Low-latency and Energy-efficient Monitoring Interconnect for Hierarchical-agent-monitored NoCs

Liang Guang, Pekka Rantala, Ethiopia Nigussie , Jouni Isoaho, Hannu Tenhunen
Department of Information Technology, University of Turku, Finland
{liagua, peaura, ethnig, jisoaho, hatenhu}@utu.fi

Abstract—**This paper presents quantitative analysis of monitoring interconnect architecture alternatives in hierarchical agent-based NoC platform. Hierarchical monitoring design methodology provides scalable dynamic management services with agents monitoring different levels. To enable low-latency and low-energy agent communication, we examined three interconnect alternatives: TDM-based virtual channeling, unified dedicated monitoring network, and separate dedicated monitoring networks. With Orion and Cadence simulators, we estimated the energy and latency of monitoring communications on the three architectures for an 8*8 mesh network in 65nm technology. The results suggest that separate dedicate links mostly minimize the communication delay and energy consumption (66.7% and 82.1% respectively compared to TDM-based interconnect), while incurring moderate area penalty.**

I. Introduction

The size of NoCs (Network-on-chip) is constantly increasing with technology scaling. The recently released TeraFLOPS processor [1] and TILE64 processor [2] integrate 80 and 64 cores respectively on a single chip. In academia, thousand-core processors have been projected and discussed [3]. While parallelizing applications onto many processing elements leads to high potential speedup, the system suffers from a number of challenging problems. Technology scaling introduces larger variations in the circuits, which reduce the reliability and yield of the products [4]. Exacerbated by the increasing variations, faults and errors become more distributed and unpredictable [5, 6]. Power consumption, especially the dramatic increase of leakage power in sub-100nm technology, poses even tougher power wall to the system [7]. The variations and faults only worsen the power constraints as the design margin is lowered to allow for parametric variations. To tackle with the run-time unpredictability and maximize power efficiency, online monitoring techniques need to be exploited from the architectural level.

Based on previous works addressing the issues of system monitoring, particularly those on NoC platforms [8, 9, 10], we identified several design mottos for monitoring architecture on NoCs. Firstly, monitoring services have to be provided distributedly as a requirement of scalability. Distributed monitoring reduces the interconnect latency for urgent monitoring services and prevents the appearance of communication bottleneck. However, no matter how large the systems are, centralized monitoring is still an indispensable complement to localized monitoring schemes. Theoretically, a centralized monitor is able to coordinate the functioning of all components and optimizes the overall system performance. In practice, as an example, [11] adopts a single processing unit for dynamic testing operations and a global-level scheduler. Admittedly, centralized monitors have only been applied to multi-core systems, but an analogy to the complicated nervous system of human beings can help motivate the need of centralized monitors. The human nervous system is a large-scale monitoring network with numerous distributed neurons as local monitors. These neurons are coordinated by upper-level centralized monitors such as the spinal cord and the brain, which balance and optimize the general body function. Thirdly, the energy efficiency of monitoring services should be maximized .

Observing these requirements for monitoring architecture of NoCs, we have proposed a novel design method: hierarchical agent monitoring architecture. This method adds a layer of monitoring components called "agents" on the system software and hardware. These agents are structured in four levels: from the top level to the bottom, they are the application agent, the platform agent, the cluster agents and the cell agents. Each level of agents are autonomous and adaptive in monitoring the components at their corresponding level while being supervised by higher level agents. This monitoring architecture learns from the structure of human nervous network, while adopting efficient simplification for more regular NoC structures. This paper briefly explains the concept, functional partition and mapping of agent hierarchy, which are more elaborately discussed in our previous works [12]. The focus of this paper is on architecturing the monitoring communication on the NoC platform. There are different alternatives in realizing the monitoring communication, either as virtual channels or as dedicated links. We discuss three monitoring interconnect methods both qualitatively and quantitatively. The quantitative analysis of area, latency and energy overhead was performed by using state-of-the-art simulators. From these analyses, we clearly identify the benefits and issues of each monitoring interconnect method, which are very meaningful to further design exploration.

The rest of the paper is organized as follows: Section II explains the agent hierarchy, mapping and functions of each level of agents; Section III illustrates three alternative monitoring interconnect architectures; Section IV presents our quantitative estimation of the area, latency and energy overhead of each architecture; Section V concludes the paper.

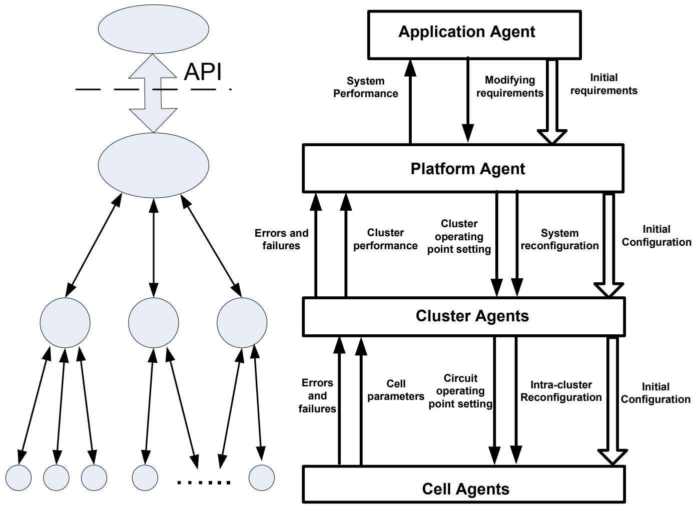

Figure 1. Hierarchical Agent Monitoring Approach

II. Hierarchical Agent Monitoring Approach

We provide a 4-level hierarchical monitoring structure for on-chip systems of any scale. Each level of monitoring is handled by agents at this level. From the top to the bottom, the system has a single application agent, a single platform agent, distributed cluster agents and cell agents (Fig. 1). The application agent is a piece of software capturing the application's functionality and run-time performance requirements and constraints. The platform agent summons enough resources for the application, parallelizes and maps instruction blocks onto acquired functional units, configures the network and monitors the system performance when the application is running. Each cluster agent monitors a group of functional units and memories with the tasks assigned by the platform agent. It tracks the performance of the components inside its own cluster at a finer granularity. The lowest-level cell agent closely monitors the circuit condition of each processing core, a switch or simply a link.

Monitoring services are provided in a hierarchical manner by the joint efforts from these agents. On Fig.1, we can identify the generic monitoring interactions among the agents. Before execution, the system, including processing elements, memories and the network, is configured with initial setting. A certain ratio of functional units are reserved as spares in case of component failures. After the application starts running, all agents become active monitors. At the circuit-level, cell agents are closely tracing their local circuit conditions, including (leakage) current, workload, and any faults or failures (for instance a link failure or a malfunctioning processing unit). These low level information is sent to the cluster agents, which figures out optimal cell settings (for instance supply voltage, bias voltage and frequency) and necessary reconfiguration within the cluster. The monitoring algorithm is only processed by the cluster agent, which takes the status of all cells into account. This manner not only reduces the implementation overhead when each cell needs to process the monitoring algorithm itself, but also achieves the overall optimal performance in the cluster. Similar monitoring interactions exist between the cluster agents and the platform agent. But instead of sending low-level circuit parameters, cluster agents send component-level resource status to the platform agent, including cluster temperature and power consumption. They also ask for more resource from spares in case of many component failures. These informations is processed by the platform agent, which may determine to adjust the working of a certain cluster, for instance scaling down the supply voltage and frequency for reducing the power or lowering the temperature. If more resources need to be provided for certain clusters, the platform may need to reconfigure the network. The platform agent is communicating with the application agent, which modifies the application requirements based on external inputs. In this case, the platform agent will change the resource utilization and reconfigure the network.

The online hierarchical monitoring approach, first of all, maximizes the monitoring efficiency in a scalable manner. Circuit-level setting is monitored by the low-level cell and cluster agents, and the localized monitoring provides faster services and reduces the communication to the platform agent. The parallelized services also minimize the total implementation overhead compared to the conventional single management unit scheme, which is not scalable for thousand-core systems. With post-silicon tuning enabled by fine-grained online monitoring services, the end product can maintain optimal performance under the influence of variations and unpredictable faults with manufacturing yield improved.

III. Monitoring Communication Interconnect

A. Agent Mapping on Regular NoC Platform

On a tile-based general-purpose NoC structure, a conventional tile comprises of a PE (processing element), a NI (network interface) and a switch. The cell agent is monitoring one of the tiles, and physically it shares the space with a processing element (Fig. 2). The cluster agents are distributed evenly among the tiles when no application pattern is assumed to specific any particular cluster topology. Depending upon the complexity of cluster monitoring algorithm, a cluster agent may physically replace a conventional PE (Fig. 2) or still shares the space with a PE. The application agent and the platform agent monitor over the whole system; without application-specific assumptions, we assume they are located together at the geographic center of the tiling area.

B. Interconnect Architecture Alternatives

Agents exchange monitoring information with their higher or lower counterparts as illustrated in Fig. 1. Compared to data communication, the monitoring information requires much lower throughput since the information volume is small ([13] reports 8% and 5% debugging monitoring traffic overhead for two streaming applications). Instead, other features are of more importance to monitoring communication: 1) reconfigurability. If the initial configuration fails (link failures or component

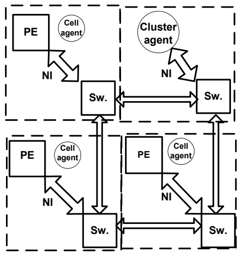

Figure 2. Layout of Cluster and Cell Agents on Tile-based NoCs (only showing a fragment of the NoC platform)

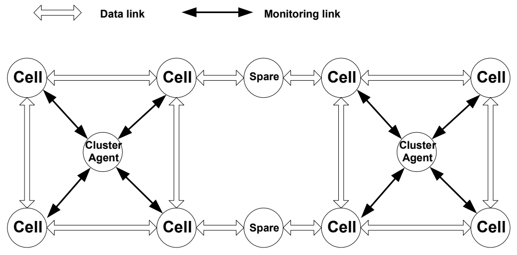

Figure 3. Non-reconfigurable Star Networks for Agent Monitoring Interconnect

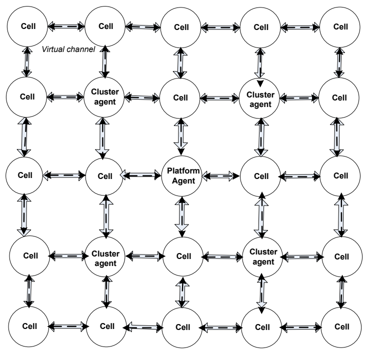

Figure 4. TDM-based virtual channel for monitoring interconnect

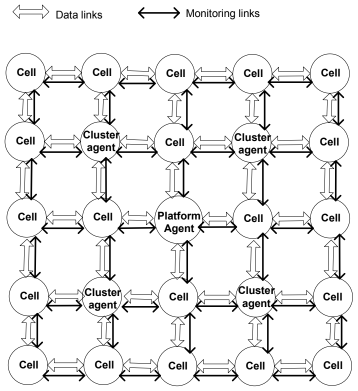

Figure 5. Unified Dedicated Network for Monitoring Interconnects

failures), the system needs be reconfigured with agent communication resumed. Some traditional interconnect methods do not support reconfigurability. For instance, if the cluster agent is connected in dedicated star network to its cell agents (Fig. 3), spares can not be dynamically configured to this cluster as new cell agents in case of component failures. 2) low latency. Even though not all monitoring services are under strict timing deadlines, the interconnect should provide the possibility of setting up low-latency connection for urgent services (for instance to deal with functional failures on processing element or links). 3) low energy. The energy overhead of monitoring communication should be minimized. The wiring area overhead has become less of a design constraint as multi-layer fabrication process provides quite abundant wiring potential for on-chip systems ([10]; TILE64 processors incorporate 5 physically separate networks, each of them being 64-bit wide). Considering these features, we identified three alternative interconnect architectures for monitoring communication:

1) TDM-based Virtual Channel: TDM (Time-Division-Multiplexing) is a conventional manner for virtual channel implementation [14, 15]. Fig. 4 illustrates the monitoring interconnect realized as TDM-based virtual channels on existing data links. This interconnect architecture incurs design complexity in virtual channel arbitration and allocation, increases the switch latency of both monitoring interconnect and data communication. The virtual channel arbitration and allocation also incur energy overhead. Wiring overhead, however, is kept to the minimum though the switch area is moderately increased.

2) Unified Dedicated Monitoring Network: A single dedicated network for the monitoring communication exploits the wiring resources and simplifies the switch arbitration (Fig. 5). Without being overlapped on the data links and switches, the monitoring communication latency and energy consumption are reduced. This dedicated monitoring network combines both the monitoring communication between the cluster and cell agents as well as that between the platform and cluster agents.

3) Separate Dedicated Monitoring Networks: We can further separate the monitoring communication between the platform agent and cluster agents on a separate dedicated network as in Fig. 6. The interconnects between cluster agents and the platform agent are fixed, which are different from those between the cell and cluster agents since cells can be reconfigured to different clusters. Building a separate network connecting the single platform agent to a small number of cluster agents reduces the communication latency and energy between these two levels of agents. It is a result of a much simpler crossbar structure and switching arbitration with the H-tree topology. The interconnects are layouted as segmented wires in parallel to the other two networks. The area penalty is small since the number of cluster agents is limited.

IV. Quantitative Comparison of Monitoring Architectures

A. Area, Latency and Energy modeling of NoC components

The NoC communication structure comprises of switches and links. We assume input-buffered switches with the structure as Fig. 7 (similar to that suggested by [16]). When a data

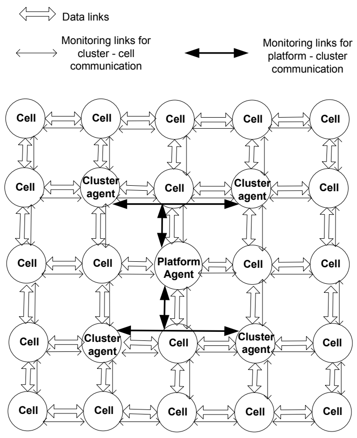
Data links
Monitoring links for cluster - cell communication
Monitoring links for platform - cluster communication
Cell
Cluster agent
Platform Agent

Figure 6. Separate Dedicated Networks for Monitoring Interconnects

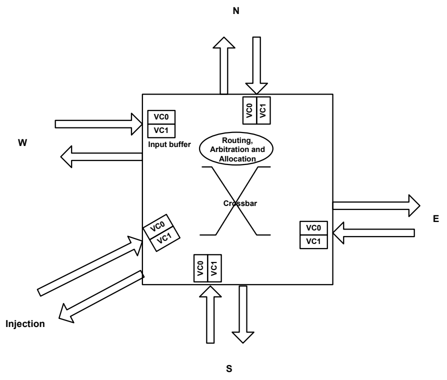

Figure 7. A 5-ported input-buffered on-chip switch model

flit [1] arrives at the switch, it will be put into the buffers [2]. The length of buffers is a design choice based on the traffic pattern, flow control mechanism, and performance/cost trade-off. After a flit has come to the head of the buffer, it will be routed and traversed through the crossbar structure to the output. There are two major crossbar structures: multiplexor-tree and matrix crossbar [17]. The virtual channel allocation, switching and routing of a data flit are all controlled by arbitration logics. The area, latency and energy overhead of a switch are contributed by the input buffers, the crossbar structure and the arbitration logic. NoC links can be modeled as segmented wires with drivers and evenly inserted repeaters. The interval length between repeaters is a design trade-off of latency and power efficiency. The smaller the interval is, the shorter the latency becomes as the result of larger driving while the power consumption including the leakage on the repeaters increases.

We estimate area and energy overhead of switches by simulating with Orion [18], a widely-used on-chip switch power simulator. The switch latency is estimated based on [16]. The wires are modeled and simulated by Cadence; with

[1]flit stands for flow control unit; a whole data packet is segmented into a number of flits each sent at once on the link

[2]there are bufferless NoC platforms as well

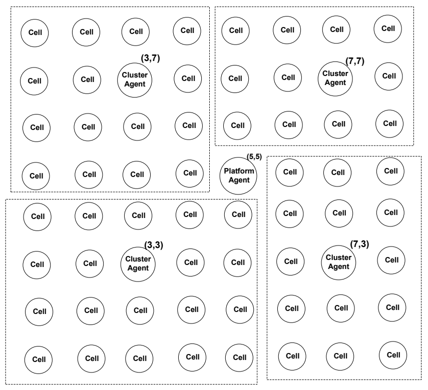

Figure 8. Locations of Platform, Cluster and Cell Agents in the Experimental Platform (with initial cluster boundary labeled)

parameter	value
length	2mm
width	210nm
spacing	210nm
repeater interval	0.25mm
repeater size	10x
driver size	12x

Table I
PARAMETERS USED IN WIRE MODELS (65NM TECHNOLOGY)

a specific latency constraint dependent on the link frequency, the repeater and driver parameters are adjusted to achieve the minimal energy.

B. Experiment Setup

We model a network similar to the TeraFLOPS processor in the same 65nm technology for realistic evaluation. The network has 8*8 processing elements mapped on a regular tile-based mesh topology. Data links are 32 bits wide and 2 mm long (TeraFLOPS uses 2mm * 1.5 mm tiles, while we simplify the tiles to be squares as 2 mm * 2 mm). The locations of the platform agent, cluster agents and cells (with cell agents) are illustrated in Fig. 8. Dedicated monitoring link is 8-bit wide and equally long. Table I gives the wiring parameters used in Cadence simulation [3]. Each router has 5 IN/OUT ports with input-buffering. For TDM channels, the buffer is 4 units (a unit is one flit wide) long since the data communication has higher workload while the unified separate network has 2-unit long buffer at each IN port. The other dedicated network for communication between cluster agents and the platform agent assumes no buffer as the traffic on this network is exclusive and infrequent. The arbitration assumes wormhole routing. The crossbar is modeled with the matrix structure. The whole NoC system is mesochronous (TeraFLOPS uses phase-tolerant mesochronous synchronization), and the frequency is set as 1GHz with the supply voltage as 1V.

The Orion simulator does not produce result for 65nm technology directly, thus we apply scaling factors (based on

[3]driver and repeater sizes are relative to the minimal inverter size.

Interconnect Architecture	Delay (cluster <-> cell agents)	Delay (platform <-> cluster agents)
TDM-based	24 cycles	24 cycles
Unified Dedicated Network	16 cycles	16 cycles
Separate Dedicated Networks	16 cycles	8 cycles

Table II
LATENCY COMPARISON OF THREE MONITORING INTERCONNECT ARCHITECTURES (NETWORK WORKING AT 1GHZ)

Interconnect Architecture	Energy (cluster <-> cell agents)	Energy (platform <-> cluster agents)
TDM-based	12.92 pJ	12.92 pJ
Unified Dedicated Network	5.40 pJ	5.40 pJ
Separate Dedicated Networks	5.40 pJ	2.31 pJ

Table III
ONE-FLIT MONITORING COMMUNICATION ENERGY OF THREE MONITORING INTERCONNECT ARCHITECTURES (NETWORK WORKING AT 1GHZ)

[19]) to the result of 70nm technology simulation using Orion. The scaling factors for energy, area, and latency are 0.86, 0.86 and 0.93 respectively. The energy of wires are simulated by Cadence. The latency in the switch buffer assumes an average 50% occupancy ratio. Virtual channel allocation, routing/ decoding and crossbar traversal each takes 1 cycle (based on [16] for pipelined routers).

C. Estimation Result

1) Latency: The latency is calculated in cycles considering the longest distances between the platform agent and a cluster agent and between a cluster agent to one of its cell agent. From Fig. 8, we see that both distances are at maximum 4 hop counts with minimal routing. The wire latency is simulated to be 198ps, and each pipeline stage latency in switches is estimated to be lower than 300ps ([16], assuming an FO4 inverter delay to be 15ps in 65nm technology). With 1GHz frequency, each link and one router pipeline stage (virtual channel allocation, routing and decoding, crossbar traversal) take 1 cycle delay. Table II summarizes the latency comparison for monitoring communication in each interconnect architecture.

2) Energy Consumption: The energy is calculated by the amount of energy consumed by an 8-bit flit (as we assume dedicated monitoring networks are 8-bit wide) traversing on the longest paths between the platform agent and a cluster agent, and between a cluster agent to one of its cell agent (4 hop counts as in Fig. 8 with no misrouting). Table III summarizes the energy consumption for monitoring communication in each interconnect architecture.

3) Area : We analyzed the total wiring and switch area for each interconnect architecture as a percentage of a TeraFLOPS chip (275mm^2) (Table IV).

4) Overhead Analysis and Trade-off: From the estimation, we can observe that TDM-based interconnect architecture incurs the most latency and energy overhead, mainly as a result of complicated switch arbitration and channel allocation process. Unified dedicated network architecture reduces the latency and energy by 33.3% and 58.2% respectively for monitoring communications. Separate dedicated network architecture reduces the latency and energy of the monitoring communication between the platform and the cluster agents further more (66.7% and 82.1% respectively). Understandably, there is area penalty involved for dedicated networks, but compared to the overall chip size, the area overhead should be affordable.

Monitoring Interconnect Architecture	Area (mm^2)	Percentage (of a chip area)
TDM-based	7.44	2.71%
Unified Dedicated Network	8.95	3.26%
Separate Dedicated Networks	9.11	3.32%

Table IV
AREA OVERHEAD OF THREE MONITORING INTERCONNECT ARCHITECTURES

V. CONCLUSION

This paper overviews the hierarchical agent-based NoC monitoring architecture and elaborately examines the latency, energy and area overhead of three alternative monitoring interconnect schemes. The TDM-based monitoring interconnection is most area efficient, but requires considerably more latency and energy overhead. Unified dedicated monitoring network, which transmits all monitoring communication network on a physically independent network, reduces the latency and energy by a great amount (33.3% and 58.2% respectively). Separated dedicated monitoring networks, which separate the monitoring communication between the platform and cluster agents onto another network with a small number of destinations, further reduces the latency and energy overhead for monitoring communication between these two levels of agents (66.7% and 82.1% respectively). Area analysis shows that dedicated networks incur an affordable amount of area penalty compared to the overall chip size. Our quantitative analysis was performed by using state-of-the-art Orion and Cadence simulators.

This work has demonstrated the effectiveness of using separate monitoring interconnect networks for hierarchical agent-based monitoring services in large-scale NoC-based embedded systems. As a prioritized future work, we are looking into specific monitoring services to be implemented on such systems.

REFERENCES

[1] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 43(1):29–41, 2008.

[2] Shane Bell, Bruce Edwards, John Amann, Rich Conlin, Kevin Joyce, Vince Leung, John MacKay, Mike Reif, Liewei Bao, John Brown, Matthew Mattina, Chyi-Chang

Miao, Carl Ramey, David Wentzlaff, Walker Anderson, Ethan Berger, Nat Fairbanks, Durlov Khan, Froilan Montenegro, Jay Stickney, and John Zook. Tile64tm processor: A 64-core soc with mesh interconnect. In *Proc. Digest of Technical Papers. IEEE International Solid-State Circuits Conference ISSCC 2008*, pages 88–598, 2008.

[3] Shekhar Borkar. Thousand core chips: a technology perspective. In *DAC '07: Proceedings of the 44th annual conference on Design automation*, pages 746–749, New York, NY, USA, 2007. ACM.

[4] M. Orshansky, R. Nassif, and D. Boning. *Design for Manufacturability and Statistical Design, A Constructive Approach*. Springer, 2008.

[5] Dongkook Park, C. Nicopoulos, Jongman Kim, N. Vijaykrishnan, and C.R. Das. Exploring fault-tolerant network-on-chip architectures. In *Proc. International Conference on Dependable Systems and Networks DSN 2006*, pages 93–104, 2006.

[6] K.A. Bowman, Xinghai Tang, J.C. Eble, and J.D. Menldl. Impact of extrinsic and intrinsic parameter fluctuations on cmos circuit performance. *IEEE Journal of Solid-State Circuits*, 35(8):1186–1193, 2000.

[7] Jan M. Rabaey. Scaling the power wall: Revisiting the low-power design rules. Keynote speech at SoC 07 Symposium, Tampere, November 2007.

[8] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen. An event-based network-on-chip monitoring service. In *Proc. of the 9th IEEE International High-Level Design Validation and Test Workshop*, pages 149–154, 2004.

[9] C. Ciordas, K. Goossens, A. Radulescu, and T. Basten. Noc monitoring: impact on the design flow. In *Proc. IEEE International Symposium on Circuits and Systems ISCAS 2006*, pages 1981–1984, 2006.

[10] D. Wentzlaff, P. Griffin, H. Hoffmann, Liewei Bao, B. Edwards, C. Ramey, M. Mattina, Chyi-Chang Miao, J.F. Brown, and A. Agarwal. On-chip interconnection architecture of the tile processor. *IEEE MICRO Magazine*, 27(5):15–31, 2007.

[11] D. Sylvester, D. Blaauw, and E. Karl. Elastic: An adaptive self-healing architecture for unpredictable silicon. *IEEE Design & Test of Computers*, 23(6):484–490, 2006.

[12] Pekka Rantala, Jouni Isoaho, and Hannu Tenhunen. Novel agent-based management for fault-tolerance in network-on-chip. In *Proc. 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools DSD 2007*, pages 551–555, 2007.

[13] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon. Transaction monitoring in networks on chip: The on-chip run-time perspective. In *Proc. International Symposium on Industrial Embedded Systems IES '06*, pages 1–10, 2006.

[14] Z. Lu and A. Jantsch. Tdm virtual-circuit configuration for network-on-chip. *IEEE Transactions on VLSI Systems*, 16(8):1021–1034, 2008.

[15] John Mark Nolen and Rabi Mahapatra. A tdm test scheduling method for network-on-chip systems. In *Proc. Sixth International Workshop on Microprocessor Test and Verification MTV '05*, pages 90–98, 2005.

[16] L.-S. Peh and W.J. Dally. A delay model and speculative architecture for pipelined routers. In *Proc. of The Seventh International Symposium on High-Performance Computer Architecture*, pages 255–266, 19–24 Jan. 2001.

[17] Hangsheng Wang. a detailed architectural-level power model for router buffers, crossbars and arbiters. Technical report, Department of Electrical Engineering, Princeton University, 2004.

[18] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Proc. 35th Annual IEEE/ACM International Symposium on (MICRO-35) Microarchitecture*, pages 294–305, 2002.

[19] W. Haensch, E. J. Nowak, R. H. Dennard, P. M. Solomon, A. Bryant, O. H. Dokumaci, A. Kumar, X. Wang, J. B. Johnson, and M. V. Fischetti. Silicon cmos devices beyond scaling. *IBM Journal of Research and Development*, 50(4/5):339–361, 2006.