A Broadcast-Enabled Sensing System for Embedded Multi-core Processors

Jia Zhao, Shiting (Justin) Lu, Wayne Burleson, and Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003

***Abstract*—Contemporary multi-core architectures deployed in embedded systems are expected to function near the operational limits of temperature, voltage, and device wear-out. To date, most on-chip sensing systems have been designed to collect and use sensor information for these parameters locally. In this paper, a new sensing system to enhance multi-core dependability which supports both the local and global distribution of sensing data in embedded processors is considered. The benefit of the new sensing architecture is verified using the broadcast of microarchitectural parameter signatures which can be used to identify impending voltage droops. Low-latency broadcasts are supported for a range of sensor data transfer rates. Up to a 9% performance improvement for a 16-core system is determined via the use of the distributed voltage droop sensor information (5.4% on average). The entire sensing system, including broadcasting resources, requires about 2.6% of multi-core area.**

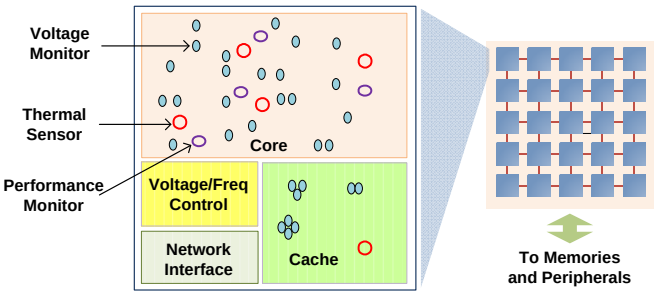

Fig. 1. Embedded multi-core processor including on-chip sensors. Multi-cores typically contain hundreds of sensors and per-core voltage and frequency control.

I. Introduction

As the range of embedded systems has expanded, the use of multi-core processors in these systems has become widespread. These processors often require access to substantial amounts of on-chip environmental data related to temperature, voltage, and run-time errors to operate dependably. On-line system adaptation, such as dynamic voltage and frequency scaling (DVFS), wear-out based enabling of redundant components, and fault recovery all rely on the availability and analysis of temperature, voltage, error, and processor performance data, among others (Fig. 1). The need to collect and use this information has motivated the development of a number of on-chip sensing systems [1][2][3] that collect sampled data from on-chip sensors, transport it to an analysis point, process the information, and dynamically adapt the embedded multi-core. In general, these on-chip sensing systems are characterized by two features:

- Most sensor data is collected and used in a confined, local portion of the multi-core die.
- Data transfer for these sensor systems generally takes place via unicast, source-to-single destination communication. Given the reduced bandwidth of sensor data versus typical multi-core application data, the on-chip communication and analysis infrastructure for sensing data is generally of limited complexity.

Commercially, both IBM and Intel have integrated sensor data collection resources into their multi-core devices. IBM EnergyScale [4] uses temperature and critical path monitors along with a microcontroller for sensor data processing. Intel's Active Management Technology [5] provides a separate on-chip communications channel to monitor device operation and control system responses at the operating system level. Both implementations and recent academic studies [3] have shown the importance of *dedicating* small amounts of on-chip interconnect and compute resources to address sensor data collection and processing. In general, commercial systems do not share sensor data interconnect with the existing multi-core interconnect for regular data (e.g. bus, network-on-chip). Additionally, it has been shown [3] that sharing one inter-core interconnect for *both* sensor and application data can lead to significant performance and reliability issues for multi-cores. As a result, a series of limited, dedicated on-chip sensor data systems which include the collection and processing of multi-core sensor data have been introduced.

Although these sensor data systems have proved useful for a number of on-chip sensing applications, a wider range of multi-core adaptation uses necessitates an enhanced infrastructure which provides the ability to use critical sensor data to control numerous processors across the die. This enhancement must be implemented while maintaining the very low overhead and minimal latency expected of sensor data processing infrastructures. In this paper, this type of globally-aware sensor infrastructure is introduced. The sensing system includes a lightweight communication network-on-chip (NoC) substrate which allows for sensor data unicast and *broadcast* without burdening the main inter-core on-chip communication system (e.g. bus or network-on-chip). A dedicated on-chip sensor controller is used to evaluate sensor data and determine appropriate system-level responses for the embedded system.

To fully assess the broadcast-enabled sensor data infrastructure, our new architecture is evaluated with the use of sensor data obtained from processor performance registers, voltage droop sensors, and temperature sensors. We demonstrate that the infrastructure not only allows for the unicast transfer of thermal information to a sensor controller, but also that new uses of sensor data are possible using sensor data broadcasting. In many cases, impending core voltage droops can be predicted before they occur through the use of processor performance information which is combined into a *signature* [6]. In this work we demonstrate that, once discovered, these signatures can be shared across all processor cores and used to reduce per-core voltage droops on a global scale, improving embedded processor dependability. Our enhanced sensor data infrastructure is used to broadcast voltage droop signatures when they are discovered. A significant performance improvement is achieved for signature broadcasting versus the case when all signatures are determined and used locally.

II. Background

A. Previous Multi-core Sensing Systems

The use of independent on-chip infrastructure to collect, analyze, and use sensor data has been of increasing interest in commercial and academic settings in the recent past. Bouajila *et al.* [7] use a non-scalable token ring interconnect for transporting sensor data, although the use of this data is unclear. Fattah [8] and Guang *et al.* [1] introduce the conceptual idea of collecting sensor data in individual cores and forwarding them hierarchically to a centralized controller. No discussion of the hardware needed to perform data analysis or system control is presented. Phanibhushana *et al.* [9] limit their study to a tree-like interconnect for sensor data collection. Although lightweight, this type of interconnect does not allow for the broadcast of time-critical sensor data.

Two more complete on-chip sensor systems evaluate data collection for multiple types of sensors and hundreds of sensors per die. Ituero *et al.* [2] present a full sensor data collection system for several types of sensors. The interface to a sensor data controller is defined along with latency and area overheads. No experiments were performed using sensor data broadcast or feedback from the controller to affect multi-core behavior. Zhao *et al.* [3] examine the collection of sensor data, its analysis, and use in DVFS. The use of sensor data broadcast is not considered. Similar work by the authors is targeted at sensor data driven DVFS implementation in 3D many-cores.

Several projects have examined on-chip communications technology which supports the broadcast of application data (not sensor data) to all or most cores. Peng *et al.* [10] describe extra control circuitry added to a NoC which allows for broadcasts using crossbars. Moadeli and Vanderbauwhede [11] describe a NoC protocol which allows for high-level wormhole multicast for data destined for multiple applications. In general, these approaches are not appropriate for the lightweight, scalable interconnect required for the transfer and analysis of sensor data to support on-chip embedded multi-core control.

B. Sensor Data Broadcasting Applications

The on-chip sensing infrastructures described in the previous section have been used for a variety of on-chip sensing and control applications for embedded applications. In general, the approaches follow a model where sensor data is collected at the cores, sent to a centralized location, and remediation responses (DVFS, system reconfiguration based on wear-out) are determined and administered with an additional set of messages sent to the affected cores. The availability of sensor data broadcast enables additional types of sensor data uses for embedded processors including:

- If an on-chip security sensor detects suspected tampering [12], messages can be sent to all processor cores protecting their memory and processing in a timely fashion.
- In the event of a voltage droop in a specific core, information about the droop can be quickly sent to other cores so residual effects can be monitored and addressed.
- Voltage droops can be *predicted* via the use of processor performance information which is compressed to form signatures. In this paper, we show that these signatures can be broadcast across the chip using the new sensor infrastructure to identify and eliminate future voltage droops, enhancing system dependability.

While the first two points are self-descriptive, the final point requires clarification. Large current swings can cause processor core voltage to swing beyond acceptable levels. Out-of-range supply voltages frequently lead to incorrect computation, requiring processor execution to be stopped and rolled back to a previously-stored intermediate checkpoint with known-good state. Voltage droops can be detected using voltage sensors based on technologies such as ring oscillators. In addition to processor rollback, the identification of a droop often leads to frequency reduction. Since checkpoints generally occur infrequently (once every 100 to 1000 instruction executions), a rollback causes a significant application performance hit. Therefore, if a droop can be predicted before it occurs, improved application performance can be achieved.

One approach for identifying voltage droops before they happen involves the use of combinations of processor performance statistics. These predictors [6] use information regarding cache misses, branches, and translation look-aside buffer (TLB) misses, among others, to identify voltage droops. When a voltage droop occurs, these parameters are measured and combined to create a signature which can be used to flag an upcoming droop the next time they have similar values. After a signature is identified, its information can be stored locally and used to dynamically identify if another droop is imminent during execution. This action allows for a frequency reduction *before* the droop occurs, eliminating the need for a costly rollback. Our broadcast-enabled sensor infrastructure *shares* these signatures across multiple cores once they are discovered. This action eliminates the need for subsequent voltage droop/rollback sequences in cores that differ from the one where the signature was first identified.

III. Broadcast-Enabled Sensor Data Architecture

Our broadcast-enabled sensor data infrastructure includes a dedicated, low-complexity NoC and a sensor controller (SC). Data from the sensors are sent to the sensor data router (SDR) using a packetizer module. The basic structure of the SDR is similar to a standard five-port NoC router with a crossbar switch and input queues, but it has been extended to support data broadcasting. Data received by the core from the SDR via a depacketizer module can be used to influence core behavior (e.g. voltage or frequency scaling). State machine controllers are used to control the interfaces. Given the volume of sensors in a typical core, an allocation of one SDR per core is used. Five input and five output queues (one per port) are used to store data packets in transit in the sensor data router. In general, this router implementation is simplified from standard NoC implementations for multi-cores, making it lightweight. Data bit widths are limited to 16 to 24 bits and buffering is limited to six flits per I/O queue.

Like the processor cores to which they interface, SDRs are organized in a 2D mesh. In unicast (single source to single destination) mode, sensor data are forwarded from individual cores to a sensor controller. SC functionality can be implemented in a small, dedicated microcontroller or in one of the cores of the multi-core. The SC uses collected thermal, wear-out, and voltage information to make decisions about individual core remediation such as DVFS or redundant resource activation. This model of data transfer to SCs, while appropriate for many remediation responses, is insufficient for some multi-core responses which require sensor data to be *broadcast* to all cores.

The broadcast control block of the SDR examines the packet header to determine if a broadcast bit is set. If so, the crossbar is set so that multiple destination ports and the processor core port are selected as destinations for the packet. An XY routing algorithm is used to promote the broadcasting of packets which originate from a single source. A packet is first sent in a north-south direction to all nodes in a column. When a packet is received at an adjacent node, it is then forwarded in an east-west direction to nodes in the respective row.

To evaluate the size of the broadcast-enable SDR, the design was written in Verilog and synthesized using Synopsys Design Compiler. A 6-flit buffer size was used for each of the five SDR ports. Each sensor data router (one per core), including interfaces, was determined to require 26,773 logic gates and require 0.068 mm^2 in 90 nm technology. This hardware cost is less than 0.3% of the hardware area of an Alpha 21264 core (7.7 million gates) [13]. The hardware cost of the additional broadcast controller is small (less than 10% of the sensor data infrastructure) since it mainly involves control logic. The power consumption of the SDR is also about 0.3% of the processor core consumption.

IV. Analysis of a Broadcast-Enabled Sensor Data System

As mentioned in Section II, the broadcasting of sensor data can be used in an embedded multi-core for security and voltage droop remediation, among other embedded processing dependability uses. To demonstrate the effectiveness of our approach, we analyze a realistic scenario which includes the unicast and broadcast of sensor information for an embedded multi-core environment.

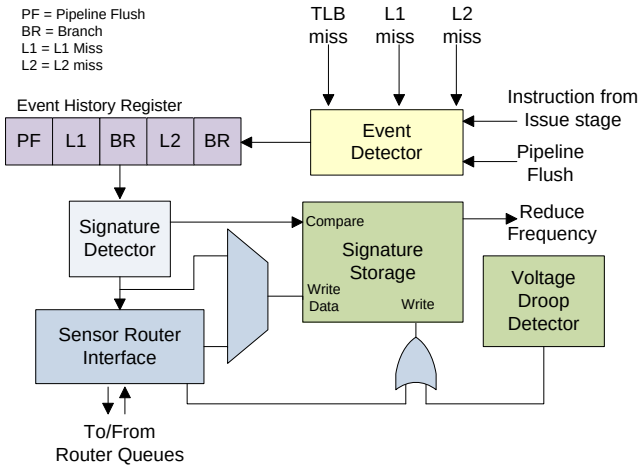

Fig. 2. Signature-based voltage droop avoidance module

Fig. 2 illustrates the voltage droop analysis module in the sensor interface of each core of a multi-core system and its connection to the associated SDR. The module dynamically stores select processor state information regarding cache misses, TLB misses, processor pipeline flushes, and control flow branches in an *event history register* [6]. Each event is represented using a three-bit code. Additionally, the program counter of the last branch instruction is stored in the signature. For 32 consecutive events, the signature requires 3 bits $\times$ 32 = 96 bits (12 bytes) plus 4 bytes for the stored *anchor* PC leading to a total of 16 bytes of signature. In the event of a voltage droop, the signature is stored, providing an early warning since similar future computations would likely also cause a droop. As execution progresses after the droop and associated processor rollback, the current signature is dynamically compared against stored signatures to determine if another voltage droop is imminent. If so, frequency is reduced for a period of time and the droop (and associated time-consuming rollback) is eliminated. Per-core signature storage includes a 128-entry content-addressable memory (CAM) which is used to match the *anchor* PC. The remaining portion of the signatures can be efficiently stored in a 8 KB Bloom filter [6]. Following synthesis it was determined that the signature generation and storage hardware in Fig. 2 represents a roughly 2.3% area overhead versus the Alpha 21264 processor.

The availability of the broadcast-enabled sensor system allows signatures generated in one core to be shared with all other cores which execute similar code segments. This action can prevent voltage droops in multiple cores even though the signature was identified in a single core. Our signature sharing method requires each core to have its own signature storage for both locally-created and broadcast-received signatures. When

a core generates a signature, it is transferred to the attached SDR and broadcast to other cores using the broadcast control module. When the 17-byte packet (1-byte header packet with routing destination information and the 16-byte signature) arrives at a core, it is stored in the core's signature storage. In Section VI, the results of this sharing are quantified.

V. Experimental Approach

To evaluate the benefits of using sensor data broadcasting, two simulators, Popnet [14] and SESC[1], have been modified and used in tandem. Enhancements were made to the Popnet interconnect simulator to determine latency and throughput for the lightweight sensor NoC. The interface has been integrated into the simulated router along with broadcast capabilities. This modified simulator provides a cycle-by-cycle analysis of sensor data transfer.

The widely-used SESC simulator provides execution performance results for our multi-core system. Parameter details used for simulation are presented in Table I. Wattch and CACTI are used to evaluate Alpha 21264 processor and cache power models, respectively. Voltage droop is determined by convolving the Alpha 21264 voltage supply system impulse response with per-cycle power consumption [15]. A voltage droop within a 4% operating margin necessitates a 500 cycle rollback recovery [6]. The generation of signatures from microarchitectural parameters is performed using the circuitry shown in Fig. 2, which uses instruction, cache, and pipeline flush information. Two multi-core configurations are considered: a 16-core system organized in a 4 $\times$ 4 mesh and an 8-core system organized as a 3 $\times$ 3 mesh with one core removed.

Our experiments assume that 8 temperature sensors are located per core [3]. Data from these sensors are sampled once every 800 clock cycles and forwarded to the sensor controller via the lightweight NoC. So, effectively, a new temperature sensor data packet is injected into the NoC by a core once every 100 cycles. Like [6], we assume that each core includes an event history register and signature storage, as shown in Fig. 2. Each signature includes a collection of 32 events. Signatures are stored in a signature storage buffer for every core. Both signatures generated locally (within the core) and globally (generated by another core and transferred to the core via the sensor data interconnect) can be stored in the table. Since signatures are tied to specific instruction sequences, the same code must be executed by multiple cores in the case of a global signature match. In the global case, signatures generated in a core are broadcast to all other cores, effectively sharing the sensor data interconnect with the temperature sensor data. It is assumed that voltage for each core is individually provided so that a droop in one core does not affect the voltage level in a neighboring core. Although previous work [16] has suggested signature sharing across cores, this paper provides the first communication mechanism to support it. A full analysis of this mechanism for signature sharing is provided.

[1]http://sourceforge.net/projects/sesc/

TABLE I
Experimental Setup

Simulator	SESC
Technology	90nm
Num Proc.	8, 16
Frequency	2GHz
Benchmarks	SPLASH2
Processor configuration	
I-cache	64KB, 4-way
D-cache	64KB, 8-way
Branch predictor	Hybrid
Branch target buffer	4K entry, 16 way
Instruction queue	16 entries
Retirement order buffer	176 entries
Load/store buffer	56/56 entries
L2 cache	1 MB, 8-way, 10 cycles

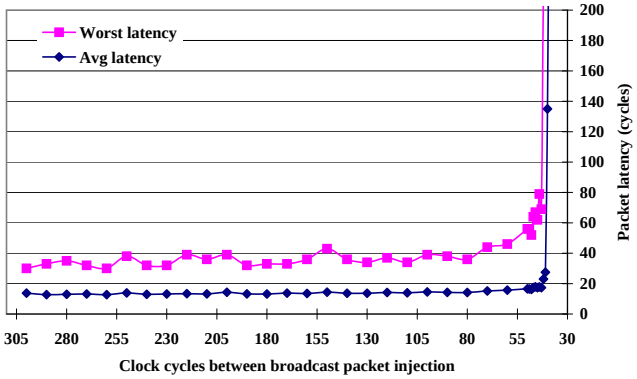

Fig. 3. Latency for broadcast packets in a 16-core multi-core for a thermal packet injection rate of 1 packet per 100 clock cycles. Broadcast packet injection rate is varied along the horizontal axis.

VI. Experimental Results

In an initial experiment, the latency of data transfer for broadcast packets using the SDR interconnect is considered if temperature sensor data are simultaneously transferred to the sensor controller (SC) using the same interconnect. Inter-core transfer of regular application data is made using a different interconnect. As mentioned in Section III, the SC is a small controller or core which is dedicated to process sensor data. In the experiment, temperature sensor data is inserted into the lightweight NoC by each core once every 100 clock cycles. Figure 3 illustrates the average and worst-case latency of broadcast data to all 16 cores for different injection rates of broadcast data. The number of cycles between the injections of 17-byte broadcast packets (including header with addressing information and signature payload; format in Section IV) for each core is shown on the horizontal axis. The smaller cycles per injection values (right side of the graph) indicate more frequent broadcast data injections. The graph shows that as long as per-core broadcast packet injections take place at a pace of one per 50 cycles or less, the average and worst case latency of broadcast packet traffic is largely unaffected. For an 8-core system, the knee of the curve is at about 30 cycles per injection.

TABLE II
SIGNATURE COUNT COMPARISON BETWEEN LOCAL SIGNATURE ONLY AND THE GLOBAL SIGNATURE SHARING FOR SPLASH2 BENCHMARKS. GLOBAL SHARING IS SUPPORTED BY SIGNATURE BROADCAST TO ALL CORES.

		8 core		16 core	
Test bench	Test case	Sign. count	% Sign. reduct.	Sign. count	% Sign. reduct.
Water-spatial	Local	18,470	-	31,038	-
	Global	5,163	62	5,028	84
FFT	Local	86	-	0	-
	Global	86	0	0	0
LU	Local	41,173	-	69,989	-
	Global	15,995	62	16,341	77
Ocean	Local	271,180	-	437,498	-
	Global	224,152	17	346,079	21

TABLE III
PERFORMANCE COMPARISON OF OUR GLOBAL SIGNATURE SHARING METHOD VERSUS THE LOCAL PREDICTION METHOD WHICH USES ONLY LOCAL SIGNATURES

		8 core		16 core	
Test bench	Test case	Time (ms)	Exec time reduct (%)	Time (ms)	Exec time reduct (%)
Water-spatial	Local	14.59	-	7.48	-
	Global	14.23	2.44	7.13	4.63
FFT	Local	16.15	-	10.87	-
	Global	16.15	0	10.87	0
LU	Local	17.20	-	8.01	-
	Global	16.43	4.48	7.32	8.62
Ocean	Local	25.90	-	15.58	-
	Global	24.46	5.57	14.31	8.18

TABLE IV
PERCENTAGE OF GLOBAL SIGNATURE MATCH LATENCIES WHICH FALL INTO SPECIFIC CYCLE COUNT RANGES. ON AVERAGE, LESS THAN 3% OF THE GLOBAL SIGNATURE MATCHES HAPPEN WITHIN 20 CYCLES OF INITIAL SIGNATURE DETECTION IN OTHER CORES

Global signature match latency (cycles)	Percentage (%) 8 core	16 core
< 20	2.49	1.84
20-100	28.24	19.92
100-1000	14.83	19.92
> 1000	54.44	60.40

In a system-level set of experiments, an 8-core and a 16-core system were simulated using SESC to assess the benefit for broadcasting voltage droop signatures across cores. The total number of signatures generated by voltage droops for the global (sharing) approach using broadcasting is compared to the total when only local signatures generated within a core are used to predict droops. Experiments with SESC using the four benchmarks in Table II indicate that in the worst case (*Ocean* for 16 cores) approximately one new signature is produced every 279 clock cycles. This value indicates an injection rate which falls in the constant region at the left of Fig. 3.

The system-level results demonstrate the importance of supporting sensor data broadcasting. As seen in Table II, signature global sharing allows for fewer signature generations and, as a result, fewer rollbacks. For the Water-spatial, LU and Ocean benchmarks, the total number of generated signatures is reduced significantly as signatures are shared across cores. The total number of generated signatures is reduced by more than 60% for the Water-spatial and LU benchmark in both the 8-core and 16-core systems. However, the FFT benchmark has very few voltage droop emergencies and therefore a very small number of signatures (86 signatures in the 8-core system and none in the 16-core system).

The reduced number of generated signatures leads directly to a performance benefit if a penalty of 500 cycles per rollback is assumed. The signature broadcast latency using the sensor data infrastructure has been included in the experiment. To evaluate the benefit of global signature sharing, the performance of four SPLASH2 benchmarks is considered. Table III shows the performance benefit for the global sharing of signatures using broadcasting versus local signature generation only. We can see from this table that global sharing provides a benefit for three benchmarks (Water-spatial, LU and Ocean) and virtually no benefit for one benchmark (FFT). The average performance benefit is 3.13% and 5.36% in the 8- and the 16-core system, respectively. The results of false positives are considered in this analysis.

Since the results in Tables II and III include the NoC interconnect latency due to broadcasting, the effect of this latency versus an ideal, zero-latency broadcast of signatures can be considered. Popnet simulation results in Fig. 3 show that in a multi-core system with the broadcast-enabled interconnect infrastructure, a newly detected signature in one core appears in the signature tables of other cores within 20 cycles of its detection, on average. We define global signature match latency as the cycle count between the detection of a signature in a core and the first time it is used in another core. As an example, consider a signature detected in core 1 during a core 1 voltage droop and then used for voltage droop emergency prediction in core 2. If the signature broadcast latency is greater than the global signature match latency, this signature will not be available in core 2's signature storage before it is needed. The signature is then detected in core 2 as a new signature in response to a voltage emergency and rollback.

The SESC simulator was used to record typical values for global signature match latencies so they could be compared to signature broadcast latency. The results for 8- and 16-core systems are shown in Table IV. This table shows the fraction of global signature match latencies in specific cycle count ranges (averaged over all four benchmarks). The table indicates that the majority of global signature match latencies are more than 20 clock cycles (over 97% on average). This result makes our broadcast-enabled sensor data interconnect infrastructure attractive since it successfully shares available signatures across all cores before the first time they are used in a different core. The performance of 8- and 16-core systems using our sensor data interconnect infrastructure is contrasted to that using an ideal (zero-latency) broadcast interconnect and the results are shown in Table V. In the ideal, zero-latency interconnect, signatures initially detected in one core can be immediately used by other cores in the system.

As shown in Table V, the system performance using our infrastructure is close to the performance using a zero-latency

TABLE V
PERFORMANCE COMPARISON OF GLOBAL SIGNATURE SHARING USING THE ENHANCED INFRASTRUCTURE VERSUS AN IDEAL, ZERO-LATENCY INTERCONNECT

		8 core		16 core	
Test bench	Test case	Time (ms)	Exec time reduct (%)	Time (ms)	Exec time reduct (%)
Water-spatial	Ideal	14.23	-	7.13	-
	Global	14.23	0.02	7.13	0.02
FFT	Ideal	16.15	-	10.87	-
	Global	16.15	0	10.87	0
LU	Ideal	16.42	-	7.32	-
	Global	16.43	0.10	7.32	0.04
Ocean	Ideal	24.15	-	14.08	-
	Global	24.46	1.30	14.31	1.59

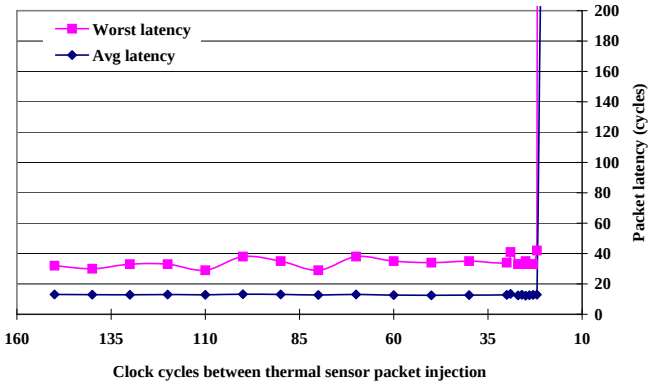

Fig. 4. Latency for temperature sensor packets in a 16-core multi-core for a broadcast packet injection rate of 1 packet per 279 clock cycles. Thermal packet injection rate is varied along the horizontal axis.

interconnect. The average performance penalty is less than 0.4% and the maximum performance penalty is less than 1.6% for the *Ocean* benchmark. This result is understandable since the low latency signature sharing achieved by our infrastructure ensures that the majority (>97%) of signatures can be broadcast to all cores in the system before they are used for the first time.

For a final experiment, we consider the capability of light-weight sensor interconnect to handle increased core-to-sensor controller traffic beyond 1 injection every 100 cycles if the interconnect is also used to broadcast signature data. In Fig. 4, the latency of temperature sensor packets is illustrated for a range of injection rates if broadcast signatures are injected by each core at a fixed rate of 1 every 279 cycles. The results show that the temperature sensor data (or data from other sensors) could be transferred to the sensor controller using an injection rate of up to about 1 every 30 clock cycles for a 16-core system. This value drops to about 1 every 10 clock cycles for an 8-core system.

VII. CONCLUSIONS

The use of sensor information is vital for multi-core processors used in embedded systems. In this paper, an on-chip sensing system for embedded multi-cores which includes broadcast capabilities is introduced. The amount of on-chip resources required for this system is quite small (<3% of chip area). This resource eliminates the need for sensor data to be transferred and processed via the multi-core's standard communication and compute infrastructure. As a way of enhancing embedded multi-core dependability, we demonstrate the importance of supporting broadcasting through the distribution of voltage droop signatures. Using interconnect and multi-core simulators it is shown that a 5.4% performance improvement can be achieved on average (9% best case) by broadcasting signatures which allow cores to avoid voltage emergencies and costly processor rollbacks[2].

REFERENCES

[1] L. Guang, E. Nigussie, J. Isoaho, P. Rantala, and H. Tenhunen, "Interconnection alternatives for hierarchical monitoring communication in parallel SoCs," *Microprocessors and Microsystems*, vol. 34, pp. 118–128, Jan. 2010.

[2] P. Ituero, M. Lopez-Vallejo, M. A. S. Marcos, and C. G. Osuna, "Light-weight on-chip monitoring network for dynamic adaptation and calibration," *IEEE Sensors Journal*, vol. 12, no. 6, pp. 1736–1745, June 2012.

[3] J. Zhao, S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, "A dedicated monitoring infrastructure for multicore processors," *IEEE Trans. on VLSI Systems*, vol. 19, no. 6, pp. 1011–1022, June 2011.

[4] M. Floyd, B. Brock, M. Ware, K. Rajamani, A. Drake, C. Lefurgy, and L. Pesantez, "Adaptive energy management features of the POWER7 processor," in *Proc., HotChips*, Aug. 2010.

[5] (2012) Intel Active Management Technology. [Online]. Available: http://www.intel.com/technology/platform-technology/intelamt/

[6] V. Reddi, M. Gupta, G. Holloway, M. Smith, G. Wei, , and D. Brooks, "Voltage emergency prediction: A signature-based approach to reducing voltage emergencies," in *Proc., Int'l Symp. on High Performance Comp. Arch.*, Feb. 2009, pp. 18–27.

[7] A. Bouajila, A. Lakhtel, J. Zeppenfeld, W. Stechele, and A. Herkersdorf, "A low-overhead monitoring ring interconnect for MPSoC parameter optimization," in *Proc., Int'l Symposium on Design and Diagnostics of Electronic Circuits and Systems*, Apr. 2012, pp. 46–49.

[8] M. Fattah, "Exploration of MPSoC monitoring and management systems," in *Proc., Int'l Workshop on Reconfigurable Communication-Centric Systems-on-Chip*, June 2011, pp. 1–3.

[9] B. Phanibhushana, P. Vijayakumar, P. Shabadi, G. Prabhu, and S. Kundu, "Towards efficient on-chip sensor interconnect architecture for multi-core processors," in *Proc., Int'l SoC Design Conf.*, Nov. 2010, pp. 307–310.

[10] Y. Peng, M. Saldana, and P. Chow, "Hardware support for broadcast and reduce in MPSoC," in *Proc., Int'l Conf. on Field Programmable Logic and Applicatons*, Sept. 2011, pp. 144–150.

[11] M. Moadeli and W. Vanderbauwhede, "A communication model of broadcast in wormhole-routed networks on-chip," in *Proc., Int'l Conf. on Advanced Information Networking and Applications*, May 2009, pp. 315–322.

[12] L. Fiorin, G. Palermo, and C. Silvano, "A security monitoring service for NoCs," in *Proc., Int'l Conf. on Hardware/Software Codesign and System Synthesis*, Oct. 2008, pp. 197–202.

[13] B. Gieseke, et al., "A 600 MHz superscalar RISC microprocessor with out-of-order execution," in *Proc., Int'l Solid State Circuits Conf.*, Feb. 1997, pp. 176–177.

[14] L. Shang, L. Peh, and N. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proc., IEEE Int'l Symposium on High-Performance Computer Architecture*, Feb. 2003, pp. 91–102.

[15] K. Hazelwood and D. Brooks, "Eliminating voltage emergencies via microarchitectural voltage control feedback and dynamic optimization," in *Proc., Int'l Symposium on Low-Power Electronics and Design*, Aug. 2004, pp. 326–331.

[16] J. Zhao, B. Datta, W. Burleson, and R. Tessier, "Thermal-aware voltage droop compensation for multi-core architectures," in *Proc., ACM Great Lakes Symposium on VLSI*, May 2010, pp. 335–340.

[2]This research was supported by the Semiconductor Research Corporation under Task 2083.001