Butterfly vs. Unidirectional Fat-Trees for Networks-on-Chip: not a Mere Permutation of Outputs*

D. Ludovici[§], F. Gilabert[†], C. Gómez[†],
M.E. Gómez[†], P. López[†], G.N. Gaydadjiev[§], and J. Duato[†]

[†] Dept. of Computer Engineering, Universidad Politecnica de Valencia, Spain.
[§] Computer Engineering Lab., Delft University of Technology, The Netherlands.

Abstract

Bidirectional topologies are usually preferred over unidirectional ones. However, recent works have demonstrated that RUFT, a traffic-balancing routing algorithm on a Unidirectional Multistage Network (UMIN), can perform as well as a Bidirectional Multistage Network, but significantly reducing both implementation and operating costs. RUFT is a simplification of the k-ary n-tree topology, the most widely-used implementation of the Fat-Tree topology. RUFT resembles the classical unidirectional Butterfly topology, but with a different connection pattern between switches from the last stage and cores. This work provides a comparison in terms of performance and implementation costs between both topologies, RUFT and unidirectional Butterfly for on-chip interconnects. Our high level and post-layout analysis shows that RUFT is a much more convenient alternative than traditional Butterfly when laying out on silicon.

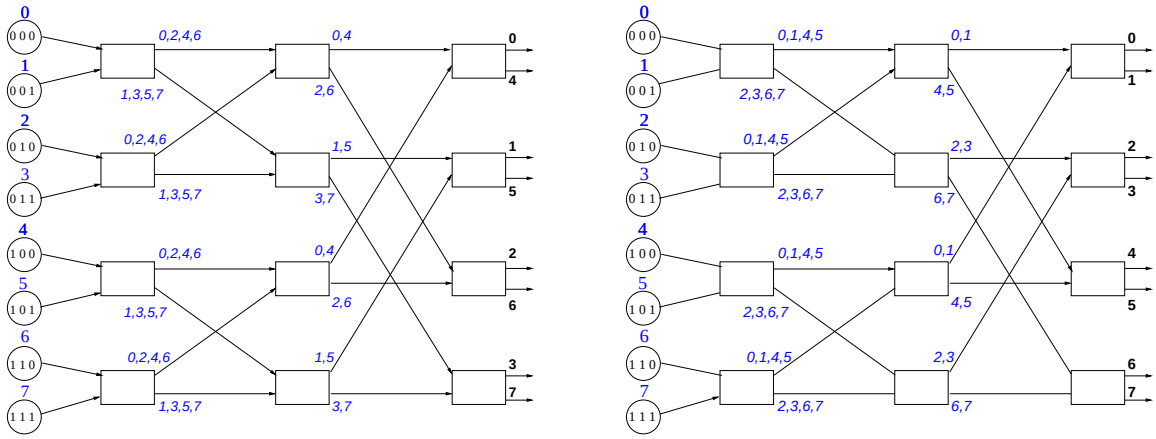
1 Introduction

The common trend in interconnect architecture research is to improve network performance at the cost of increasing physical implementation complexity. For instance in [1] the mesh topology is modified to increase its performance by halving the number of switches, but doubling the switch radix. Hence, switch complexity is doubled. Another example is shown in [11], where two H-Trees are combined to build a torus-like topology with a reduced network diameter, thus decreasing packet latency. For this, network interfaces are modified to allow packet switching, also increasing the number of network links and switches.

Nowadays, there is an emerging relevance of power consumption [12] and implementation costs [3]. As integration scale grows up, leakage power [8] and heat dissipation are becoming two mayor issues to consider when designing a new system. This has forced researchers to consider new design paradigms that prioritize costs and power consumption over performance. Power related problems become even more important in very restricted domains such as Networks-on-Chip (NoCs). NoCs are a new paradigm for scalable communication among the several cores located into a single chip [2]. The basic idea is borrowed from the off-chip network domain, and consists of a point-to-point switch based network with packet-switched communication that forwards packets to destinations through a certain number of intermediate hops. As aforementioned, NoCs have restrictions in terms of power consumption and cost (mainly silicon area and design cost), since total system cost and power consumption should not be seriously affected by the employed NoC.

Following this trend, a new topology, Reduced Unidirectional Fat-Tree (RUFT), was proposed in [7]. This topology is the result of combining a simplified Fat-Tree [9] with a load-balanced routing algorithm, halving the amount of hardware required while achieving a similar performance. These papers revisited unidirectional topologies, showing that they are competitive enough to be used in future high-performance systems. Although initially proposed for off-chip designs, RUFT was considered in [10] as a suitable NoC topology for Multi-Processor Systems-on-Chip designs (MPSoCs). This work concluded that RUFT is a power-wise topology whose performance scales very well as the number of cores in the system grows. RUFT resembles the classical unidirectional Butterfly topology [5], as the

*This work was supported by the Spanish MEC under Grant TIN2006-15516-C04-01 and by CONSOLIDER–INGENIO2010 under Grant CSD2006-00046.



(a) A RUFT derived from a 2-ary 3-tree.

(b) A 2-ary 3-fly.

Figure 1. Topologies with switches of arity 2 and 3 stages. Switch ports show their reachable cores.

only difference is the connection pattern between the switches from the last stage and the cores. This work points out that although apparently similar, RUFT and unidirectional Butterfly (Butterfly for now on) cannot be considered such when the physical implementation perspective is taken.

The paper is organized as follows. First, Section 2 presents a brief description of both topologies. In Section 3 a high level comparison between the investigated interconnection networks is discussed. Section 4 points out physical design issues and a post-layout analysis is provided in Section 5. Finally, conclusions are drawn in Section 6.

2 Topologies Analysis

This section provides a brief description of the investigated topologies: RUFT and Butterfly. **Reduced Unidirectional Fat-Tree (RUFT)** is a topology derived from the k-ary n-tree [13], which is the most widely-used implementation of Fat–Tree. Fat–trees are a particular type of a set of topologies known as multistage interconnection networks (MIN). In MINs, switches are deployed in a set of stages. Each switch can only be connected to switches belonging to their previous or to their next stage. In bidirectional MINs, cores are attached to the switches of the lowest stage. k-ary n-trees are implemented by using identical switches of a fixed radix. The number of stages is n and k is the arity or the number of links of a switch that connect to the previous or to the next stage (i.e., the switch radix is $2k$). Notice that k-ary n-trees are bidirectional MINs. A k-ary n-tree connects $N = k^n$ cores using nk^{n-1} switches and $2nk^n - k$ unidirectional links.

RUFT is a topology obtained by simplifying k-ary n-trees. This simplification is granted by the load balancing provided by the deterministic routing algorithm for k-ary n-trees proposed in [6]. In this way, k-ary n-trees become k-ary n-RUFTs, an unidirectional MIN with the same connection pattern between switches as a k-ary n-tree. As switches become unidirectional, the amount of resources required for their implementation is halved, i.e., switch radix is k. The main difference is that switches belonging to the last stage are directly connected to the cores, according to the paths provided by the routing algorithm in which each switch of the last stage only receives packets destined to the k destinations that can be reached through its output ports. For example, Figure 1(a) shows a RUFT with switches of arity 2 and 3 stages. As it can be seen, switches of the last stage only receive packets destined to two destinations. An unidirectional reduced k-ary n-tree is able to connect $N = k^n$ processing nodes using nk^{n-1} unidirectional switches and nk^n unidirectional links. The main drawback of UMINs is the length of the links that connect the switches from the last stages to the cores. Those links may become too long, but as demonstrated in [10], they do not compromise RUFT feasibility. The resulting topology resembles an unidirectional butterfly with a permutation on the reachable destinations from the last stage.

Butterfly is another type of unidirectional multistage interconnection network. A k-ary n-fly is implemented by using unidirectional switches of radix k organized in n stages, as in RUFT. A k-ary n-fly is also able to connect $N = k^n$ processing nodes using nk^{n-1} switches and $nk^n - k$ unidirectional links. An example

Topology	Max. Time	Min. Time	Rel. Diff.
2-ary 4-RUFT	4617231	4567331	1.1%
2-ary 4-fly	4639669	4585143	1.2%
4-ary 2-RUFT	4308401	4266017	1.0%
4-ary 2-fly	4345453	4294243	1.2%

Table 1. Maximum and minimum execution times in cycles for all processor-memory placements in several 16-core topologies.

Topology	Time
2-ary 6-RUFT	1188676
2-ary 6-fly	1215068
4-ary 3-RUFT	1096946
4-ary 3-fly	1103594

Table 2. Execution time in cycles for 64 cores topologies.

of this topology is shown in Figure 1(b). The main difference between RUFT and Butterfly is the connection pattern between last stage switches and cores. Figure 1 shows in bold black the destinations reachable through the output ports of the switches from the last stage in both topologies. As it can be observed, this difference forces routes through Butterfly to completely differ from the ones provided by RUFT. As in the case of RUFT, Butterfly topologies also present longer links between switches of the last stage and cores. In Section 4 we will analyze the effect of those links over the feasibility of the topology.

3 Performance Evaluation

In an MPSoC there are two types of cores: processors and memories. We define *core placement* as the policy used to map processors and memories on a given topology. The core placement directly influences the communication pattern of a topology and its overall performance. In this section we analyze the impact of the core placement on the overall performance of the two topologies under test. We will show that differences between both topologies do not depend on the core placement policy employed. In this way, we assure that, from a performance point of view, the choice of the best topology is not affected by the core placement.

To assess the real impact of the core placement over performance, we performed a parametric in-depth exploration of all the possible processor-memory placements in a 16-core NoC. In this exploration, we used the cycle-accurate transaction-level simulator proposed in [10], based on the Xpipes-Lite NoC architecture [14]. For each of the possible placements, 250000 transactions were executed according to an uniform distribution of transaction destinations. Furthermore, the network was working at the same injection rate, near the saturation point. Two different configurations of 16-core topologies were considered. The first configuration is composed by 32 switches of arity 2 organized in 4 stages, that is 2-ary 4-RUFT or 2-ary 4-fly. The second one is composed by 16 switches of arity 4 organized in 2 stages, that is 4-ary 2-RUFT or 4-ary 2-fly. Table 1 shows the results of the topology exploration, measuring the execution time of the test in cycles. The 2nd and 3rd columns represent the maximum and minimum achieved execution time of all configurations, respectively. The 4th column contains the relative difference between both times with respect to the minimum. As can be observed, the impact of core placement over performance is negligible for all configurations, since the maximum relative difference is 1.2%. Regarding the comparison between both topologies, there is not a great difference in performance. When considering only the best case (the minimum execution time), the Butterfly has a 0.6% and a 0.4% higher execution time than RUFT for the 2-ary and the 4-ary configurations, respectively. This is a negligible difference.

Finally, in order to validate the previous results, we extend this analysis to a larger system. We consider two different 64-core configurations. The first configuration is composed by 192 switches of arity 2 organized in 6 stages, that is 2-ary 6-RUFT or 2-ary 6-fly. The second one is composed by 48 switches of arity 4 organized in 3 stages, that is 4-ary 3-RUFT or 4-ary 3-fly. Table 2 shows the total execution time in cycles for each 64-core configuration. Since the exploration space of possible core mappings is overly large, we have only considered an intuitive core placement policy: interleaving processors and memories. The simulations were executed under the same conditions as the previous experiment. The Butterfly has a 2.2% and a 0.6% higher execution time than RUFT for the 2-ary and the 4-ary configuration respectively. As it can be seen, the gap in performance between both topologies is again negligible. This result is certainly determined by the homogeneous and well distributed traffic generated by each processor cores. Implications of more irregular traffic patterns on topology performance and differentiation is work underway. We now delve into physical design issues, aiming at assessing whether the topologies under test are equivalent also when it comes to silicon implementation.

Topology	Max Arity	Post-. Synthesis.	Post-place&route	Area 16 cores	Tot. Wire Length
2-ary 4-fly	2x2	0.6 ns	4.0 ns	603k μm2	9605 mm
2-ary 4-RUFT	2x2	0.6 ns	1.15 ns	795k μm2	8370 mm

Table 3. Physical synthesis reports

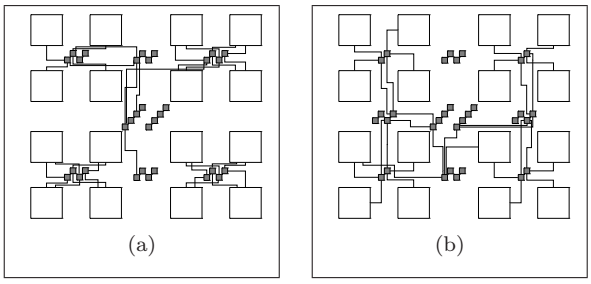


Figure 2. Floorplan of topologies under test: a) 2-ary 4-fly b)2-ary 4-RUFT. Only the main wiring patterns are reported.

4 Physical Design

Synthesis time constraints forced us to limit the physical design to 16 core systems. We implemented the 2-ary 4-RUFT and 2-ary 4-fly leveraging network building blocks of the Xpipes-Lite NoC architecture. Initial topology specification is in RTL-equivalent SystemC. The backend synthesis flow leverages standard industrial synthesis and physical design tools. Our work targets a 65nm low-power STMicroelectronics SVT technology library [4]. For a comprehensive discussion about the backend flow, see [10]. Next, the criteria utilized to floorplan the design of the topologies under test is discussed.

The 2-ary 4-RUFT (Fig.2(b)) is an unconventional fat-tree where links are unidirectional. A switch belonging to the last stage of this topology is directly connected to the network interface of a core. This link is viewed in [7] as the intuitive weakpoint of the layout of this interconnection network. In order to overcome this problem, our floorplanning directive is to minimize the wirelength of such a critical set of links. In this respect, cores are clustered in groups of four and switches from the last stage are positioned in the middle of each 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 stage are placed at the left and at the right of the chip center. This also achieves an easy connection with the second stage, whose switches are 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.

The butterfly fat-tree (2-ary 4-fly) is a popular interconnection network that has been widely investigated in literature. In spite of its good theoretical properties, the butterfly is a challenging topology when laid out on a silicon surface. In this respect, the best known approach [15] utilizes a floorplan strategy where switches of different stages (but belonging to the same row) are clustered together and placed nearby their computational unit. This intuitive approach exhibits sensible scalability issues due to the fact that floorplan is realized in a vertical fashion with consequent link length penalty. In order to overcome this limitation, we adopted a new floorplan approach in line with the one presented for the 2-ary 4-RUFT topology. Cores are clustered in groups of 4 units. Each cluster gathers consecutive computational units (i.e., $0 \ldots 3$) and the correspondent switches from the first and last stage are placed in the middle of the cluster. With our approach, link length has been minimized as much as possible compared to traditional butterfly floorplans. Nonetheless, long wires connecting switches from 3rd and 4th stage represent an intrinsic limitation that implies a considerable performance drop after place&route (Table 3, 4th column).

5 Post-Layout Analysis

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 3, 3rd column. We found the critical path to be always in the switch. Consequently, we 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 3. Performance degradation turns out to be very significant for the 2-ary 4-fly, thus pointing out the

critical role of interconnects. In fact, for both topologies, the critical path goes through the switch-to-switch links. While data/flit wires are sampled at the input and output port of the switch, flow control wires go through the FSM of the flow control stages at switch I/O. As a consequence, these control wires go through logic gates whose delay adds up to the link delay, determining the critical path. The impact of the link delay is evident, in that the critical delays of the topologies are differentiated by the longest link in the topology itself. In general, we assessed a higher wiring overhead (last column in Table 3) for the 2-ary 4-fly due to its longer links that connect switches from the 3rd and 4th stages.

From the area viewpoint, both topologies present the same I/O buffering requirements. Therefore, a similar area footprint was expected. In this respect, we found an interesting consequence of the much higher post-layout frequency of 2-ary 4-RUFT: in order to meet its stringent timing requirements, the place&route tool utilizes larger buffers to speed up signals thus increasing significantly the area overhead. Obviously, should we target the same affordable speed for both topologies, this effect would vanish, thus resulting in more or less the same area for the two solutions.

6 Conclusions

In this paper, we proposed RUFT as a feasible and more powerful alternative to traditional butterfly topologies for NoCs. We analyzed the impact of all possible core-placement alternatives detecting a negligible worst-case relative difference of 1.2% in terms of performance. The same conclusions for systems with a reduced number of stages (4-*ary* 2-RUFT and 4-*ary* 2-fly) have been drawn. As the number of stages increases, RUFT becomes a slightly more attractive solution with a 2.2% performance improvement with respect to a 6-stage butterfly, although the difference is still marginal. We then performed an in-depth physical level analysis for 16 cores systems showing that RUFT best suits 2D silicon layout compared to butterfly. This enables considerable higher post-layout frequencies due to shorter and better equalized link lengths. Therefore, whenever the execution cycle performance of both topologies can be retained the same (as in the traffic scenario analysed in this paper), RUFT should be the preferred solution because of its structure better matches layout constraints. Future work includes the exploration of a broader range of traffic patterns to assess the behaviour of RUFT and butterfly topologies and the analysis of their floorplan scalability.

References

[1] J. Balfour and W. J. Dally, "Design Tradeoffs for Tiled CMP On-chip Networks", *Proc. Annual Int. Conf. on Supercomputing*, 2006.

[2] L. Benini, "Networks on chip: a new paradigm for systems on chip design", *In Proc. of Conf. on Design, Automation and Test in Europe*, 2002.

[3] F.Boekhorst, "Ambient Intelligence, the Next Paradigm for Consumer Electronics: How will it Affect Silicon?", *ISSCC 2002* , February 2002.

[4] "Circuits Multi-Projects, Multi-Project Circuits", http://cmp.imag.fr

[5] J. Duato, et al., *Interconnection Networks. An Engineering Approach*, Morgan Kaufmann, 2004.

[6] C. Gómez, et al., "Deterministic versus Adaptive Routing in Fat-Trees", *in Proc. Workshop on Communication Architecture on Clusters (CAC'07)* , as a part of IPDPS'07, March, 2007.

[7] C. Gómez, et al., "Beyond Fat–tree: Unidirectional Load–Balanced Multistage Interconnection Network", *IEEE Computer Architecture Letters*, 2008.

[8] N. S. Kim, et al., "Leakage current: Moore's law meets static power", *Computer*, 2003.

[9] C. Leiserson, "Fat-trees: Universal Networks for Hardware Efficient Supercomputing". *IEEE Tran. on Computer*, 1985.

[10] D. Ludovici, et al., "Assessing Fat-Tree Topologies for Regular Network-on-Chip Design under Nanoscale Technology Constraints", to appear in *Proc. of Conf. on Design, Automation and Test in Europe*, 2009.

[11] H. Matsutani, et al., "Performance, Cost, and Energy Evaluation of Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network", *Proc. of IEEE International Parallel and Distributed Processing Symposium*, 2007.

[12] Trevor Mudge, "Power: A First-Class Architectural Design Constraint",*IEEE Computer*, 2001.

[13] F. Petrini and M. Vanneschi, "k-ary n-tress: High Performance Networks for Massively Parallel Architecture", *IEEE Micro*, 1995.

[14] S.Stergiou et al., Xpipes Lite: a Synthesis Oriented Design Library for Networks on Chips, *DAC*, 2005.

[15] T. T. Ye and G. De Micheli, "Physical planning for on-chip multiprocessor networks and switch fabrics", *In Proceedings of the Application-Specific Systems, Architectures and Processors (ASAP)*, 2003