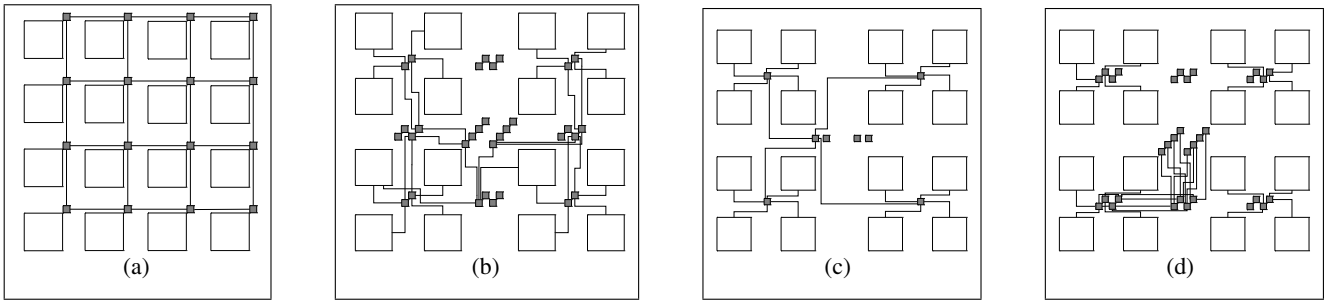


Figure 2. Floorplans of topologies under test. a) 4-ary 2-mesh b)2-ary 4-tree RUFT c)4-ary 2-tree S-RUFT d)2-ary 4-tree FT. Only the main wiring patterns are reported.

Topology	Max Arity	Post-. Synthesis	Post-place&route	Area 16 cores	Area 64 cores	Area overhead for Retiming
4-ary 2-mesh	5x5	0.9 ns	1.19 ns	802k $\mu m2$	3691k $\mu m2$	0%
2-ary 4-tree RUFT	2x2	0.6 ns	1.15 ns	795k $\mu m2$	4769k $\mu m2$	37%
2-ary 4-tree FT	4x4	0.8 ns	1.29 ns	1280k $\mu m2$	8076k $\mu m2$	19%
4-ary 2-tree S-RUFT	4x4	0.8 ns	2.1 ns	400k $\mu m2$	2400k $\mu m2$	112%

Table 2. Physical synthesis reports

Finally, the S-RUFT topology now becomes a 4-ary 3-tree (3 switch stages). Without going into further details, this floorplan turns out be as inefficient as that of the RUFT topology, and results in very long links between the last stage of switches and the network interface of connected cores.

3. Post-Layout analysis

Timing

We target a 65 nm low-power STMicroelectronics SVT technology library [15]. In all topologies, the network building blocks have been synthesized for maximum performance. The post-synthesis critical paths (ignoring place&route effects) are reported in Table 2, 3rd column. We found the critical path to be always in the switch and to reflect the maximum switch radix of the topology. Obviously, the lower switch radix of the 2-ary 4-tree RUFT results in a much shorter critical delay.

We then iterated place&route starting from the post-synthesis target frequencies. Timing closure was achieved at the post-layout speed reported in the 4th column of Table 2. Performance degradation turns out to be very significant, thus pointing out the critical role of interconnects. In fact, for all topologies (and even for the 2D mesh) the critical path goes through the switch-to-switch links (wire delay plus a few flow control gate delays). The impact of the link delay is evident, in that the critical delays of the topologies are differentiated by the longest link in that topology.

The 2-ary 4-tree RUFT wastes part of its speed with respect to the 2D mesh due to the use of longer links. In practice, the theoretical performance enhancement associated with a lower switch radix does not materialize after place&route, but has served as timing margin against physical degradation effects. The 2-ary 4-tree FT has incurred a lower degradation than the 2-ary 4-tree RUFT, but its post-synthesis performance was lower, therefore it ends up running even slower than the 2D mesh. Finally, for the 4-ary 2-tree S-RUFT the above effects are even more apparent due to the longer wires that are needed to connect a low number of switching resources sparse all around with each other. The lower area footprint is achieved at the cost of a remarkable 162% speed degradation after place&route.

Overall, in spite of a good post-synthesis frequency, thanks to the lower radix, MINs typically suffer from a more significant performance degradation after place&route due to their more intricate wiring compared to a 2D mesh. Hence, final performance cannot be predicted from early post-synthesis results in a straightforward way without accounting for interconnect-related effects.

When scaling the system to 64 cores, link pipelining was considered to mitigate the impact of link delay on system performance. Please observe that a pipeline stage is not just a simple retiming stage, but needs to take care of flow control and hence is a flow control stage with 2 slot buffers (see [14] for more details). We target a reasonable 750 MHz clock speed for the network, and place pipeline stages across links inducing timing violations at the target clock speed.

In the experiment, we conservatively inferred unrepeated AND retimed switch-to-switch links for 64-node networks, which is certainly a worst case for MIN network latency. The expected power and area share of wire repeaters in future systems [5] discouraged their use in this early work on topologies for large scale systems. Moreover, they are likely to incur placement concerns in NoC layouts. This was left for future work. For RUFT we found that, as expected by [10], a high number of stages (11) needs to be placed in the links connecting the last switch barrier to the network interfaces of destination cores. For the fat-tree, as expected by previous work (e.g., [6]), latency of links close to the root grows. However, due to the better scalability of fat-tree floorplan, the worst-case latency of the fat-tree is lower than RUFT.

Area

Table 2, 5th column, reports total floorplan cell area. The 2D mesh and the 2-ary 4-tree RUFT exhibit almost the same area, in that they have exactly the same number of I/O ports for a 16-core system, which are the ones that mainly determine switch area. The 2-ary 4-tree FT has a significant 60% area overhead with respect to the 2D mesh, which can be correlated with a 75% increase in the number of switch ports. Interestingly, although featuring the same number of switches, the 2-ary 4-tree RUFT achieves a significant area saving with respect to the FT since it employs lower-radix switches. Obviously, the 4-ary 2-tree S-RUFT exhibits the lowest area footprint with just 8 4x4 switches.

When looking at area projections for 64-core systems (based on post-synthesis area reports of network building blocks), we observe that area of MINs has increased much more than that of the 2D mesh. This is due to a higher increase factor of the switch count in MINs than in 2D meshes to interconnect a larger number of cores. This makes area footprint of 2-ary 4-tree FT hardly affordable and that of 2-ary 4-tree RUFT larger than the 2D mesh.

When we account for area overhead induced by retiming and flow control stages, Table 2 shows that an impressive 37% overhead is incurred by RUFT due to the poor scalability of its floorplan. Obviously, S-RUFT is the more penalized topology, and more than 50% of its area with 64 nodes might be occupied by retiming stages. Please consider that this is a worst-case scenario pointing out scalability issues, which can be mitigated by repeater insertion in the links or by custom floorplans for a given system size.

4. System Level Analysis

Based on the physical insights reported above, we anno-

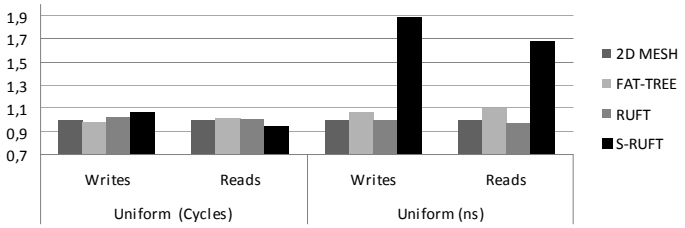


Figure 3. 16-core system. Normalized performance.

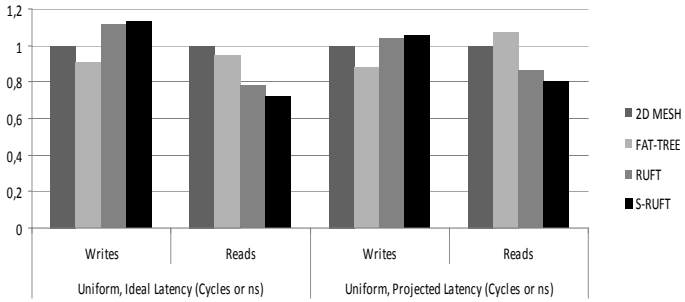


Figure 4. 64-core system. Normalized performance.

tated maximum clock speed of each topology to a cycle-accurate transaction-level (TL) simulator of the xpipesLite NoC architecture [11]. We model actual read and write OCP transactions at the network boundary. In uniform traffic, OCP transaction destinations are randomly chosen among all the available memories in the system. Transaction sizes are randomly chosen, betweeen 4 and 16 data burst beats. All networks were operated close to their saturation points.

Figure 3 shows performance results for a 16-core system under uniform traffic. Results are normalized to the 2D mesh. The left side of the figure shows total simulation time in clock cycles, while the right one shows the elapsed time in nanoseconds (accounting for the actual post-layout clock period from Table 2). Small differences in performance can be seen when execution cycles are evaluated. When only write transactions are considered, the best result is achieved by the fat-tree, that is the topology that provides the larger bisection bandwidth (see Table 1). On the contrary, topologies ensuring lower latency (like S-RUFT) can better handle read transactions. For this small scale system, fat-tree and 2D mesh have the same diameter and, because of the chosen mapping, the same average distance between every source-destination pair. This explains their performance balancing.

Unfortunately, when the real achievable speed is considered for each topology, S-RUFT becomes unusable. The fat-tree suffers from around 10% performance penalty with respect to the 2D mesh, in spite of its higher number of network resources, while RUFT proves an equivalent solution to 2D mesh. Given the lower wiring complexity of 2D mesh, this latter might be the reference solution for small scale systems.

When we move to a 64-core system, the global picture is totally different, as illustrated in Fig.4. Since link pipelining is now used, the same frequency of 750 MHz for each topology is enforced and the proper latency is assumed for the links of the MINs. For the sake of comparison, also the ideal case with 1 cycle latency on each link is reported.

Figure 4 now shows a neater performance differentiation between topologies. When considering only write transactions, the best topology is the fat-tree, reducing execution time by 10% over the 2D mesh. In fact, this traffic is bandwidth-intensive and the fat-tree provides the higher bisection bandwidth. Oddly, although both RUFTs provide in theory higher bisection bandwidth than the 2D mesh, this latter shows a better performance. RUFTs tend to confine congestion by constraining its spread across the network. Although this effect reduces global network congestion, backpressure propagation is such to reduces the rate at which communication-intensive cores can inject new packets, thus increasing overall execution time.

Read transactions cause a different scenario. Essentially, the traffic crossing the network is bounded. In fact, due to the blocking behaviour of the network interfaces on read transactions (until the response packet arrives), the maximum number of outstanding transactions is one for each processor core. It follows that performance is driven by the diameter of each topology. Under these conditions, S-RUFT becomes the best topology, with a reduction in total time of 28.1% over the 2D mesh, followed by RUFT with a reduction of 22.2%. Although fat-tree improves performance over 2D mesh, it is not enough to outperform both RUFTs.

These conclusions may change when projected link latencies are introduced. Their impact over topology performance depends on the type of traffic. For write transactions, Fig. 4 shows an improvement in performance of all MINs. The reason lies in the fact that retiming stages introduce more buffering as well. In the presence of bandwidth-intensive traffic, this feature results in less core blocking (due to backpressure from the network) and hence enhanced performance.

However, for read transactions network latency makes the difference in order to block network interfaces as short as possible. In this scenario, link latency degrades performance of all MINs, and fat-tree performs even worse than the 2D mesh. By combining repeater stages with retiming stages in each wire segment, performance can be brought back closer to the ideal one, but the power balance then becomes unclear and its analysis is left for future work.

5. Discussion and Conclusions

This work proves that fat-trees are feasible for on-chip networks from a physical design viewpoint. Unfortunately, for small scale systems, they are not able to capitalize on their better performance scalability yet. As the system size scales up, 2D meshes however suffer from poor performance scalability. Hence, the need for alternative topologies becomes more stringent. k-ary n-trees can provide that performance scalability, but at an impractical power and area cost. In this scenario, unidirectional MINs (like RUFT and S-RUFT) become attractive for their reduced power and area overhead. Unfortunately, their advantages cannot be easily materialized due to a more intricate physical design. Due to link pipelining, area and power cost of k-ary n-trees becomes prohibitive. Overall, we also prove that ignoring physical synthesis effects in high-level topology evaluations might result into largely misleading indications.

Acknowledgments

This work has been partially supported by the GALAXY European Project (FP7-ICT-214364) and partly by the Hipeac Network-of-Excellence (Interconnect Cluster).

6. REFERENCES

[1] Vu-Duc Ngo, H.Nam Nguyen, Hae-Wook Choi; "Analyzing the Performance of Mesh and Fat-Tree Topologies for Network on Chip Design", L.T.Yang et al. (Eds): EUC 2005, LNCS 3824, pp.300-310, 2005.
[2] L. Benini, G. De Micheli, "Networks on chip: a new SoC paradigm", IEEE Computer, vol. 35, no. 1, pp.70-78, Jan. 2002.
[3] S. Kumar et al., " A Network on Chip Architecture and Design Methodology", IEEE Computer Society Annual Symposium on VLSI, April 2002. pp. 105-112.
[4] Rijpkema, E.; Goossens, K.; Radulescu, A., "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip", Design, Automation and Test in Europe (DATE'03), Mar. 2003, pp. 350-355.
[5] D. Sylvester and K. Keutzer, "Getting to the bottom of deep sub-micron II: A global paradigm", Proc. IEEE Int. Symp. Physical Design, pp.193-200, 1999.
[6] H.Matsutani, M.Koibuchi, D.F.Hsu, H.Amano, "Three-Dimensional Layout of On-Chip Tree-Based Networks", Int. Symp. on Parallel Architectures, Algorithms and Networks, pp.281-288, 2008.
[7] Antonio Pullini et al., "Bringing NoCs to 65 nm", IEEE Micro 27(5): pp.75-85 (2007).
[8] A.Adriahantenaina, H.Charlery, A.Greiner, L.Mortiez, C.A.Zeferino, "SPIN: a Scalable, Packet Switched, On-chip Micro-Network", DATE'03, Embedded Software Forum, pp.70-73, 2003.
[9] C.Gomez et al., "Deterministic versus Adaptive Routing in Fat-Trees", CAC'07, as part of IPDPS'07, 2007.
[10] C.Gomez et al.; "Beyond Fat-Tree: Unidirectional Load-Balanced Multistage Interconnection Network", Computer Architecture Letters, June 2008.
[11] Francisco Gilabert, Simone Medardoni, Davide Bertozzi, Luca Benini, Mara Engracia Gmez, Pedro Lpez, Jos Duato; "Exploring High-Dimensional Topologies for NoC Design Through an Integrated Analysis and Synthesis Framework", Int. Network-on-Chip Symp., pp.107-116, 2008.
[12] F.Petrini, M.Vanneschi, "k-ary n-trees: High Performance Networks for Massively Parallel Architectures", Int. Parallel Processing Symposium, 1997, pp.87-93.
[13] P.P.Pande, C.Grecu, M.Jones, A.Ivanov, R.Saleh; "Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures", IEEE Trans. on Computers, Vol.54, no. 8, 2005.
[14] A. Pullini, F. Angiolini, D. Bertozzi, L. Benini, "Fault Tolerance Overhead in Network-on-Chip Flow Control Schemes", Proceedings of 18th Annual Symposium on Integrated Circuits and System Design (SBCCI) 2005, Florianpolis, Brazil, Sep 4-7, 2005, pp. 224-229.
[15] Circuits Multi-Projects, Multi-Project Circuits; http://cmp.imag.fr
[16] P.P.Pande, C.Grecu, A.Ivanov, R.Saleh "Design of a Switch for Network on Chip Applications", ISCAS'03, pp.V.217-V.220, Vol.5, 2003.
[17] S.Stergiou et al., "Xpipes Lite: a Synthesis Oriented Design Library for Networks on Chips", DAC, pp.559-564, 2005.