On-Chip Sensor Network for Efficient Management of Power Gating-Induced Power/Ground Noise in Multiprocessor System on Chip

Weichen Liu, *Member*, *IEEE*, Yu Wang, *Member*, *IEEE*, Xuan Wang, *Student Member*, *IEEE*, Jiang Xu, *Member*, *IEEE*, and Huazhong Yang, *Senior Member*, *IEEE*

Abstract—Reducing feature sizes and power supply voltage allows integrating more processing units (PUs) on multiprocessor system on chip (MPSoC) to satisfy the increasing demands of applications. However, it also makes MPSoC more susceptible to various reliability threats, such as high temperature and power/ground (P/G) noise. As the scale and complexity of MPSoC continuously increase, monitoring and mitigating reliability threats at runtime could offer better performance, scalability, and flexibility for MPSoC designs. In this paper, we propose a systematic approach, on-chip sensor network (SENoC), to collaboratively predict, detect, report, and alleviate runtime threats in MPSoC. SENoC not only detects reliability threats and shares related information among PUs, but also plans and coordinates the reactions of related PUs in MPSoC. SENoC is used to alleviate the impacts of simultaneous switching noise in MPSoC's P/G network during power gating. Based on the detailed noise behaviors under different scenarios derived by our circuit-level MPSoC P/G noise simulation and analysis platform, simulation results show that SENoC helps to achieve on average 26.2 percent performance improvement compared with the traditional stop-go method with 1.4 percent area overhead in an 8*8-core MPSoC in 45 nm. An architecture-level cycle-accurate simulator based on SystemC is implemented to study the performance of the proposed SENoC. By applying sophisticated scheduling techniques to optimize the total system performance, a higher performance improvement of 43.5 percent is achieved for a set of real-life applications.

Index Terms—Sensor network, reliability, dynamic control, low-power, noise, power grid, system on chip

1 Introduction

Multiprocessor system on chip (MPSoC) is becoming a favorite choice to satisfy the ever growing performance demanded by applications [1], [2], [3]. On one hand, shrinking feature size allows for more and better functions on MPSoC. On the other hand, it also makes MPSoC more susceptible to various reliability threats, such as high temperature and power/ground (P/G) noise. Improving reliability has become an important aspect of MPSoC design.

As the scale and complexity of MPSoC continuously increase, a systematic approach that not only detects reliability threats but also mitigates such threats accordingly at runtime could potentially offer better performance, scalability, and flexibility for MPSoC designs. In this paper, we propose a systematic approach, on-chip sensor network (SENoC), to collaboratively predict, detect, report, and alleviate runtime threats in MPSoC. SENoC not only detects reliability threats and shares related information among processing units (PUs), but also plans and coordinates the reactions of related PUs in MPSoC. SENoC is integrated with network-on-chip (NoC) to ensure that critical information and decision is delivered in a timely fashion.

To highlight the details of our idea, SENoC is used to alleviate the impacts of simultaneous switching noise in MPSoC's P/G network during power gating. Tight low power requirements force MPSoC to aggressively adopt low power techniques. While power gating can dramatically reduce leakage power in MPSoC, it exacerbates simultaneous switching noise on the power delivery network, and can result in performance degradation and even functional errors [4], [5], [6]. As that will be studied in Section 4, Fig. 5 shows the impact of such noise to the surrounding PUs of the chip. This motivates us to design the SENoC to effectively solve the simultaneous switching noise in MPSoC's P/G network. Based on the detailed noise behaviors under different scenarios derived by our circuit-level MPSoC P/G noise simulation and analysis platform, simulation results show that SENoC helps to achieve on average 26.2 percent performance improvement compared with the traditional stop-go method with 1.4 percent area overhead in an 8*8-core MPSoC in 45 nm. Based on our conference publication [7], we further propose sophisticated scheduling techniques to offset the overheads induced by power gating related operations, and optimize the total system performance. Efficient working strategies and communication protocols in SENoC are also presented. We implement a SystemC-based

- *W. Liu is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong. E-mail: weichen@ust.hk.*
- *Y. Wang and H. Yang are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. E-mail: {yu-wang, yanghz}@tsinghua.edu.cn.*
- *X. Wang and J. Xu are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {eexwang, eexu}@ust.hk.*

Manuscript received 1 Nov. 2011; revised 9 June 2012; accepted 11 June 2012; published online 22 June 2012.
Recommended for acceptance by D. Kaeli.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-11-0807.
Digital Object Identifier no. 10.1109/TPDS.2012.193.

architecture-level cycle-accurate simulator for the SENoC, and observe a higher performance improvement of 43.5 percent on average for a set of real-life DSP and multimedia applications with the enhanced scheduling techniques applied.

The rest of the paper is organized as follows: Related works are presented in Section 2. Section 3 gives an overview of SENoC. Serving as the basis of our approach, the circuit-level modeling for power gating induced P/G noise is presented in Section 4. In Section 5, we focus on the behavior models of the key components in SENoC and formalize their mechanisms. Section 6 shows the implementation and simulation results. Section 7 concludes the paper.

2 Related Work

In this section, we summarize the existing techniques that work on optimizing the power gating process and mitigating the induced P/G noise. Circuit/block level techniques can be applied to improve the efficiency of the power gating technology, and system-level approaches can be used for overhead control and tradeoff analysis. Sensor-based system designs are also studied.

Many circuit level techniques are proposed for the P/G noise mitigation with power gating techniques. Optimized sleep transistor designs for power gating are critical to a successful low-power design. Various techniques were presented for power-on current rush control through the sleep transistor implementation that reduces power supply voltage fluctuation [8], [9], [10], [11]. The optimization of the sleep transistor distribution and the sizing selection were demonstrated to be effective to trade off among the worst case noise, area overhead, and leakage power consumption by assigning large sleep transistor to control the function units with high current density [8]. Gupta et al. analyzed the voltage variations in chip multiprocessors using a distributed power-delivery network [12]. Reddi et al. used recurring program and microarchitectural event activity to predict the voltage droop [13]. Our proposed model is compatible with these models. It is designed in fine-grained granularity for the purpose of improving the simulation accuracy.

The scheme of multiple power gating modes tried to control the power-on current rush by dividing the total voltage transition into multiple steps with smaller transition, and as a result, the worst case rush current among all these small steps was reduced [9], [10], [11]. A low-power switched decoupling capacitor circuit was proposed to suppress on-chip resonant supply noise [14]. It enhanced the efficiency of the on-chip decoupling capacitor by replacing the traditional passive decoupling capacitors with some new controllable active capacitance structures to achieve the equivalent capacitance with less area overhead and small leakage power consumption. Powell et al. proposed a microarchitectural technique by controlling instruction issue to guarantee the worst case magnitude of resonant P/G noise with low energy and performance loss [15], [16]. It alleviates the switching activity induced resonant P/G noise by controlling instruction access to reduce simultaneous switching activities in the pipeline. Jiang et al. proposed a scheduling technique to address system-level power gating with several gated blocks and optimize the wake up order of these blocks in terms of noise [17]. It tries to alleviate the rush current during the power gating of heterogeneous blocks by optimizing their wakeup schedule, and reduces the power gating induced P/G noise with increased wake up time. Our work schedules application's execution at the system level to reduce the performance loss due to power gating without the consideration of reducing the P/G noise magnitude.

A power-gating driven floorplanner (PGFP) was developed in [18] to assist the design of power gated chips. Mohamood et al. [4] proposed a design methodology for power integrity aware floorplanning using microarchitectural feedback to guide the module placement. Healy et al. [5] presented an improved design methodology to combat the ever-aggravating high-frequency power supply noise (di/dt) in modern microprocessors. The ideas in [4], [5] are to build up dynamic controlling mechanisms at the microarchitecture level by dynamically monitoring the access patterns of microarchitecture modules and prevent the troublesome simultaneous switching activities among all the modules. These works mainly focused on block-level design techniques, while in this paper, we investigate processor-level power gating protection strategies based on our detailed P/G noise analysis platform for MPSoC, and study the relationship between the performance degradation and the noise protection during powering on/off PUs.

Tiny on-chip sensors can measure various runtime parameters, such as noise, error, temperature, switching activity, clock duty-cycle, and technology parameters. Ernst et al. proposed Razor, a circuit-level design for monitoring and correcting timing errors in low power systems [19], [20]. Razor motivates many sensor designs for chip reliability [21], [22]. Petrescu et al. proposed a signal integrity architecture to monitor various on-chip physical parameters, especially voltage and temperature [23]. The proposed voltage monitor was well developed with a high time resolution and accuracy, which provides opportunities to capture the profiling of the medium frequency and high frequency P/G noise during runtime. The voltage sensor is implemented with a high bandwidth 7-bit DAC in the 90 nm technology. It can achieve a DC-resolution of 10 mV and measure 100 ps wide spikes. Poirier et al. and McGowen et al. described the control system on a 90-nm Itanium processor which utilizes on-chip sensors to measure power and temperature and modulates voltage and frequency to optimize performance [24], [25]. It showed the promising potential for the sensor-based feedback control system to be implemented in the real commercial products to further improve their runtime performance. Sohn et al. proposed a sensor-based solution for SRAM to overcome the uncertainty and fluctuation of device parameters [26], [27]. These works show that the information gathered by on-chip sensors can be used to effectively improve the reliability and performance of different functional units, and it will come to practice as a powerful strategy for the next generation commercial products.

Architecture level frameworks are proposed to collect useful information and adjust the function units at runtime. Machine Check Architecture (MCA) is an error protection mechanism mainly designed for general purpose processors, and the mechanism can be used to detect, locate, and log system faults like parity errors in cache, ECC errors in DRAM and system bus errors [28]. MCA is a more general architecture that can be widely used in many error protection scenarios. Compared to that, SENoC is designed

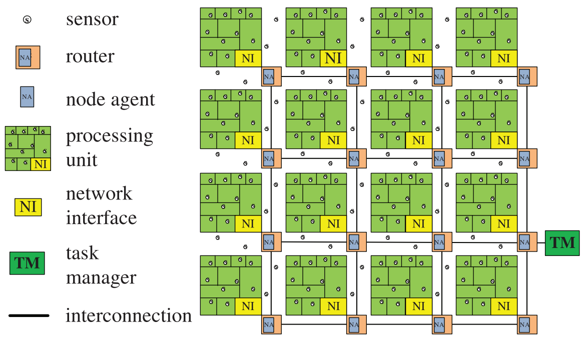

Fig. 1. SENoC on a 4*4-core MPSoC.

to target NoC-based MPSoC architectures. Currently, the sensors can only be used for voltage monitoring, and the mechanism is designed for solving the P/G noise problem. Chan et al. proposed to use system management bus to communicate between IP cores and the thermal-aware power management IP for system-level power management [29]. Yin et al. proposed a hierarchical architecture to collect runtime parameters using network on chip [30]. Several architectures for the control system and some small techniques were proposed to trade off among the energy power consumption overhead, area overhead, and communication latency of the control system. Ciordas et al. proposed a monitoring service framework to support the runtime observability of NoC behaviors and application debugging [31], [32]. It not only evaluated the tradeoff between the latency and the hardware overhead, but also took the design flow impact into consideration compared with the traditional NoC design. These frameworks for sensor-based control systems try to achieve efficient and real-time responses with small system, low hardware overhead and good scalability for highly integrated NoC design.

3 An Overview of SENoC

The SENoC is composed of on-chip sensors, node agents, and a task manager (TM). It is integrated with network-on-chip and uses NoC as the communication medium. Besides detecting runtime threats and sharing related information among PUs, the SENoC also decides and coordinates the reactions of all the PUs in a MPSoC. Fig. 1 illustrates the SENoC for a 4*4-core MPSoC with a mesh-based NoC.

Tiny sensors are embedded inside and between PUs to measure various parameters, such as voltage and temperature. Under normal operating conditions, the parameters are within a safe range, and sensors only report them upon requests from the node agents. However, sensors will immediately report any parameter beyond the safe range to node agents. They are usually placed to the positions in each PU which are close to the functional units which have high power consumption or are sensitive to temperature or voltage conditions.

Node agents are integrated with NoC routers and connected to sensors through network interfaces. Each node agent processes the warnings from a group of sensors inside a PU. It will preprocess and filter warning as well as compress warnings from multiple sensors. Based on the warning severity, a node agent can send alert packets to nearby node agents or all the node agents in addition to TM.

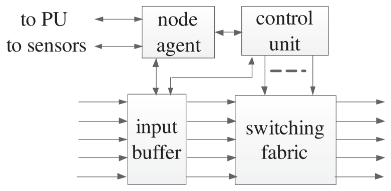

Fig. 2. An NoC router with an integrated node agent.

Node agents not only take commands for TM, but also collaborate with each other to quickly respond to a threat.

The SENoC uses NoC as the communication medium. We integrate the node agent into an input-buffered pipelined NoC router (Fig. 2). The integration allows node agents to send and receive packets in NoC in a timely fashion, which is critical to the SENoC. NoC routers use virtual channels to guarantee the delivery time of high-priority control packets. In addition to unicast, multicast is used to send packets simultaneously to multiple node agents. The SENoC uses four packet types—command packet, alert packet, report packet, and postalert packet. The TM processes packets and creates a system-wide plan to minimize threat impacts. It is connected to the NoC and sends command packets to node agents to execute the plan. When receiving a command packet, node agents will inform the PU to take specific actions, such as entering sleep mode through clock gating. Off-chip memories are connected to the routers in an efficient way to support the backup and resumption of runtime information for the PUs. In this way, each PU selects the nearest memory as its destination, and the number of memory blocks used is minimized.

4 The Physical Model of Power Gating-Induced P/G Noisy

In this paper, SENoC is used to alleviate the impacts of simultaneous switching noise in MPSoC's P/G network during power gating. In order to analyze the performance of SENoC, it is necessary to understand the impacts of power gating induced P/G noise among PUs in MPSoC. We build a P/G noise simulation and analysis platform and systematically explore MPSoC P/G noise behaviors under different power gating scenarios [33]. The noise behaviors serve as the basis for the P/G noise aware task management methodology based on SENoC.

The simultaneous switching noise induced by turning on/off PUs at different locations in MPSoC is evaluated based on circuit-level simulations using HSPICE. We assume homogeneous MPSoC systems in the simulation model. Based on [6], [12], we assume all the PUs charge the same amount of load capacitance during the power gating process. Each PU is composed of multiple identical subblocks which can be controlled separately. A subblock consists of the mesh-based power grid model, the load capacitance of the function logics, the decoupling capacitor, and an inverter. The load capacitance of each subblock is modeled by the average over the entire PU following the uniform distribution. The inverter is placed between each pair of power grid node and its adjacent ground grid node, and works as a sleep transistor to represent PU switching activity. Different combinations of the inverter control can be used to mimic different power gating scenarios. A subblock of the P/G network circuit model is shown in

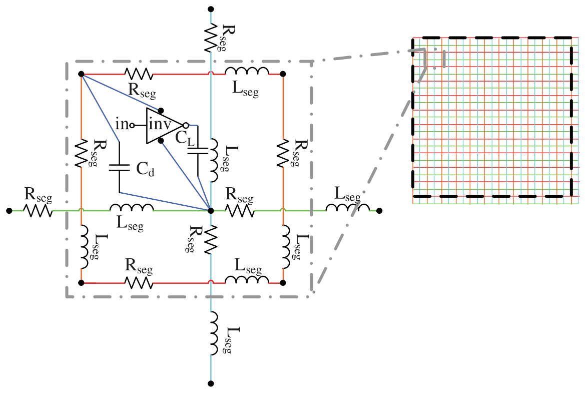

Fig. 3. In order to facilitate the P/G network analysis, each wire segment is modeled as a chain of L-type RLC equivalent circuits. An inverter with a capacitance load is used to imitate logics. A decap is connected to the intersection points on the vdd/vss grids.

Fig. 3. We use the 45 nm bulk CMOS model [34] for transistors ($V_{dd} = 0.8$ V) and the standard cell library is from the Nangate Open Cell Library [35] for the whole design. The parameters are set as follows: $R_{seg} = 0.628\ \Omega, L_{seg} = 0.005$ nH, $C_L = 0.23$ pF. For the power grid model, they are derived from the typical design of global interconnects implemented between the metal layers M4 to M7 [36], [37]. R_{seg} and L_{seg} are calculated based on the estimation on the length and area of the cross section of the segment. While for the lumped capacitance of the PU per 100 μm^2, C_L is estimated based on the capacitance/area ratio of the inverter gate in the 45 nm Nangate library, where it is assumed half of the gate capacitors are charged during power gating [35]. Jiang et al. used a similar power gating model for P/G noise analysis in [6]. Our model is compatible with related models. It is designed in fine-grained granularity for the purpose of improving the simulation accuracy.

The MPSoC is modeled with a set of PUs, $P = \{p_i | i \in [1, N]\}$. The PU states considering power gating induced P/G noise are defined in Table 1. A PU in *ToOn/ToOff* is defined as an *attacker*. A PU who carries a running task is defined as an *active* PU. An active PU within the impact range of an attacker is defined as a *victim*. (Please note that some power-on PUs could be in *Idle* or *Free*, and they are not victims in our definition.) For a PU p, $IR(p)$ is defined as the set of PUs within the impact range of an attacker p, while $PV(p)$ is defined as the set of p's potential victims, i.e., $PV(p) = \{q \mid q \in IR(p), q \text{ is active}\}$, and we have $PV(p) \subset IR(p)$.

If we assign a task i to a power gated PU p, the powering-on/off noise when the task begins/finishes will attack the PUs in $IR(p)$ which is provided through our P/G model. The noise protection method will migrate the data to the off-chip memory, clock gate the victim PUs before powering on/off the attacker, and later wake them up when the attacker is fully turned on/off. Fig. 4 shows the timing of a power-on event, and the timing of a power off event is similar. $T_{ClkToOff}$ and $T_{ClkToOn}$ are the times needed to clock gate a PU and to wake it up from the clock gated state, respectively. T_{ToOn} and T_{ToOff} are the settle times for a PU to power on and power off. In order to ensure the reliability of MPSoC, here $T_{ToOn} \geq \max_{q \in PV(p)}\{T_{settle}(p), T_{safe}(q)\}$, $T_{ToOff} \geq \max_{q \in PV(p)}\{T_{settle}(p), T_{safe}(q)\}$, where $T_{settle}(p)$ is the period that p powers on, and $T_{safe}(q)$ is the period that

TABLE 1
PU States in Power Gating

PU State	Description
Off	Power gated
ToOn	The off to on transition
ToOff	The on to off transition
Idle	Clock gated (power is on)
ClkToOff	Clock gating transition
ClkToOn	Clock wake-up transition
Free	Both power and clock are on, but there is no task running on the PU
Active	A task is running on the PU

victim q returns to the normal voltage level. T_{On} and T_{Off} are the noise protection penalties for a victim when an attacker powers on and off, respectively, where

$$T_{On} = T_{ClkToOff} + T_{ToOn} + T_{ClkToOn},$$
$$T_{Off} = T_{ClkToOff} + T_{ToOff} + T_{ClkToOn}.$$

Assume that the number of victims of attacker p at the time instant t is $N_{PV}(p,t)$. We define $P_{On}(p,t)$ and $P_{Off}(p,t)$ as the total performance penalties to power on and power off p, respectively, where $P_{On}(p,t) = T_{On} \times N_{PV}(p,t)$, $P_{Off}(p,t) = T_{Off} \times N_{PV}(p,t)$. Initially, these timing parameters, such as $T_{ClkToOff}$, T_{ToOn}, $T_{ClkToOn}$, and T_{ToOff}, are set to the worst case values to ensure the reliability. However, with the help of SENoC, these parameters can be dynamically determined using on-chip sensors and reduced depending on the scenarios.

In the traditional stop-go strategy, all the active PUs are protected against P/G noise during powering on/off a PU. The algorithm is simple and safe, but is too conservative according to our P/G noise model. Based on the P/G noise simulation and analysis platform, we simulate MPSoCs with different scales and conditions. Fig. 5 shows the peak P/G noise levels of PUs induced by attackers located at different locations on a 4*4-core MPSoC. In order to obtain the worst case scenarios, the PU conducting power gating switches all the subblocks simultaneously, while the other PUs are set to the power off state and do not help suppress the noise. It is observed that the peak noise induced by power gating is around 160 to 250 mV for the PUs at different locations. PU1 is at the corner of the chip, which makes it suffer from the boundary effect and have higher peak noise of around 250 mV. While PU6 is near the center of the chip and has lower peak noise of around 160 mV. Kim et al. reported similar observations in [38] that the power gating induced P/G noise for a small circuit of two linear-feedback shift registers and a 32-bit carry look-ahead adder fabricated with 130 nm technology is already about 9 percent of the supply voltage and becomes a reliability threat in low power circuit design. Different impact ranges can be observed in Fig. 5: when PU1

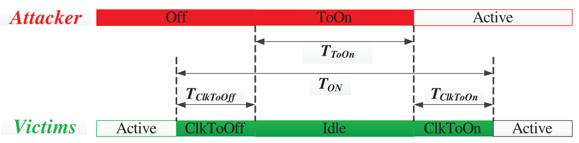

Fig. 4. Timing of a power-on event.

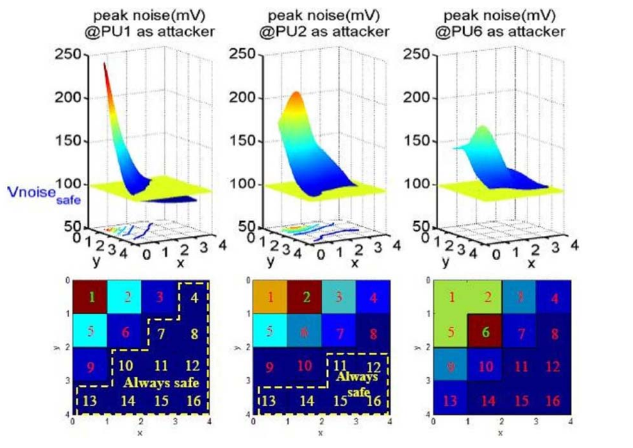

Fig. 5. Noise levels and impact ranges of power gating induced P/G noises in a 4*4-core MPSoC.

is the attacker, at most five PUs need protection; when PU2 is the attacker, at most nine PUs need protection; when PU6 is the attacker, all the other active PUs need protection. SENoC can help with the voltage level collection, noise information transfer, analysis, and planning. The detailed procedure will be discussed in Section 5.

5 SENoC for MPSoC P/G Noise Alleviation

In this section, we will present the mechanisms used by SENoC to alleviate the impacts of P/G noise during power gating. We formalize our approach by optimizing the MPSoC performance under the task constraints and PU operation constraints under P/G noise attacks.

5.1 SENoC Components

The task manager is a global controller for task management and chip maintenance. It is in charge of the management of the entire MPSoC, making system-wide decisions of task distribution and PU coordination. The TM is used to assign new tasks to PUs, and maintain the status of all the PUs. It has a scheduling control logic, which is used for online task scheduling implementation and performance control. The detailed working strategy of the TM is discussed in Section 5.3.

The TM multicasts command packets to deliver new tasks to the PUs, and receives report packets and postalert packets to maintain the states of the PUs. Upon receiving a report packet or a postalert packet, it updates the task table and PU table. The TM also maintains a queue for ready tasks. If the queue is not empty, it selects a task in the queue and a useable PU to carry on the task according to the scheduling policies. The task should be scheduled to meet its timing constraints. The information is packetized in a command packet which is then multicasted through the NoC. More information about the TM and the task/PU tables can be found in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.193.

Packets are generated by the TM or node agents based on the runtime information provided by the SENoC, transmitted through the NoC and analyzed by the node agents. Four types of packets are defined in the SENoC, including command packet, report packet, alert packet, and postalert packet. Detailed information can be found in the supplementary file, available online. For each type of packet, a tag including the information of the packet type, the sender, and the receiver is attached. For a command packet, the potential victims calculated by the TM are attached. Therefore, a node agent who receives a packet will be capable of determining the next action of the PU after combining with the information reported by the local sensors.

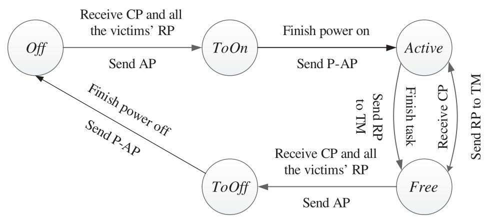

Fig. 6. The state transition machine of an attacker.

A node agent is capable of interpreting the information in the packets, and collecting real-time information from local sensors, according to some built-in functional components. Node agents generate a unified form of packets to ensure reliable and secure packet delivery and maintain the efficiency of the MPSoC. More information on the functional components can be found in the supplementary file, available online.

A PU p becomes an attacker when it is going to power on/off. Upon receiving the command packet for executing a task, and understanding its role as an attacker after analyzing it, p should know the set of its victims and wait for their report packets. After receiving the report packets from all its victims, it will multicast alert packets at the beginning of the powering on, multicast postalert packets at the end of the powering on, and unicast a report packet to tell its state of *Free* to the TM after finishing the task. The state transition of an attacker is shown in Fig. 6.

A PU q becomes a victim when it is in *Active* and a neighbor PU p is going to power on/off. For the sake of protecting its own process, q has to pause and change to the *Idle* state until p's powering action exerts no impact on q. The local sensors are used to detect this impact and trigger the transition. q experiences transient states *ClkToOff* when being protected, and *ClkToOn* when resurging. Report packets are sent to the attacker and the TM to tell its state and action. The state transition and packet delivery of a victim is shown in Fig. 7.

5.2 Overall Operations of the SENoC

The operations of the SENoC can be classified into the powering-on operation and the powering-off operation which are very similar. We use the powering on operation as an example in this section to explain the process. Note that we only allow one PU to conduct power gating at a time with this strategy. This simplified the TM design and its working efficiency. We describe the operations of the TM as follows when there is a task ready to execute.

If there is some PU in *Transient* state, do nothing. Otherwise, if there is no PU in *Transient* state, the TM selects a PU to execute the task. This operation is detailed as follows:

If there is no PU in *Free* or *Off*, do nothing. Otherwise, if there is a PU p in *Free*, it will be selected to execute the task. The TM synthesizes a command packet, multicasts it, and updates the PU table. p carries out the task after receiving the command. Otherwise, if there is a PU p in *Off*

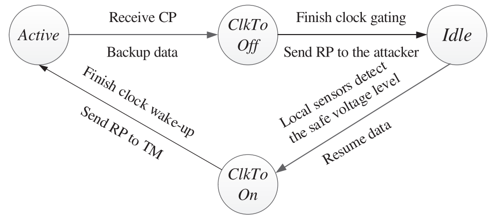

Fig. 7. The state transition machine of a victim.

with the least number of victims and relatively good performance, it will be selected and become an attacker. A new *Safe voltage* will be calculated by the TM according to p's performance and the previous safe voltage. The safe voltage is the minimum voltage level of a PU to keep it tolerant to the increased gate delay and guarantee correct execution. The TM multicasts a command packet. An involved PU will know the role of itself as an attacker or a victim after its node agent receives the command packet. Assume PU q is one of p's victims, and it transits to *ClkToOff* after receiving the command packet. During *ClkToOff*, it makes preparations for clock gating, including sending useful data to an external memory (off-chip memory) for backup, unicasts a report packet to p after finishing such operations, and transits to the *Idle* state during the same period.

After receiving report packets from all its victims, the attacker p multicasts an alert packet, and transits to *ToOn* during which it switches on sleep transistors and powers on. When its sensors report the safe voltage level, a postalert packet is generated and multicasted. A report packet is sent to the TM to update the PU table and task table. Meanwhile, p transits to the *Active* state and starts executing the task. For the victim q in the *Idle* state, it transfers to *ClkToOn* after its own sensors detect the *Safe voltage*. During the *ClkToOn* state, it prepares for resurging from clock gating, including loading relevant data from off-chip memory. q unicasts a report packet to the TM after finishing these operations, and at the meantime transits to the *Active* state and continues running the halted task. The TM will update q's information after receiving the report packet. After receiving all the report packets, the TM will update the PU table and task table accordingly. Then, the TM will move on to the next action of assigning a new task or turning off the PU.

This protocol design actually makes the system tolerant to faulty sensors which may be affected by a noisy environment. If the sensors on a victim PU cannot work properly, the PU will wait for the postalert packet sent by its attacker with the notice of power gating finished, and then can resume from the clock gated state safely. In this way, the proposed methodology can still guarantee the correctness of the system with faulty sensors, and the performance overhead compared to correct sensors is minimized to an upper bound constrained by the communication protocol. Meanwhile, there are multiple sensors distributed on each PU, which further enhances the degree of tolerance to faulty sensors. The timing delay of sensors can also be tolerated due to similar reasons.

5.3 Two-Stage Scheduling Strategy for Performance Optimization

In order to effectively utilize the MPSoC resources, it is essential to have an efficient scheduling strategy for the TM

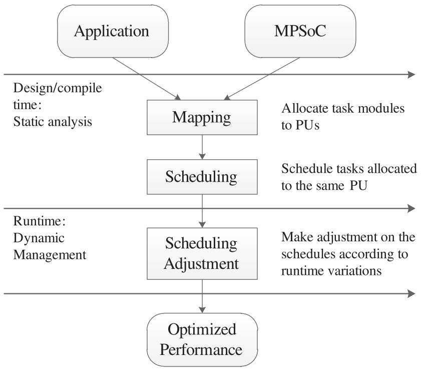
Application
MPSoC
Design/compile time: Static analysis
Mapping
Allocate task modules to PUs
Scheduling
Schedule tasks allocated to the same PU
Runtime: Dynamic Management
Scheduling Adjustment
Make adjustment on the schedules according to runtime variations
Optimized Performance

Fig. 8. Two-stage static scheduling and dynamic adjustment strategy.

to the centralized control of the entire system. State-of-the-art techniques for power gating aware scheduling are limited to online algorithms. We propose a new scheduling strategy that is to optimize the performance of the entire system in two stages.

The approach is a composition of an offline static mapping and scheduling algorithm as well as an online dynamic light-weight adjustment technique. At design time, without any online information, we try to optimize an application's performance on the MPSoC using static real-time scheduling techniques. At runtime, the centralized TM is aware of the situation of the entire system including the state of every PU and every task, and we develop a dynamic adjustment strategy that generally follows the static scheduling result obtained offline, but makes slight adjustment to the static decisions according to the practical information of task running variations due to power gating related operations. The combined static scheduling and dynamic adjustment (SSDA) strategy is proposed to optimize the overall system performance. The strategy overview is described in Fig. 8.

5.3.1 Optimizing Offline Scheduling by Static Analysis

We conduct static optimizations in order to maximize applications' performance theoretically according to the worst case estimation of task executions and network transmissions. Formally, given an application represented by a task graph $G(V, E)$, and an MPSoC PU, the problem is to find a mapping $M : V \rightarrow PU$ for each task in V to a PU in P, and a static-order schedule $S : V \rightarrow N$ for the tasks assigned to each PU, where each task in V mapped to a PU is assigned a natural number indicating its sequence of execution on that PU, such that the application performance is optimized. In static order scheduling, the tasks assigned to the same PU execute following a predefined sequence strictly. It is proven to be a more effective strategy than statical-time scheduling and list scheduling for multi-processor systems [39].

In the SSDA framework, many static optimization strategies can be applied for making task mapping and scheduling decisions, like the works presented in [13], [40], [41]. In this paper, we do not explore the performance difference by different strategies, but focus on the impact of the flexible framework. Based on the SSDA framework, different scheduling algorithms can be selected for further performance optimization, and we leave it as possible

TABLE 2
Definition of the Symbols Used in Offline Scheduling

Attribute	Description
$e(v)$	The worst-case execution time of task v
$l(v,u)$	The number of packets that task v generates to edge (v,u)
$d(p,q)$	The Manhattan distance between PUs p and q
$w(v,p)$	The weight of mapping task v to PU p
$t(v,p)$	The required time for task v to finish on PU p
$f(p)$	the time for executing previously assigned tasks on PU p
$n(v,p)$	The total amount of network transmission generated by task v when it is assigned to PU p
$m(v)$	The mapping of task v
$s(v)$	The schedule of task v

future work. We apply load balanced mapping and static-order scheduling strategies to obtain static scheduling result. The basic idea is to distribute processing and network transmission workload evenly and make high utilization to the hardware resources. The mapping strategy is to assign the tasks to PUs one by one in the order defined by the graph topology, and the schedule on each PU is determined by invoking the tasks mapped to the PU in self-timed manner. The objective is to minimize the end-to-end delay of the application's execution with the consideration of interprocessor communication overhead. For a task $v \in V$, the weight of mapping it to a PU $p \in P$ is calculated by the following cost function:

$$w(v,p) = c_1 t(v,p) + f c_2 n(v,p), \tag{1}$$

in which $t(v,p)$ is the required time for task v to finish execution on PU p, defined by the time for executing previously assigned tasks on p plus the worst case execution time of v, and $n(v,p)$ is the total amount of network transmission, defined by the number of packets generated by task v from p to other PUs (Table 2 lists the definitions of the symbols):

$$t(v,p) = f(p) + e(v), \tag{2}$$

$$n(v,p) = \sum_{(v,u)\in E} l(v,u) \times d(p, m(u)), \tag{3}$$

where c_1 and c_2 are the user-specified constant factors to trade off between the two concerns, and f is an architecture-specific scaling factor which balances the two terms in different measurement units. For example, setting $c_1 = 1, c_2 = 0$ will obtain the mapping result that balances the PUs' workload regarding execution time, and setting $c_1 = 2, c_2 = 1$ will balance PUs' workload as well as network traffic with bias toward PU workload. For performance evaluations in Section 6, we set both factors c_1 and c_2 to 1, meaning that task execution and network transmission are considered to be of the same importance to the system performance. In ideality, factor f can be set to 3 for the number of clock cycles a packet crossing a router, while the value should be larger when network contention occurs. The detailed mapping and scheduling algorithm with pseudocodes and explanations can be found in the supplementary file, available online.

5.3.2 *Improving Online Performance by Dynamic Adjustment*

We make use of online messages and conditions to make slight adjustment to the statically determined schedules with the intent of further improving the application performance. A lightweight online scheduling adjustment strategy is used to improve runtime performance under the uncertainties of task execution variations due to power gating operations.

In this work, task mappings are statically optimized and fixed at runtime to reduce the online management complexity. At runtime, the TM sends command packets to the respective PUs for task invocations according to the schedule table obtained by static analysis, and keeps a PU table to monitor the status of all the PUs. Each PU is able to send report packets to the TM with three kinds of events: task has started, task has finished, and task has halted. The dynamic scenarios of task executions will be kept in the schedule table and used by the TM for runtime decisions. The scheduling adjustment strategy follows the work-conserving principle to keep the PUs work efficiently. The TM keeps each PU work-conserving and invokes tasks in the defined order in the schedule table. The actual schedule should be similar with the offline result but slightly different. Node agent on each router handles power gating and task state change locally, but the incidents are reported to the TM to allow global synchronization.

The online adjustment algorithm is given in Algorithm 1. Function NextTask(p, S) (Lines 3, 5) is used to fetch the next task on PU p in the predefined static order. PU state reporting includes task v started, finished, and halted. The schedule table $st(V)$ contains the records of all predicted and actual task executions at offline and online. The algorithm allows dynamic adjustment on the predefined static order schedules to invoke ready tasks earlier if its preceding task is blocked for execution at runtime. The strategy is to keep PU work conserving, i.e., a PU cannot be free if some task assigned to it is ready for execution. Algorithm 1 has linear complexity to the input size of the problem and runs very fast in practice.

Algorithm 1. The light-weight online dynamic scheduling adjustment algorithm

```
Require: task graph $G(V,E)$, MPSoC $P$, mapping $M(V,P)$,
         static order schedules $S(V,N)$
 1: for each PU $p$ do
 2:   if $p$ reports free then
 3:     $v =$ NextTask($p, S$)
 4:     while $v$ is not ready do
 5:       $v =$ NextTask($p, S$)
 6:     end while
 7:     Schedule $v$ on $p$
 8:     Update $st(v)$
 9:   end if
10:   if $p$ reports PU state then
11:     Update $st(v)$
12:   end if
13: end for
14: return makespan and schedule table $st(V)$
```

The SSDA framework combines the offline static optimization and online dynamic lightweight adjustment to guarantee a correct and performance-effective system. The dynamic adjustments to the schedules at runtime are mainly induced by the power gating-related operations, like sensors reporting safe voltage level on victim PUs. This hybrid strategy offers both performance enhancement and flexibility to such scenarios.

6 Performance Evaluation

We first show the timing analysis for SENoC in Section 6.1. Based on the analysis, extensive simulations are performed to compare the performance of the MPSoCs with and without SENoC in Sections 6.2 and 6.3.

6.1 Timing Analysis for SENoC

SENoC and PUs consume time in several ways, such as the signal transfer time and processing time. A comprehensive timing analysis in the SENoC is provided in the supplementary file, available online, in which the time usage parameters to be used in the timing analysis are defined in detail. The values of these parameters are determined according to the time used by some unit operations (like a unit packet crossing a router, a power gating and clock gating operation, a sensor and node agent operation, etc.) and the runtime situations (like packet size, router buffering delay, network contention, etc.). In our simulations, we use the following reference values for the unit operations in the number of clock cycles. The time for a unit packet crossing a router is assumed to be 3 clock cycles. The time for a power gating operation is assumed to be at the magnitude of 200 clock cycles, and that for a clock gating operation is at the magnitude of 100 clock cycles [42]. These parameters are determined by the power/clock gating process, which are not optimized in this work. SENoC is compatible with circuit level techniques that can optimize these parameters and by integrating them the system performance can be further enhanced. The sensor and node agent are assumed to operate in 1 clock cycle [23]. The actual values of the parameters at runtime are determined based on these basic assumptions and the runtime conditions during simulation. The detailed timing analysis can be found in the supplementary file, available online.

6.2 Implementation and Simulation Setup

The simulations are based on the MPSoC P/G noise simulation and analysis platform, which is described in more detail in Section 4 and [33]. The P/G noise analysis platform is built up with HSPICE and C. Scheduling algorithms are implemented with C, Matlab, and SystemC, respectively. The experiments are performed on a server with 2 Intel Core2 Xeon and 8 GB memory. The simulations assume that the average power consumption of a single PU is 30 mW, and the area of a single PU is 660 $\mu\mathrm{m} \times 660\ \mu\mathrm{m}$. Based on the HSPICE simulation results using $45\ \mathrm{nm}$ standard logic cells, the noise toleration of $V_{dd} - V_{ss}$ is set to be $100\ \mathrm{mV}$, and hence V_{safe} is set to be $700\ \mathrm{mV}$. The corresponding impact range $IR(p)$ of each attacker p is derived for 4*4-core to 8*8-core MPSoCs. The P/G network RLC parameters are obtained from the PTM interconnect model [34]. The power consumption of the monitoring network for an 8*8 MPSoC is estimated to be 38.1 mW using HSPICE simulation, including all the sensors, node agents and the TM. It introduces around 1.9 percent overhead to the total power consumption.

We adopt four basic topological structures of tasks to make comparisons:

1. $TASK_{NC}$. Tasks with no correlation,
2. $TASK_{SP}$. Several sequential tasks in parallel,
3. $TASK_{TT}$. Tree-connected tasks,
4. $TASK_{FC}$. Fully correlated tasks (a connected DAG with multiple inputs and multiple outputs).

What's more, for task numbers of 60 and 80, we adopt two substructures for each task set of $TASK_{SP}$, $TASK_{TT}$, and $TASK_{FC}$: structure expanded by depth (designated by a subscript of d) and by width (designated by a subscript of w), from 40-task structures.

6.3 Simulation Results

In our simulation, we test MPSoC using SENoC and stop-go method for 6 to 80 tasks with different task structures. Both works are at the system level and are compatible with circuit-level techniques to further optimize system performance. The actual execution time is longer than ideal execution time for all the cases in our study. Difference in task structures, MPSoC scales, task numbers, and algorithms affects execution time differently. We define $r_i = \frac{T_{end}(SENoC) - T_{end}(ideal)}{T_{end}(ideal)}$ to measure the efficiency of the SENoC approach, where $T_{end}(SENoC)$ and $T_{end}(ideal)$ denote the finish time for all the tasks in the SENoC and the ideal case assuming power gating is not adopted, respectively. To compare different methods directly, we define $r_s = \frac{T_{end}(stop\text{-}go) - T_{end}(SENoC)}{T_{end}(stop\text{-}go)}$ to compare the SENoC and stop-go methods, where $T_{end}(stop - go)$ denotes the finish time for all the tasks in the stop-go approach. Results show that the SENoC helps to achieve an average performance improvement of 26.2 percent on average for an 8*8-core MPSoC. We estimate the area overhead based on the $45\ \mathrm{nm}$ design and the NoC simulations using NS2 [43]. The area overhead of the SENoC for the 8*8-core MPSoC is 1.4 percent. More detailed experimental results for different task structures in 8*8 MPSoC can be found in the supplementary file, available online.

Fig. 9 shows the performance improvement of MPSoC using SENoC with 40 tasks under different task structures and MPSoC scales. The performance improvement increases dramatically as the MPSoC scale increases in $TASK_{NC}$. However, for task structures $TASK_{SP}$, $TASK_{TT}$, and $TASK_{FC}$, the increase in performance is not that obvious. The tasks in $TASK_{NC}$ can run in higher level of parallelism since they don't have dependency relation with each other. As the number of processors increases, the benefits from using the impact range in the SENoC become more obvious because lower number of PUs are affected by power gating compared to the conservative stop-go method. The improvements for the other task set structures are relatively limited due to the data dependencies that reduce the tasks and PUs running in parallel. Generally, the trend of increase in performance slows down when MPSoC scale increases.

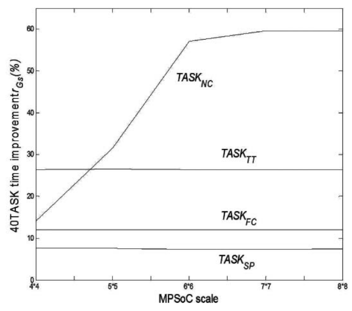

Fig. 9. Performance improvement of MPSoC using SENoC with 40 tasks under different task structures and MPSoC scales.

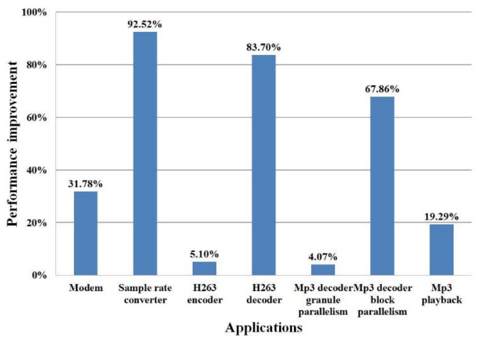

Fig. 10. Performance improvement of the SENoC compared to the stop-go method for realistic applications.

This is due to the fact that a relatively small number of PUs can handle the given set of tasks.

Besides, the finish times of $TASK_{CP}$ and $TASK_{PT}$ using SENoC is reduced drastically in most cases. This indicates that the SENoC helps reduce the number of power off and on, thus effectively avoids powering on/off induced P/G noise. Therefore, a MPSoC can work more safely with higher reliability by using SENoC.

In order to further verify the effectiveness of the proposed approach for real-life problems, we develop a cycle-accurate SENoC simulation model using SystemC [44], and conduct extensive performance evaluations on several well-known realistic DSP and multimedia applications [45]. We compare our solution with the traditional stop-go method on the total performance. The simulator is built with parameterized configurations, where parameters like MPSoC size, global clock frequency, cache size, etc., are adjustable. We construct a mesh-based MPSoC architecture with 4*4 processing units, XY routing and wormhole switching protocols. Task executions are simulated with random execution times following Gaussian distribution with mean of 80 percent of their worst case execution times and variance of 0.2. The global clock frequency is set to 1.25 GHz. The enhanced scheduling strategies proposed in Section 5.3 are applied for further performance improvement.

Fig. 10 shows the experimental results given by the relative performance improvement of our technique over the traditional stop-go method. Results show that our approach outperforms the stop-go method by 43.5 percent on average. There are mainly three reasons for the improvement. First, sensors are used to detect the safe voltage level, instead of waiting for the attacker's postalert packet, which significantly reduces the time that victims are in the clock gating state, especially when the network is congested by traffic transmission which may cause long delay of packet delivery. Second, the improved power gating induced P/G noise model reduces the number of protecting PUs so that the power gating efficiency is improved. Last, the further improvements benefit from the more efficient scheduling strategy that helps reduce the performance overhead induced by power gating and improve the system resource utilization. Most operations in the SENoC are actually distributed to the respective PUs and node agents. The workload of the centralized TM mainly comes from the lightweight dynamic scheduling adjustment algorithm (Algorithm 1), which has linear complexity and can scale to larger number of processors. The NoC uses virtual channels to deliver control packets in higher priority than payload packets. This guarantees the control message delivery between the TM and PUs will not be delayed by regular data communications. Therefore, the system can scale well for large applications and MPSoCs.

7 Conclusion

This paper proposes a systematic approach, on-chip sensor network (SENoC), which not only detects reliability threats and shares related information among PUs, but also plans and coordinates the reactions of related PUs in MPSoC. SENoC is integrated with NoC to ensure that critical information and decision is delivered in a timely fashion. SENoC is applied to alleviate the impacts of simultaneous switching noise in MPSoC's P/G network during power gating. Based on the detailed noise behaviors under different scenarios derived by our circuit-level MPSoC P/G noise simulation and analysis platform, it shows that SENoC helps to achieve on average 26.2 percent performance improvement compared with the traditional stop-go method with 1.4 percent area overhead in an 8*8-core MPSoC in 45 nm. With the enhanced scheduling techniques applied to offset the overheads of power gating related operations and optimize the total system performance, a higher improvement of 43.5 percent is observed for a set of real-life applications by architecture-level cycle-accurate simulations based on SystemC.

Acknowledgments

This work is partially supported by HK GRF (620911, 621108), HKUST PDF, DAG07/08.EG04, SBI06/07.EG01-4, NSTMP (2011ZX01035-001-001-002, 2010ZX01030-001-001-04, 2010ZX01030-001), and NSFC (60870001, 61028006).

References

[1] S. 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 J. Solid-State Circuits*, vol. 43, no. 1, pp. 29-41, Jan. 2008.

[2] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C.-C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook, "Tile64 - Processor: A 64-core Soc with Mesh Interconnect," *Proc. IEEE Int'l Solid-State Circuits Conf. (ISSCC '08)*, pp. 88-598, 2008.

[3] S. Borkar, "Thousand Core Chips: A Technology Perspective," *Proc. 44th Ann. Design Automation Conf. (DAC '07),* pp. 746-749, 2007.

[4] F. Mohamood, M. Healy, S.K. Lim, and H.-H. Lee, "Noise-Direct: A Technique for Power Supply Noise Aware Floorplanning Using Microarchitecture Profiling," *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC '07),* pp. 786-791, 2007.

[5] M. Healy, F. Mohamood, H.-H.S. Lee, and S.K. Lim, "A Unified Methodology for Power Supply Noise Reduction in Modern Microarchitecture Design," *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC '08),* pp. 611-616, 2008.

[6] H. Jiang, M. Marek-Sadowska, and S. Nassif, "Benefits and Costs of Power-Gating Technique," *Proc. IEEE Int'l Conf. Computer Design: VLSI in Computers and Processors (ICCD '05),* pp. 559-566, Oct. 2005.

[7] Y. Wang, J. Xu, S. Huang, W. Liu, and H. Yang, "A Case Study of On-Chip Sensor Network in Multiprocessor System-On-Chip," *CASES '09: Proc. Int'l Conf. Compilers, Architecture, and Synthesis for Embedded Systems,* pp. 241-250, 2009.

[8] K. Shi and D. Howard, "Challenges in Sleep Transistor Design and Implementation in Low-Power Designs," *Proc. 43rd ACM/IEEE Design Automation Conf.,* 2006.

[9] S. Kim, S. Kosonocky, and D. Knebel, "Understanding and Minimizing Ground Bounce during Mode Transition of Power Gating Structures," *Proc. Int'l Symp. Low Power Electronics and Design (ISLPED '03),* pp. 22-25, 2003.

[10] S. Kim, S. Kosonocky, D. Knebel, K. Stawiasz, D. Heidel, and M. Immediato, "Minimizing Inductive Noise in System-on-a-Chip with Multiple Power Gating Structures," *Proc. 29th European Solid-State Circuits Conf. (ESSCIRC '03),* pp. 635-638, 2003,

[11] K. He, R. Luo, and Y. Wang, "A Power Gating Scheme for Ground Bounce Reduction during Mode Transition," *Proc. 25th Int'l Conf. Computer Design (ICCD '07),* pp. 388-394, 2007.

[12] M. Gupta, J. Oatley, R. Joseph, G.-Y. Wei, and D. Brooks, "Understanding Voltage Variations in Chip Multiprocessors Using a Distributed Power-Delivery Network," *Proc. Design, Automation Test in Europe Conf. Exhibition (DATE '07),* pp. 1-6, Apr. 2007.

[13] V. Reddi, M. Gupta, G. Holloway, M. Smith, G.-Y. Wei, and D. Brooks, "Predicting Voltage Droops Using Recurring Program and Microarchitectural Event Activity," *IEEE Micro,* vol. 30, no. 1, p. 110, Jan./Feb. 2010.

[14] J. Gu, H. Eom, and C. Kim, "A Switched Decoupling Capacitor Circuit for On-Chip Supply Resonance Damping," *Proc. IEEE Symp. VLSI Circuits,* pp. 126-127, 2007.

[15] M. Powell and T. Vijaykumar, "Pipeline Damping: A Microarchitectural Technique to Reduce Inductive Noise in Supply Voltage," *Proc. 30th Ann. Int'l Symp. Computer Architecture,* pp. 72-83, June 2003.

[16] M. Powell and T. Vijaykumar, "Exploiting Resonant Behavior to Reduce Inductive Noise," *Proc. 31st Ann. Int'l Symp. Computer Architecture,* pp. 288-299, June 2004.

[17] H. Jiang and M. Marek-Sadowska, "Power Gating Scheduling for Power/Ground Noise Reduction," *Proc. 45th ACM/IEEE Design Automation Conf. (DAC '08),* pp. 980-985, 2008.

[18] H. Jiang and M. Marek-Sadowska, "Power-Gating Aware Floorplanning," *Proc. Eighth Int'l Symp. Quality Electronic Design (ISQED '07),* pp. 853-860, 2007.

[19] D. Ernst, N.S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture,* pp. 7-18, Dec. 2003.

[20] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N.S. Kim, and K. Flautner, "Razor: Circuit-Level Correction of Timing Errors for Low-Power Operation," *IEEE Micro,* vol. 24, no. 6, pp. 10-20, Nov.-Dec. 2004.

[21] M. Nicolaidis, "Design for Soft Error Mitigation," *IEEE Trans. Device and Materials Reliability,* vol. 5, no. 3, pp. 405-418, Sept. 2005.

[22] W. Liu, J. Xu, X. Wang, Y. Wang, W. Zhang, Y. Ye, X. Wu, M. Nikdast, and Z. Wang, "A Hardware-Software Collaborated Method for Soft-Error Tolerant Mpsoc," *Proc. IEEE CS Ann. Symp. VLSI,* pp. 260-265, July 2011.

[23] V. Petrescu, M. Pelgrom, H. Veendrick, P. Pavithran, and J. Wieling, "Monitors for a Signal Integrity Measurement System," *Proc. 32nd European Solid-State Circuits Conf. (ESSCIRC '06),* pp. 122-125, 2006.

[24] C. Poirier, R. McGowen, C. Bostak, and S. Naffziger, "Power and Temperature Control on a 90nm Itanium Reg;-Family Processor," *Proc. IEEE Int'l Solid-State Circuits Conf.,* vol. 1, pp. 304-305, 2005.

[25] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, M. Millican, W. Parks, and S. Naffziger, "Power and Temperature Control on a 90-nm Itanium Family Processor," *IEEE J. Solid-State Circuits,* vol. 41, no. 1, pp. 229-237, Jan. 2006.

[26] K. Sohn, N. Cho, H. Kim, K. Kim, H.-S. Mo, Y.-H. Suh, H.-G. Byun, and H.-J. Yoo, "An Autonomous SRAM with On-Chip Sensors in an 80nm Double Stacked Cell Technology," *Proc. Symp. VLSI Circuits,* pp. 232-235. 2005,

[27] K. Sohn, H.-S. Mo, Y.-H. Suh, H.-G. Byun, and H.-J. Yoo, "An Autonomous SRAM with On-Chip Sensors in an 80-nm Double Stacked Cell Technology," *IEEE J. Solid-State Circuits,* vol. 41, no. 4, pp. 823-830, Apr. 2006.

[28] N. Pandit, Z. Kalbarczyk, and R. Iyer, "Effectiveness of Machine Checks for Error Diagnostics," *Proc. IEEE/IFIP Int'l Conf. Dependable Systems Networks (DSN '09),* pp. 578-583, July 2009.

[29] C. Chan, Y. Chang, H. Ho, and H. Chiueh, "A Thermal-Aware Power Management Soft-IP for Platform-Based SoC Designs," *Proc. Int'l Symp. System-on-Chip,* pp. 181-184, 2004.

[30] A.W. Yin, L. Guang, P. Liljeberg, P. Rantala, E. Nigussie, J. Isoaho, and H. Tenhunen, "Hierarchical Agent Architecture for Scalable Noc Design with Online Monitoring Services," *Proc. First Int'l Workshop Network on Chip Architectures,* 2008.

[31] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, "An Event-Based Network-On-Chip Monitoring Service," *Proc. Ninth IEEE Int'l High-Level Design Validation and Test Workshop,* pp. 149-154, 2004.

[32] C. Ciordas, K. Goossens, A. Radulescu, and T. Basten, "NoC Monitoring: Impact on the Design Flow," *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS '06),* 2006.

[33] Y. Xu, W. Liu, Y. Wang, J. Xu, X. Chen, and H. Yang, "On-Line Mpsoc Scheduling Considering Power Gating Induced Power/Ground Noise," *Proc. IEEE CS Ann. Symp. VLSI,* pp. 109-114, 2009.

[34] W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for sub-45 nm Early Design Exploration," *IEEE Trans. Electron Devices,* vol. 53, no. 11, pp. 2816-2823, Nov. 2006.

[35] Nangate Open Cell Library, http://www.opencelllibrary.org, 2012.

[36] S. Pant and E. Chiprout, "Power Grid Physics and Implications for Cad," *Proc. 43rd ACM/IEEE Design Automation Conf.,* pp. 199-204, 2006.

[37] A. Balijepalli, S. Sinha, and Y. Cao, "Compact Modeling of Carbon Nanotube Transistor for Early Stage Process-Design Exploration," *Proc. ACM/IEEE Int'l Symp. Low Power Electronics and Design (ISLPED),* pp. 2-7, Aug. 2007.

[38] S. Kim, S. Kosonocky, D. Knebel, K. Stawiasz, and M. Papaefthymiou, "A Multi-Mode Power Gating Structure for Low-Voltage Deep-Submicron CMOS ICs," *IEEE Trans. Circuits and Systems II: Express Briefs,* vol. 54, no. 7, pp. 586-590, July 2007.

[39] W. Liu, M. Yuan, X. He, Z. Gu, and X. Liu, "Efficient Sat-Based Mapping and Scheduling of Homogeneous Synchronous Dataflow Graphs for Throughput Optimization," *RTSS '08: Proc. Real-Time Systems Symp.,* pp. 492-504, 2008.

[40] W. El-Essawy and D. Albonesi, "Mitigating Inductive Noise in SMT Processors," *Proc. Int'l Symp. Low Power Electronics and Design,* pp. 332-337, Aug. 2004.

[41] W. Liu, Z. Gu, J. Xu, X. Wu, and Y. Ye, "Satisfiability Modulo Graph Theory for Task Mapping and Scheduling on Multiprocessor Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 22, no. 8, pp. 1382-1389, Aug. 2011.

[42] K.-i. Kawasaki, T. Shiota, K. Nakayama, and A. Inoue, "A Sub-Us Wake-Up Time Power Gating Technique with Bypass Power Line for Rush Current Support," *Proc. IEEE Symp. VLSI Circuits,* pp. 146-147, June 2008.

[43] NS2, http://nsnam.isi.edu/nsnam, 2012.

[44] SystemC, http://www.systemc.org, 2012.

[45] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast, and Z. Wang, "A Noc Traffic Suite Based on Real Applications," *Proc. IEEE CS Ann. Symp. VLSI,* pp. 66-71, July 2011.

Weichen Liu (S'07-M'11) received the BEng and MEng degrees from the Harbin Institute of Technology in 2004 and 2006, respectively, and the PhD degree from the Hong Kong University of Science and Technology in 2011, all in computer science and engineering. As visiting scholars, he worked in Signal Processing and Wireless Communications Lab at the National Tsinghua University of Taiwan in 2005, Beijing Key Lab on Intellectual Communication Software and Multimedia at Beijing University of Post and Telecommunications in 2006, and Mobile Computing Systems Lab at the Hong Kong University of Science and Technology in 2011-2012, respectively. He served on the technical program committee of the 25th International Conference on VLSI Design, and served as a technical reviewer of more than 20 premier international journals and conferences. He authored and coauthored more than 30 research papers in peer-reviewed international journals, conferences, and books. He received the Best Paper Candidate Award from the Seventh IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2009), and the Best Poster Award from the Sixth Annual AMD Technical Forum and Exhibition (AMD-TFE 2010). His research interests include embedded systems, real-time systems, multiprocessor systems, system-on-chip, network-on-chip, hardware/software codesign, mobile computing, formal methods, and design space exploration. He is a member of the IEEE.

Yu Wang (S'05-M'07) received the BS degree in 2002, and then PhD degree with honor in NICS Group, Electronics Engineering Department, Tsinghua University, in 2007, supervised by Prof. Huazhong Yang (Tsinghua University) and Prof. Yuan Xie (Penn. State University). He is currently an associate professor in the Electronics Engineering Department, Tsinghua University. His research mainly focuses on fast/parallel circuit analysis, low power and reliability aware circuit design methodology, application specific hardware computing, and on-chip communication strategies for MPSOC. He has authored and coauthored more than 70 papers in refereed journals and conferences. Among them, five conference papers are nominated as the best paper candidate in ISLPED 2009, CODES 2009, ASPDAC 2010, ASPDAC 2012. He is a TPC co-chair of ICFPT 2011 and publicity co-chair of ISLPED 2011; he is also a TPC member for several conferences, such as ICCAD, ISQED, ISVLSI, ISLPED, DATE, etc. He is a member of the IEEE.

Xuan Wang received the BS degree in electrical engineering from Shanghai Jiaotong University, Shanghai, China, in 2009. Since 2009, he has been working toward the PhD degree in the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST). His research interests include embedded system, multiprocessor system, network-on-chip and fault-tolerant design and reliability issues in very deep submicron technologies. He is a student member of the IEEE.

Jiang Xu (S'02-M'07) received the PhD degree from Princeton University in 2007. From 2001 to 2002, he was at Bell Labs, NJ, as a research associate. He was a research associate at NEC Laboratories America, NJ, from 2003 to 2005. He joined a startup company, Sandbridge Technologies, NY, from 2005 to 2007 and developed as well as implemented two generations of NoC-based ultralow power multiprocessor systems-on-chip for mobile platforms. In 2007, he joined the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology as an assistant professor, and established the Mobile Computing System Lab. He currently serves as an associate editor of *ACM Transactions on Embedded Computing Systems* and *IEEE Transactions on Very Large Scale Integration Systems*. He is an ACM Distinguished Speaker and a Distinguished Visitor of IEEE Computer Society. He served on the organizing committees and technical program committees of many international conferences. He authored or coauthored more than 50 book chapters and papers in peer-reviewed journals and international conferences. He coauthored a book titled *Algorithms, Architecture and System-on-Chip Design for Wireless Applications* (Cambridge University Press). His research areas include network-on-chip, multiprocessor system-on-chip, embedded system, computer architecture, low-power VLSI design, and HW/SW codesign. He is a member of the IEEE.

Huazhong Yang (M'97-SM'00) received the BS degree in microelectronics in 1989 and the MS and PhD degrees in electronic engineering in 1993 and 1998, respectively, all from Tsinghua University, Beijing. In 1993, he joined the Department of Electronic Engineering, Tsinghua University, Beijing, where he has been a full professor since 1998. He was recognized as "2000 National Palmary Young Researcher" by NSFC. His research interests include chip design for communication and multimedia applications, synthesis of analog integrated circuits (IC), power estimation and synthesis of digital ICs, noise and delay estimation of deep submicron ICs, yield enhancement, optimization and modeling. He has been in charge of several projects, including projects sponsored by the 863 program, NSFC, the 11th five-year national program and several international cooperation projects. He has authored and coauthored more than 200 technical papers and six books. He is a senior member of the IEEE.