I.J.E.M.S., VOL.3(3) 2012: 396-401 ISSN 2229-600X

INTERNATIONAL JOURNAL OF ENGINEERING AND MANAGEMENT SCIENCES

© 2004-2012 Society For Science And Nature(SFSN) All Rights Reserved

www.scienceandnature.org

# DESIGN EXCHANGE OF THE SINGLE CHIP MULTI-COMPUTER NETWORKS

**Chandra Shekhar,**
Research Scholar, Singhania University, Pacheri Bari, Jhunjhunu, Rajasthan

**ABSTRACT**
A critical issue for multiprocessor SOCs that has received very little attention is the question of how the removal of chip boundaries might change the design of the multiprocessor system, including both the design of the components as well as how they are integrated. Since it was the fact that an entire processor could be placed on a single chip—thereby removing long intraprocessor signaling delays--that spurred the microprocessor revolution, it seems possible that removal of packaging boundaries might cause a similar paradigm shift with multiprocessors. At the very least, there is likely to be a major shift in the design space as the distribution of propagation delays and the ratio of propagation delay to switching time is fundamentally altered. In this paper, a large number of parallel processor network router designs are made and compared in terms of cycle-by-cycle performance, cell area, and cycle time. Although these designs are directed to parallel processor networks for use on a single substrate, they are also applicable to other design situations. The designs are based on a single basic design framework with a large number of different options. By comparing the designs that result from these options, a designer can make deductions about the relative cost and performance benefits of particular options and make decisions about which design to use for his particular situation

**KEYWORDS:** SOC, Signaling Delay, Design Space, Distribution of Propagation Delay.

## INTRODUCTION

A recent trend in high performance computing (HPC) has been towards the use of parallel processing to solve computationally-intensive problems. Several parallel architectures, which offer corresponding increases in performance as the number of processors is increased, have been designed in the last few years. Nowadays, with the enormous transistor budgets of 45-nm and 32- nm technologies on a silicon die, it is feasible to place large CPU clusters on a single chip (System on Chip, SoC) [15] [16] allowing both large local memories and the high bandwidth of on-chip interconnection. Using this chip-scale multiprocessing, the number of processors on a chip may in the near future scale to dozens or hundreds, depending on their complexity. The basic requirement for building such a SoC turned out to be the low power consumption, in order that system parts could be close together and communication time would be thus minimized. For the same reason, the CPU cores should be simple and processing nodes should be interconnected as effectively as possible.

Buses and point-to-point connections are the main means to connect the components. Buses can efficiently connect 3-10 communication partners but they do not scale to higher numbers. Even worse, they behave very unpredictably, as seen from an individual component, because many other components also use them. A second problem comes from the physics of deep submicron technology. Long, global wires and buses become undesirable due to tight timing constraints and skew control, high power consumption and noise phenomenon.

As a consequence, in 1999 several research groups started to investigate systematic approaches to the design of the communication part of SoCs. This research area has been called Network on Chip (NoC) [14][15][17]. A NoC is constructed from multiple point-to-point data links interconnected by switches (routers), so that messages can be relayed from any source module to any destination module over several links by making routing decisions at the switches. Although NoCs can borrow concepts and techniques from the well-established domain of computer networking, it is impractical to blindly reuse features of "classical" computer networks and symmetric multiprocessors[14]. In particular, NoC switches should be small, energy-efficient, and fast. The routing algorithms should be implemented by a simple logic, and the number of data buffers should be minimal. These requirements have converged on the use of pipelined, distance-insensitive wormhole (WH) message switching and source-based routing algorithms.

There are several examples of NoCs that have found applications in the commercial sphere at the present time. One of the leaders in this area is Tilera Corporation that has introduced Tilera Tile- Gx processor family with 16 to 100 full-featured processing cores interconnected by a 2D mesh. Other compact high performance systems were produced by SiCortex, Inc. This vendor has offered highly compacted systems based on unidirectional Kautz networks and scalable up to 5832 cores with only 20kW of power consumption. Also IBM has brought its solution called IBM

Cell broadband engine on the market. This multimedia chip integrates a Power PC processor and 8 SPE elements interconnected by a ring NoC called Element Interconnection Bus. The chip has found many commercial applications not only in the gaming industry , but also in the second most powerful supercomputer in the world called Roadrunner. Similarly, Intel is working on its own NoC solution under the project Intel Tera-scale research program (Larabee). The goal of the project is to design a NoC with 1TFLOP performance composed of about 80 cores interconnected by either a mesh topology or a hierarchical ring topology. The small scale system can be also based on common AMD Opteron and Intel Nehalem architecture processors. The Network on Board is then created using HyperTransport or QuickPath links into bidirectional rings or hypercubes (AMD Direct Connect Interconnection). And this is only the beginning of the NoC area. Many other implementations will certainly come into existence in the next years.

In order to be able to utilize the performance of such a SoC, the parallel programming paradigms have to be taken into account. Provided that computation times of executed tasks are known, as is usually true in case of application-specific systems, the only thing that matters in obtaining the highest performance are durations of various collective communications. Some embedded parallel applications, such as network or media processors, are characterized by independent data streams or by a small amount of inter-process communications. However, many general-purpose parallel applications display a bulk-synchronous behavior: the processing nodes access the network according to a global, structured communication pattern. Examples of collective communication (CC) patterns include broadcast, in which a message is sent from one process to all the other processes in a group; global combine, in which a global operation, such as maximum or sum, is performed on a distributed set of data items; and barrier synchronization, in which every member of a set of processes must reach a given point in its execution before any member can proceed. The growing interest in the use of collective routines is evidenced by their inclusion in the Message Passing Interface (MPI) standard and by their increasing role in supporting data-parallel languages. Many existing SoCs do not support collective operations in hardware. In these environments, collective operations must be supported in software by sending multiple point-to-point messages. Such implementations are termed unicast-based and typically are implemented as a sequence of synchronized steps, each of which involves the sending of one or more messages among processes. However, in situations where many messages exist in the network concurrently, a large internode distance can lead to contention among messages. Therefore, unicast-based collective operations, which typically involve many messages, should be designed so that they not only minimize the number of message-passing steps, but also minimize or eliminate contention among the constituent messages.

At present, many different universal topologies of interconnection networks are in use and other application-specific ones can be created on demand. While the time complexity of certain communication patterns has a lower bound given by a particular interconnection, finding a sequence of communication steps (a schedule) approaching this limit is more difficult, and in some cases, such schedules are not known as of yet.

Naturally, many projects have addressed the design of fast collective communication algorithms for wormhole-switched systems in recent years. Since any data loss is not acceptable in NoC, the deadlocks, livelocks and starvations, even links/node overloads, have to be prevented in such schedules. Hence, many approaches have analyzed the structure and properties of underlying NoC topology and communication pattern with the aim of designing contention-free communication schedules that attain the lower bound of time complexity of given CC patterns. Unfortunately, these schedules are not general at all, and only work for a few regular topologies like hypercube or square mesh/tours even then in only a couple of possible instances. Another idea is to design some families of parameterized algorithms that can be tuned to perform well on different architectures under various system conditions. Unfortunately, this kind of CC schedules is not optimal in most cases, and moreover they are restricted by other parameters of the NoC such as port model, minimal routing strategy, symmetry of the network and so on.

With an increasing number of novel NoC topologies (e.g. spidergon, Kautz, fat topologies) a hunger for a general technique capable to produce optimal or near optimal schedules for an arbitrary network topology and a given CC pattern steadily grows. The designed schedules could serve for writing high-performance communication functions for a concrete topology. Consequently, these functions could be included into, for example, a well-known OpenMPI library to accelerate given CCs and prevent data losses.

## PROBLEM AND MOTIVATION

The problem we address in this paper is the design of a network appropriate for a single chip multiprocessor. This problem is important because the changing ratio of switching to propagation time is likely to have a significant effect on the relative benefits of various design choices. The single-chip assumption is also interesting from a methodological standpoint as it allows us to use standard workstation EDA tools for accurate circuit-level simulations. When these are combined with cycle-level simulations as we do in this paper, it becomes possible to examine a large number of design alternatives without compromising accuracy.

## LITERATURE REVIEW

Although there has been extensive work in the area of switch design and multiprocessor networks, relatively few studies include hardware costs, either in their effect on chip area or on operating frequency. Of the studies that do pay attention to cost, even fewer do so quantitatively, or for more than one particular implementation of a design.

An exception is Chien [1, 6], who has performed a detailed hardware cost and operating frequency analysis for a particular class of routers in terms of the number of inputs or outputs, number of virtual channels and routing freedom. But, this analysis does not account for some of the options we would like to consider such as virtual cut-through routing, unidirectional vs. bidirectional switches, the varying buffer sizes, dynamic buffer sharing between lanes and does not address the network performance achieved by varying these parameters. Another gap is that virtual cut-through switching has not been explored to nearly the extent of wormhole switching. This is especially important since even a preliminary glance at the problem of low-cost networks, points to virtual cut-through as a way of cutting down on the number of virtual channels and thereby getting more switch for your silicon. Some that do, Duato et al [12], do not perform a hardware cost analysis or have a design that is not suited for our particular environment.

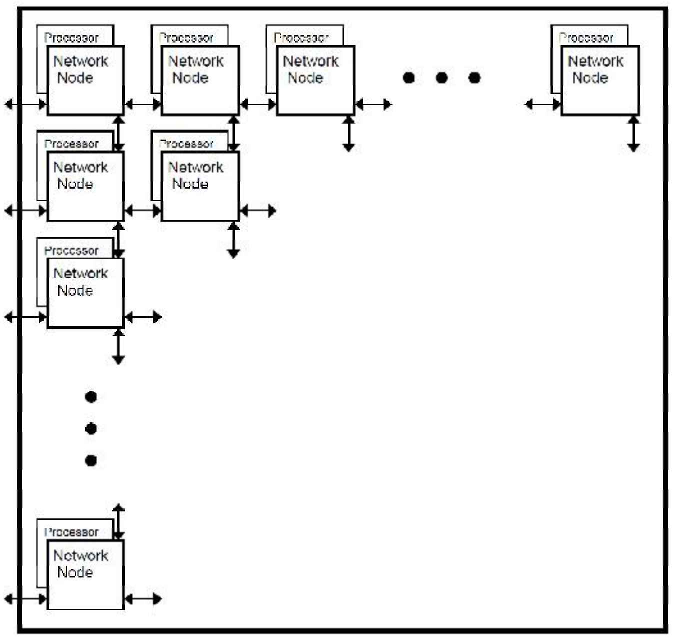
We are designing for an environment where the node to node delay is very small because the nodes are on the same chip and are placed adjacent or very near their communication neighbors. Little work has been done for this environment. (Even Dally's Torus Routing Chip [10] had off- chip delays between nodes and couldn't operate faster than the off-chip and wire delays.) The Abacus SIMD Array [4] is one that uses network nodes on the same chip, but this network is much too simple for our purposes.

Much work is done for nodes that are more featured and complex than we use. The SP2 is an example. Its design is more complex than we use and that makes the operating frequency lower than we are able to achieve. The SP2 design uses this complexity to increase its performance for its particular environment and this is good. The reason is that the SP2 has very long node to node delays and even with extensive pipelining one could not possibly achieve the small cycle times we do for our design. So the additional complexity probably does not affect the operating frequency.

## THE BASIC MODEL

We envision these multiprocessors on a chip as having dozens of nodes on a silicon substrate, increasing to hundreds as technology allows. Such a chip would have an array of processor node pairs, each with a connection to near neighbors. All processor-to-processor data transfers take place in the network nodes using the following scenario: i) the source processor transfers the data packet to its associated network node, ii) the packet is transferred from network node to network node until it reaches the destination network node, and iii) the packet is then transferred from the destination network node to its associated processor. All processor instruction and control transfers are outside the network and we do not consider them here. The design of the off-chip connections depends on the details of the design external to the chip and the particular pin constraints of the chip's package. Since this is very design specific we do not consider the details here, except to note that the off-chip connections are allowed for in our designs.

While designs with $(2^N)^2$ nodes, and array of $2^N$ on a side, may be easier to program than other sizes, our designs do not depend on any particular size. But, our simulated performance is only applicable for designs with the same number of nodes on each side. The following figure shows at a high level how such a chip might be organized.



Because the processors and network nodes are on a single chip and arranged in a 2 dimensional array with short interconnections, the data transfer between one node to another is very fast. This, combined with the network nodes' simple and efficient design, presents the opportunity for a design with a short cycle time, and therefore, high performance. The regular organization of the chip simplifies the design and testing of the actual implementation. We assume small fixed size packets and deterministic routing. The routers' designs are allowed to take advantage of this particular environment whenever possible. The resulting routers are simple, low-cost, and suitable for parallel processor systems on a single substrate.

We generally assume input buffering, but to be sure that we are not going down the wrong path, we also study a design that has output buffering. Output buffering has higher performance when considering network throughput per cycle, but the design is more complex and managing the buffer at the output results in a much longer critical path. To see whether the increase in throughput per cycle compensates for the increase in cycle time we evaluate a network with nodes with output queuing, similar to the IBM SP2 design, but which still taking advantage of our particular environment.

## METHODS

Because our focus is how to build an embedded network for single chip coprocessors, we concentrate on low-cost, cost-effective mechanisms: The routing is deterministic dimension order. The topology is either a 2D mesh or torus. A single physical channel exists between nodes per direction per dimension. The packets are small and have fixed size, 6 and 24 flits in our experiments. These sizes approximate transfers of single words or cache lines.

Even so, there are still a large number of router design options to consider: These include:

- Whether the switching mode is wormhole (WH) or virtual cut-through (VCT)
- Whether the connections are unidirectional or bidirectional
- Whether the crossbars are full or cascaded
- The optimal number of virtual lanes per channel.
- The optimal FIFO buffer size.
- Whether to share FIFO buffer space for more than 1 lane in the same RAM.

We evaluate the design alternatives in terms of both the cost in chip area and the performance in terms of effect on network capacity and operating frequency. This requires using three different methods, which are now described.

To determine network capacity, we use a cycle-driven simulator and simulate a network of reasonable size, 8 by 8. We determine network capacity by assigning injectors a fixed capacity (load) and run enough cycles to determine if the network saturates with this load. We believe our method is more representative of real systems, even if it requires more computer time per design. A simulation run consists of a single network type / load / routing pattern / packet size combination. After performing multiple runs with different loads, we determine the network capacity in terms of flits/node/cycle. The routing patterns we use for performance evaluation are standard synthetic patterns: random, hot-spot, and near-random.

To determine cell area we represent the designs in Verilog High-level Design Language, HDL, then perform logic synthesis targeting LSI Logic's G10-p 0.18 micron technology. The logic synthesis package we use, from Synopsys, is very good at deriving a design with optimum cell area. Not all designs are evaluated completely; we take advantage of the hardware design's modularity to evaluate some of the modules separately. For example, the number of ports in the output queue can be evaluated for many design points and a formula for cell area based on the number of ports generated. These formulas are used in the designs instead of generating a complete Verilog HDL for each design option. Combining these formulas yields a cell area for each design option.

To determine cycle time, the designs are evaluated and critical paths determined manually. The logic design involving these critical paths is done by hand and timed using gate delay for the target technology. We do this manually because the logic synthesis package we use, from Synopsys, is not very good at deriving a design with optimum timing.

## CYCLE-LEVEL NETWORK SIMULATION

We use a register transfer level simulator to measure capacity and latency. Since our designs are synchronous, the simulator can be cycle-driven and validation with the hardware model is simple. We assume an internode routing time of one cycle.

We use three communication patterns: random, hot-spot, and random-near. For the random load, all destinations are equally probable. For the hot-spot load, we use a similar scheme as described in [3], four destinations are four times as likely as the others. For the near-random load, the coordinates of the destination are chosen independently for each dimension, likelihood of a destination is inversely proportional to its distance from the source. We use two packet sizes, 6 flits and 24 flits. These sizes were chosen to represent the transfer of a word and a small cache line transfer, respectively. Also, the smaller packets typically span a small number of nodes in transit while the larger packets span the entire path through the network. Together they offer two qualitatively different workloads.

The load is presented in terms of the expected number of flits injected per node per cycle. We prefer this measure to that of fraction of overall capacity in that it forces separate evaluation for each communication pattern. Our primary choice of performance measure is network load capacity. One reason for this is its intrinsic importance. The other is the observation, repeated numerous times, that the switching mode, buffer size, number of lanes, and other parameters which are the objects of this study all have only a small effect on latency until saturation is approached and then the effect is quite predictable [12]. The latency/load graph in Figure 2 depicts this effect. There are three `bundles' of series: one each for unidirectional WH, bidirectional mesh WH, and bidirectional torus WH. The bundles are formed around configurations with matching internode bandwidth.

The bidirectional and unidirectional VCT series, if shown, would be superimposed on their WH counterparts. Within each bundle, the capacity measure, which indicates where the series becomes vertical, is therefore sufficient to characterize the particular series. A run consists of a single combination of network, load, communication pattern, and packet size. A run terminates either after 80,000 cycles or when the network goes into saturation. The first 50,000 cycles are used for transient removal; thereafter latencies are recorded. Generally, steady-state is reached after only a few thousand cycles: the extra cycles are used to increase the sensitivity of the saturation point measurement by making it more likely that a network running even slightly above capacity will saturate. Saturation is determined to have taken place if the injector queue of any node overflows its capacity of 200 flits. This criterion is justified because it can only be caused by prolonged back pressure that is very unlikely to be caused by a local hotspot created in a load significantly less than the saturation point. Each combination of network, communication pattern, and packet size was simulated with respect to a number of loads (typically 12-15) which converged about the saturation point. Thus most of the latency/load points recorded are at and beyond the knees of the latency/load graphs where maximum sensitivity is required. The maximum load that does not cause saturation is determined to be the capacity of the network. The standard deviation on the capacity measure was found to be .011 flits per node per cycle.

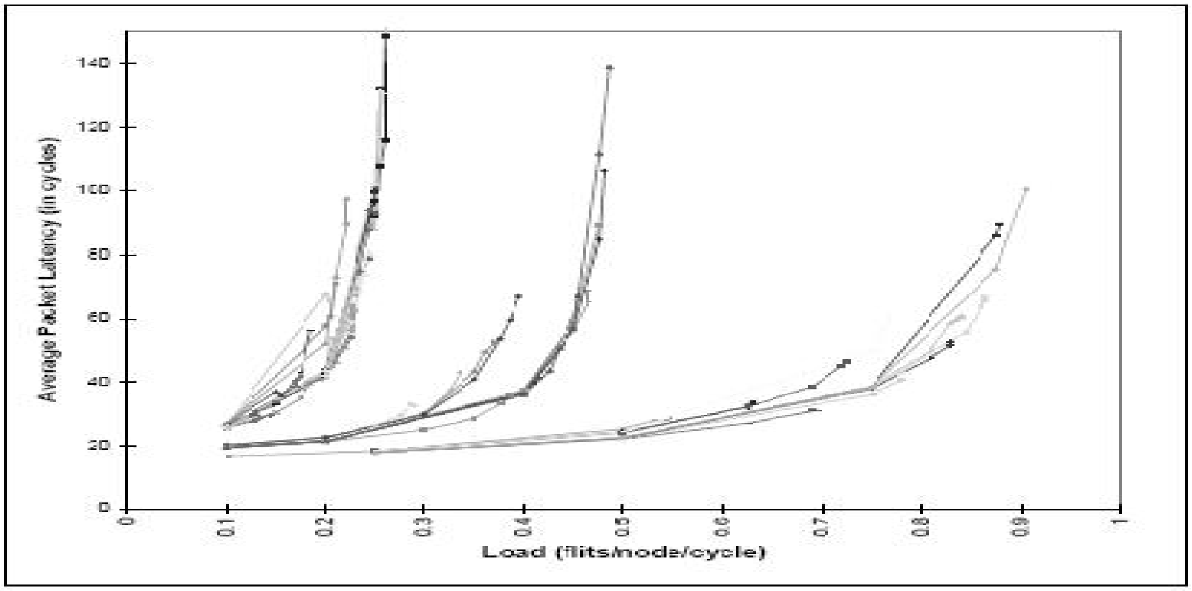


Figure 2 Shown is a graph of latency versus applied load for a number of switching designs with the random communication pattern and a packet size of 5 flits. Measurements were taken using the RTL simulator. The three 'bundles' of series correspond to the unidirectional torus wormhole, the mesh wormhole, and the bidirectional wormhole configurations respectively.

## RESULTS

The results we present in this paper include some key observations about virtual cut-through routing, performance of the designs with a cycle-driven simulator, router cell area, and router cycle time analysis. The results presented show the effect of these options on the routers' cell area and cycle timing when mapped into a current generation 0.18 micron technology, LSI Logic's G10-p Cell-Based ASIC technology.

This work adds to previous studies in that it accounts for:

- Variations in the number of lanes in the VCT configurations.
- Not only bidirectional tori, but meshes and unidirectional tori are considered.
- Deadlock and timing considerations unique to VCT.
- Static virtual channel selection versus dynamic lane selection methods.
- Both physical properties, such as critical path timing and layout area, and latency/bandwidth results from register transfer level simulations.

Some of the key results we present are:

- Lanes are as useful to VCT networks as they are to wormhole networks.
- When operating frequency is factored in, increasing the number of lanes beyond 2 per physical channel is not likely to be cost effective.
- For equal numbers of buffers and space per buffer, VCT switching is likely to have better performance than WH switching. The reason is that the space guarantee associated with VCT switching is easy to implement for small packets and has powerful consequences in load balancing among lanes and, to a lesser extent, flow control latency.
- All the designs described are evaluated with respect to area/performance for each workload and the cost-effective ones enumerated.
- While investigating the design framework, it was necessary to work out some issues that are themselves contributions. These are:
- Observations about VCT deadlock and how to prevent it.
- The application of virtual channel load balancing to unidirectional WH torus networks.

## CONCLUSION

In this work we have endeavored to exhaustively explore the design space of low-cost multicomputer networks including issues in switching, lane selection, buffer size, topology, routing algorithm, and packet size. Virtual Cut-Through is examined and found to be both more and less like Wormhole switching than previously described. More in that virtually all of the critical hardware including multi-lane channels can be identical to that of WH switches. Less in that VCT has its own deadlock issues and is open to a different lane selection paradigm. It has been stated elsewhere that VCT versus WH routing is a tradeoff between buffering and congestion. We suggest that for small packets and equal buffering, it is really mostly the difference between static and dynamic lane selection.

## REFERENCES

[1] Aoyama, K., and Chien, A. A. The Cost of adaptivity and virtual lanes in a wormhole router. *Journal of VLSI Design* (1994).

[2] Bolding, K. Non-uniformities introduced by virtual channel deadlock prevention. Tech. Rep. UW-CSE-92-07-07, Dept. of Comp. Sci. and Eng., U. of Washington, Seattle, WA 98195, 1992.

[3] Bolding, K., Fulgham, M., and Snyder, L. The case for adaptive routing. *IEEE Trans. On Computers C-46*, 12 (1997), 1281-1292.

[4] Bolotski, M, et.al. Abacus: A 1024 Processor 8ns SIMD Array.

[5] Carbonaro, J., and Verhoorn, F. Cacallino: The teraflops router and NIC. In *Proc. of Hot interconnects Symposium IV* (1996).

[6] Chien, A. A. A cost and speed model for k-ary n-cube wormhole routers. In *Proc. of Hot Interconnects '93* (1993).

[7] Dally, W. J. Performance analysis of *k*-ary *n*-cube interconnection networks. *IEEE Trans. On Computers C-39*, 6 (1990), 775-785.

[8] Dally, W. J. Virtual channel flow control. *IEEE Trans. on parallel and Distributed Systems 3*, 2 (1992), 194-205.

[9] Dally, W. J., and et al. The message processor: A multicomputer processing node with efficient mechanisms. *IEEE Micro 12*, 2 (1994), 194-205

[10] Dally, W. J., and Seitz, C. L. The torus routing chip. *Distributed Computing 1* (1986), 187- 196.

[11] Dally, W. J., and Seitz, C. L. Deadlock free routing in multiprocessor interconnection networks. *IEEE Trans. on Computers C-36*, 5 (1987).

[12] Duato, J., Yalamanchili, S., and Ni, L. *Interconnection Networks: An Engineering Approach*. IEEE Comuter Socieity Press, Los Alamitos, CA, 1997.

[13] Stunkel, C. B., and et al. The SP2 high-performance switch. IBM Systems Journal 34, 2 (1995), 185 204.

[14] Slavko Gajin et al., An accurate performance model for network-on-chip and multicomputer interconnection networks, Journal of Parallel and Distributed Computing, 2012

[15] Walid Lafi, Didier Lattard, Ahmed Jerrya, An asynchronous hierarchical router for networks-on-chip-based three-dimensional multi-processor system-on-chip, Software:: Practice and Experience, vol 42, issue 7, 2012

[16] Remi Busseuil et al., Adaptation Strategies in Multiprocessors System on Chip , IFIP Advances in Information and Communication Technology, Volume 373/2012, 2012.

[17] Simone Terenzi et al., Optimizing built-in pseudo-random self-testing for network-on-chip switches, Proceedings of the Interconnection Network Architecture: On-Chip, Multi-Chip Workshop, pp 21-24, 2012