Hierarchical Agent Architecture for Scalable NoC Design with Online Monitoring Services

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

***Abstract*—Hierarchical Agent Architecture is proposed to provide online monitoring services to NoC-based systems. Based on circuit conditions traced at the run-time, system settings are monitored adaptively by agents at each architectural level. This monitoring approach partitions various online diagnostic and management services onto hierarchical implementation levels so as to provide scalability and variability for large-scale NoC design. This paper explains the monitoring interaction between agent levels, and focuses on system optimization alternatives handled by different agent levels. It further quantitatively analyzes the feasibility and design alternatives in monitoring communication interconnection upon regular tile-based NoC layout. Though still under intensive research, the proposed architecture is endowed with promising potential for highly-integrated NoC design.**

I. Introduction

With continuous technology scaling, the size of NoCs (Network-on-chip) is constantly increasing. Parallelizing applications onto many processing elements leads to high potential speedup as demonstrated by the recently released TeraFLOPS processor [1] and TILE64 processor [2] which integrate 80 and 64 cores respectively on a single chip. In academia, thousand-core processors have been projected and discussed [3]. However, system designers are challenged with a number of daunting issues. Conventional concerns, such as power consumption, will continue to pose tough, if not stronger, constraints on design and implementation methods. Especially the dramatic increase of leakage power in sub-100nm technology requires urgent consideration from all architectural levels [4]. New design considerations including increasing influence from PVT (process, voltage and temperature) variations [5] and unpredictable hardware and software errors only exacerbate the design complexity. Variations and faults also worsen the power constraints as the design margin is lowered to tolerate parametric variations. To deal with these issues, the system-level design method should support online dynamic services at different implementation level, so as to achieve maximum system efficiency with run-time coarse/fine-granular tuning.

A few previous works have addressed system monitoring services on NoC platforms [6, 7, 8]. From them, several distinctive requirements for managing NoC structures in a scalable manner can be identified. Firstly, local circuits need to be provided with distributed monitoring modules. Distributed monitoring reduces the local operation delay or interconnect latency for urgent monitoring services, and it prevents the appearance of communication bottleneck. However, despite the system size, centralized monitoring is still an indispensable complement to localized monitoring schemes. Theoretically, a centralized monitor, with the knowledge of all on-chip resources, is able to coordinate and balance the functioning of all components with the aim of optimizing the overall system performance. In practice, as an example, [9] adopts a single processing unit for dynamic testing operations and a global-level scheduler. For the scenario of thousand-core NoCs with no concrete analysis available, an analogy to the overwhelmingly complex 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. For either distributed or centralized monitoring schemes, the energy efficiency of monitoring services should be maximized.

We propose a hierarchical agent architecture endowed with the required monitoring features. This architecture adds a monitoring layer of agent hierarchy onto the NoC platform. Agents are autonomous and adaptive monitors to be implemented with various approaches, and they are responsible for monitoring different architectural levels. Local agents provide fast and low-overhead monitoring services to individual functional components, and report low-level conditions and performance to higher-level agents. The latter supervise the general system performance on a coarse granularity. This architecture aims to achieve overall system performance by balancing the monitoring among all on-chip resources, while providing a wide design and synthesis space for the realization of agents at each level.

This paper examines the functional partition of agent levels and the monitoring interaction between them to perform monitoring services with an joint effort (Section II). Upon a tile-based NoC platform, we demonstrate the flexible incorporation of system optimization techniques with agent monitoring architecture (Section III). As an extra communication layer upon existing interconnect, alternatives in realizing agent communications are examined quantitatively in Section IV, which suggests an optimal design trade-off for monitoring communication interconnects. Section V concludes the paper.

II. Hierarchical Agent Monitoring Architecture

A. Agent Hierarchy

The architecture ranks the agents into four level: from the top to the bottom level, a single application agent, a single platform agent, distributed cluster agents (one per each cluster) and distributed cell agents (one per each cell) (Fig. 1). The application agent is a piece of software capturing application functionality and run-time performance requirements and constraints. The platform agent, based on the application specification and resource availability, utilizes appropriate resources, maps and schedules the instructions onto the acquired resources, configures the network, and monitors general system performance during application execution. Each cluster agent monitors a whole cluster, which is a group of processors with accompanying components (caches, scratchpad memories, switches, links, etc.). A cluster is logically divided into cells, each of which is a basic functional unit, such as a processing unit, a switch or a link. The cells are equipped with their own local monitors, the cell agents, which trace and adjust the local circuit conditions.

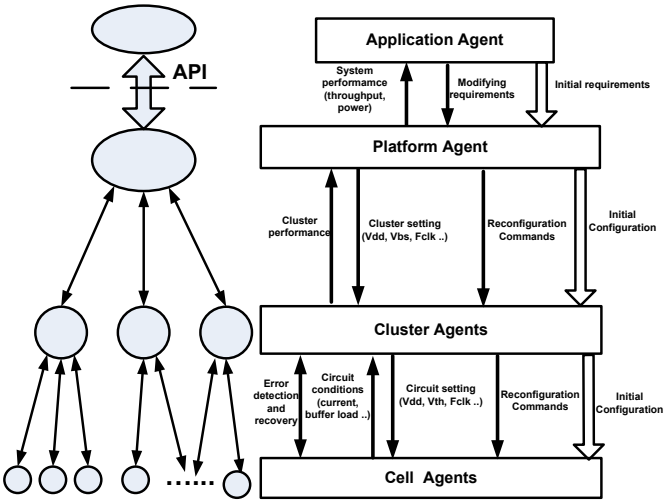


Figure 1. Hierarchical Agent Monitoring Approach

B. Hierarchical Monitoring Approach

The proposed architecture highlights hierarchical approaches to various monitoring services, for instance power-optimization and fault-tolerance, by the joint efforts from all levels of agents.

Before execution, the platform agent utilizes a number of resources and configures the network based on the initial application requirements with power and performance awareness [10]. A number of resources are reserved as spares in case of component failures. The initial configuration is enforced from the platform agent to the cluster and then cell agents.

After the application starts running, the cell agents are tracing their local circuit conditions, such as current (including leakage current for idle components), workload, and any faults or failures (for instance a link failure or a malfunctioning processing unit). Cell agents attempt to fix the errors if feasible (for example by retransmission in case of transient crosstalk-induced error [11]). The traced circuit conditions along with not-solved-yet failures are sent to cluster agents. Cluster agents attempt to adjust the cell settings based on these information. For instance they may scale the supplies of a certain cell (DVFS: dynamic voltage and frequency scaling) or the threshold voltage by using ABB (adaptive body biasing [12]). If a component has failed to work, they will acquire spare components and configure them into the cluster. Cluster agents send cluster performance to the platform agent. The information concerning cluster performance is represented at a coarser granularity than those sent between cluster and cell agents, for example, the power consumption of the cluster, or average network workload within the cluster, the error rate of the cluster. Based on these information, the platform agent may reconfigure the system, for instance assigning more spares into a failure-prone cluster, or scale down the voltage and frequency of a cluster with overwhelming power consumption. The overall system performance, for instance the throughput and the power consumption, is reported by the platform agent to the application agent, which may modify the real-time application requirements. Fig. 1 illustrates these hierarchical monitoring interactions.

The hierarchical agent-based monitoring approach is distinctive as being scalable and implementation-flexible for any-sized NoCs. The distributed cell agents, as physically adjacent to the functional units and exclusively responsible for local monitoring, can provide fast and fine-grained monitoring services to local circuits. The cluster agents are exclusively responsible for their own clusters, thus cluster-level monitoring is still low-latent and requires limited amount of processing capacity. The platform agent, though monitoring the whole system, only handles the resources at a coarse granularity. For instance, in terms of fault-tolerance, only errors which can not be fixed by low-level agents are reported to and handled by the platform agent. In this manner, no communication or processing bottleneck will appear in any large-scale platform. The supervision of higher level agents over lower-level ones ensures the optimal overall system performance. Hierarchical monitoring approach also provides a wide design and synthesis space for implementing various management algorithms and circuits. Low-level circuit optimization methods, such as power or clock-gating can be implemented as dedicated circuits triggered by cell agents. High-level component management methods, such as DVFS or ABB, can be enforced by cluster level agents. Low-level circuit optimization should be simple in terms of synthesis to offer fast operation with small overhead. High-level operations can require more processing power since they are typically much less frequent than low-level operations. As the highest-level monitor, the platform agent configures the system with optimal general settings, for instance, an appropriate network connection to reduce inter-cluster communication. Only with the concept of monitoring hierarchy can various optimization methods be implemented efficiently with different design and synthesis constraints.

III. Hierarchical Monitoring Services on NoCs

A. Agent Mapping on Regular NoC Platform

To discuss the feasible mapping of agents on NoC platform, we consider the regular tile-based mesh structure. A conventional tile comprises of a PE (processing element), a NI (network interface), a switch and the links. On such tile-based NoC platform, we naturally locate a cell agent for each tile, though distributed monitoring circuits may be located at particular places within the cell, for instance, a power-gating sleep-transistor on the link. The cell agent physically shares the space with a processing element. The cluster agent is located at fixed locations at design time, and cells are configured into the clusters dynamically at the run-time. Depending upon the complexity of cluster monitoring algorithm and maximum number of cells to be monitored, a cluster agent may physically replace a conventional PE or still shares the space with a PE. The application agent and the platform agent monitor over the whole system; without application-specific knowledge, we assume they are located together at the geographic center of the tiling area. Fig. 2 illustrates the feasible mapping of agents on the regular NoC structure.

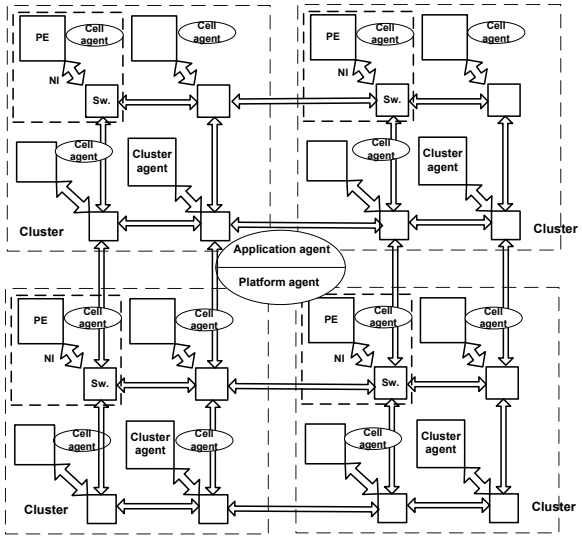


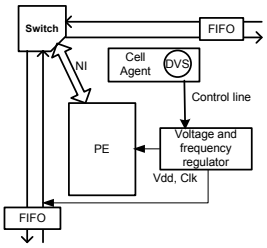
Figure 2. Illustration of Agent Mapping on NoCs

To offer scalability for thousand-core systems, clusters can be divided into hierarchical subclusters and similar monitoring functional partition will be applied. It conceptually originates from the manner a biosystem or human society organizes its overwhelming amount of resources.

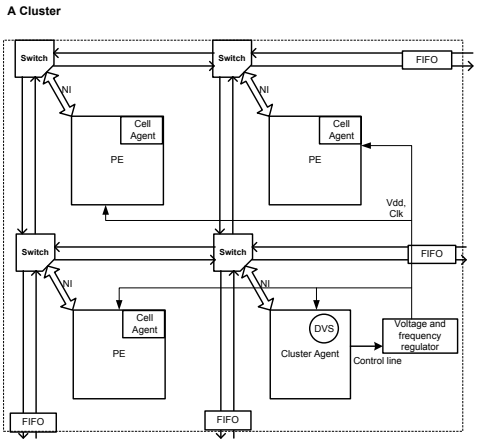
B. Low-power Optimization with Agents

In the hierarchical agent architecture, various monitoring services can be incorporated at different implementation level considering the specific trade-off on the actual platform. Here we explain the design consideration with dynamic power optimization as an example of various feasible services.

One of the major dynamic power saving techniques is DVFS, which is traditionally provided on a chip-wide domain



(a) Cell-level DVFS (showing one cell)



(b) Cluster-level DVFS (showing one cluster)

Figure 3. Power Optimization Services by Different Agent Levels

[13]. But chip-level single power domain is not able to utilize the local traffic variation in exploiting the supply scaling potential, thus per-core based DVFS is proposed [14]. In the cell-divided NoC platform, a cell can be conveniently set with a supply regulator with the cell agent in charge of the voltage and frequency adjustment (Fig. 3(a)). The overhead for per-cell based DVFS is significant. [15] reports $0.14mm^2$ area overhead and 83.2% peak efficiency of a DC-DC converter in $90nm$ technology. Each time the voltage is converted, extra energy will be consumed for the power regulation.

To alleviate the per-core-based DVFS overhead, the concept of voltage islands [16, 17] has been proposed. Up-to-date, voltage islands are statically determined at design time. To incorporate multiple voltage islands on the NoC platform, each cluster agent determines the voltage and frequency for its own cluster (Fig. 3(b)). The area and energy overhead is reduced proportional to the number of cells in a cluster. Per-cluster-based power optimization, however, does not support the reconfiguration of cells into different clusters at the run-time, though assigning spares into clusters initially still provides cell replacement possibilities against component failures.

The granularity of monitoring services is a design choice dependent on the size of the actual platform, the workload and constraints of the application. In terms of power optimization, per-cluster-based monitoring with lower implementation overhead seems to be more feasible in the long term with smaller-sized processing cores. In general, any monitoring service can

be configured at the design time or execution time (with the support of reconfigurable platform) to be handled by different level of agents, correspondingly in various granularities.

IV. Design Trade-offs for Agent Communication

A. Monitoring Communication Interconnect Alternatives

Agents exchange monitoring information with their higher or lower counterparts as illustrated in Fig. 1. The monitoring communication needs to be reconfigurable so new cells can be incorporated to certain clusters at the run-time. Some conventional interconnection does not support reconfiguration (for instance, the star-like network as in Fig. 4). Instead, we consider three interconnect alternatives which all support run-time reconfiguration but have different area, energy and latency overheads. Throughput is not a prioritized design constraint, since the monitoring communication is low in data volume ([18] reports 8% and 5% debugging monitoring traffic overhead for two streaming applications).

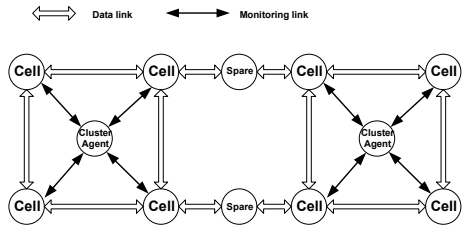


Figure 4. Non-reconfigurable Star Networks for Agent Monitoring Interconnect

The first alternative is to realize monitoring communication as TDM (Time-Division-Multiplexing)-based virtual channel upon existing links. This option 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.

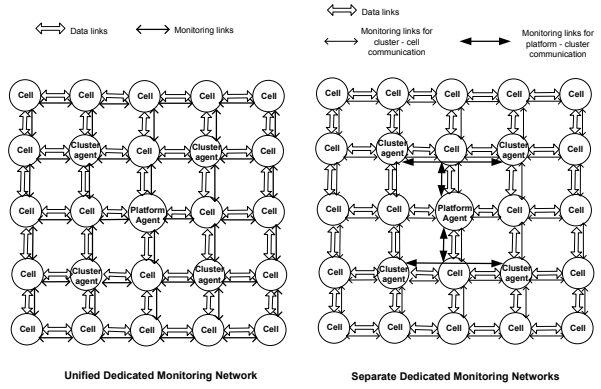


Figure 5. Alternative Dedicated Monitoring Interconnect Architectures

The second alternative is to adopt a "unified dedicated monitoring network" for monitoring communication (Fig. 5 on the left side). It is called "unified" as monitoring communication between both cluster-cell agents and platform-cluster agents is transmitted on the same dedicated network. This option utilizes more wiring resources but simplifies the switch arbitration between data and monitoring communication, thus reducing the communication energy and latency.

The third alternative is to adopt "separate dedicated monitoring networks" for monitoring communication (Fig. 5 on the right side). Compared to the unified monitoring network, this option adds another network connecting the single platform agent to a small number of cluster agents. As a result, the communication between platform and the cluster agents is simplified with very limited wiring overhead.

B. Quantitative Analysis of Monitoring Interconnects

To quantitatively compare the implementation overhead of three monitoring interconnect architecture, we model a network similar to the TeraFLOPS processor in the same 65nm technology. The network has 8*8 processing elements mapped on a regular tile-based mesh topology. We assume input-buffered pipelined switches with the structure suggested by [19] with matrix crossbar [20]. For TDM channels, each input buffer is 4-flit long while the unified separate network has 2-flit-long input buffer considering the higher traffic load of data communication. The other dedicated network for communication between cluster agents and the platform agent assumes no buffer since the traffic on this network is exclusive and infrequent. The arbitration assumes wormhole routing. NoC links are modeled as segmented wires with drivers and evenly inserted repeaters[1]. Data links are 32 bits wide and 2 mm long [2], and dedicated monitoring link is 8-bit wide and equally long. The locations of the platform agent, cluster agents and cells (with cell agents) are illustrated in Fig. 6. The whole NoC system is assumed to be mesochronous with network frequency as 1GHz and the supply voltage as 1V.

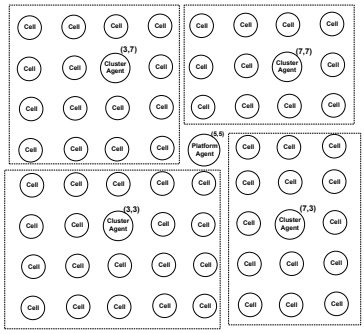


Figure 6. Locations of Platform, Cluster and Cell Agents in the Experimental Platform (with initial cluster boundary labeled)

We estimate the area and energy overhead of switches by simulating with Orion [21], a widely-used on-chip switch

[1]wire width: 210nm; spacing: 210nm; repeater interval: 0.25mm; repeater size: 10x minimal inverter size; driver size: 12x.

[2]TeraFLOPS uses 2mm * 1.5 mm tiles, while we simplify the tiles to be 2 mm * 2 mm squares

power simulator. The switch latency is estimated based on [19]. The wires are modeled and simulated by Cadence. The Orion simulator does not produce result for 65nm technology directly, thus we apply scaling factors (based on [22]) 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.

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. 6, 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 ([19], 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 I summarizes the latency comparison for monitoring communication in each interconnect architecture.

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

2) Energy Consumption: The energy is calculated by the amount of energy consumed by a 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. 6 with no misrouting). Table II summarizes the energy consumption for monitoring communication in each interconnect architecture.

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

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

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 III
AREA OVERHEAD OF THREE MONITORING INTERCONNECT ARCHITECTURES

C. Optimal Design Trade-off for Future NoCs

The estimated figures show that separate dedicated monitoring networks are the most energy-efficient and low-latency interconnection for monitoring communication. Compared to TDM-based interconnection, it reduces the latency by 66.7% and energy consumption 82.1% for the communication between the platform and cluster agents, while achieving the same latency and energy efficiency as unified dedicated network for the communication between the cluster and cell agents. However there is area penalty involved: the area overhead is increased from 2.71% to 3.32%. Nonetheless 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 ([8]; TILE64 processors incorporate 5 physically separate networks, each of them being 64-bit wide). With transistor feature size and wire dimension continue to decrease in the foreseeable future, the separate monitoring networks will provide the most optimal trade-off exploiting the on-chip wiring resources while minimizing the more critical power consumption and global interconnect latency.

V. CONCLUSIONS

Hierarchical agent monitoring architecture provides great scalability and design flexibility for future large-scale NoC systems. With an extra monitoring layer comprised of four levels of agents, the system is potentially able to achieve maximized efficiency with online monitoring services. This paper elaborately explains the hierarchical monitoring approaches enabled by the interactions of all levels of agents, and examines the design alternatives for low-power optimization of different granularities as an example of flexible functional partitions among agent levels. Quantitative analysis for agent interconnection alternatives suggests reasonable trade-offs between area, energy and latency overhead, and motivates separate dedicated monitoring networks for inter-agent communication. This work demonstrates the potential and feasibility of multi-level online monitoring layer upon the overwhelming amount of on-chip resources, which provides a great diversity of design options in a scalable manner.

At present, specific monitoring services on regular NoC platform with the proposed architecture is under intensive study and analysis.

REFERENCES

[1] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erra-

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

[5] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer. High-performance cmos variability in the 65-nm regime and beyond. *IBM Journal of Research and Development*, 50(4/5):433–449, 2006.

[6] 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.

[7] 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.

[8] 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*, 27(5):15–31, 2007.

[9] 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.

[10] Jingcao Hu and R. Marculescu. Energy and performance-aware mapping for regular noc architectures. *IEEE Transactions on COMPUTER-AIDED DESIGN of Integrated Circuits and Systems*, 24(4):551–562, 2005.

[11] Teijo Lehtonen, Pasi Liljeberg, and Juha Plosila. Online reconfigurable self-timed links for fault tolerant noc. *VLSI Design*, 2007:13, 2007.

[12] S.M. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proc. IEEE/ACM International Conference on Computer Aided Design ICCAD 2002*, pages 721–725, 2002.

[13] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proc. of 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39)*, pages 347–358, 2006.

[14] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *International symposium on high-performance computer architecture*, Feb. 2008.

[15] P. Hazucha, G. Schrom, Jaehong Hahn, B.A. Bloechel, P. Hack, G.E. Dermer, S. Narendra, D. Gardner, T. Karnik, V. De, and S. Borkar. A 233-mhz 80%-87% efficient four-phase dc-dc converter utilizing air-core inductors on package. *IEEE Journal of Solid-State Circuits*, 40(4):838–845, 2005.

[16] Lap-Fai Leung and Chi-Ying Tsui. Energy-aware synthesis of networks-on-chip implemented with voltage islands. In *Proc. 44th ACM/IEEE Design Automation Conference DAC '07*, pages 128–131, 2007.

[17] D.E. Lackey, P.S. Zuchowski, T.R. Bednar, D.W. Stout, S.W. Gould, and J.M. Cohn. Managing power and performance for system-on-chip designs using voltage islands. In *Proc. IEEE/ACM International Conference on Computer Aided Design ICCAD 2002*, pages 195–202, 2002.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.