A Monitor Interconnect and Support Subsystem for Multicore Processors

Sailaja Madduri, Ramakrishna Vadlamani, Wayne Burleson and Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA, United States

Abstract*— In many current SoCs, the architectural interface to on-chip monitors is ad hoc and inefficient. In this paper, a new architectural approach which advocates the use of a separate low-overhead subsystem for monitors is described. A key aspect of this approach is an on-chip interconnect specifically designed for monitor data with different priority levels. The efficiency of our monitor interconnect is assessed for a multicore system using both an interconnect and a system-level simulator. Collected monitor information is used by a dedicated processor to control the frequency and voltage of individual multicore processors. Experimental results show that the new low-overhead subsystem facilitates employment of thermal and delay-aware dynamic voltage and frequency scaling.

I. INTRODUCTION

Contemporary single-chip systems typically exhibit stringent processing, communication, and power constraints that must be carefully addressed during system design. Performance issues are particularly acute for components which must provide real-time service and high throughput in the presence of uncertainties such as temperature, workload, wear-out, and supply noise. Recent high-end processors from Intel (Montecito), AMD (Opteron) and IBM (Cell) use extensive on-chip monitors for run-time estimates of temperature, power, clock jitter, supply noise and performance. However, a unified approach to the interconnection of monitors and use of monitor information has not yet been developed.

Specific uses of monitors in terms of system critical soft-error failures, wear-out detection and security issues require fast connections on a global scale. These connections can be supported by a dedicated interconnect that is coupled to the main multicore architecture. In order to maximize monitor effectiveness, monitor data often needs to be collated from across the chip and evaluated in real time as an SoC operates. This data can then be used to alter SoC operation in response to environmental conditions.

We view the integration of monitors and the collection and processing of monitor information as an important unaddressed SoC design issue. As an initial step in the development of a complete monitor subsystem for SoCs, a low-overhead on-chip interconnect, which is optimized for monitors, has been designed. Although simplified compared to other on-chip interconnect approaches, our new interconnect technique supports irregular routing topologies, priority based data transfer and customized monitor interfacing. Collected monitor data values are manipulated by one or more processors and the results are used to control SoC run-time operation.

The overhead and performance of the monitor network-on-chip (MNoC) interconnect for an eight core multiprocessor has been measured via hardware synthesis, simulation, and multicore architectural simulation. For an eight core system, the area and power overhead for the interconnection of 192 thermal monitors is found to be less than 0.5%. Architectural simulations show that multicore performance can be significantly improved when MNoC-collected thermal and delay data is used to perform dynamic frequency and voltage scaling.

II. MONITORS AND RELATED INTERCONNECTS

Several recent chips have explored the benefits of monitoring based control. In a specific example of monitor data use, 90-nm Itanium processors use a series of voltage and thermal sensors in conjunction with a controller. This Foxton technology [1] allows for dynamic voltage and frequency scaling based on sampled monitor data. A similar approach for a Hitachi multiprocessor [2] uses thermal and performance information to control voltage and bandwidth allocation. All of these systems assume small numbers of cores and monitors connected in an ad hoc fashion.

A relatively small number of SoC projects have examined the integration of multiple sensors and associated control onto a single SoC substrate. Velusamy et al. [3] describe the interconnection of an array of thermal monitors to a PowerPC with a CoreConnect on-board peripheral bus. Monitor information is then used to control system clock frequency. Although effective, this bus-based approach is not scalable beyond a small number of cores [4] and uses far more resources than necessary to implement communication and control. The IBM Power6 architecture [5] interconnects multiple sensors and actuators via a high-speed serial bus. The described interconnect primarily serves as an external interface to voltage and thermal control via an I2C bus for a modest number of cores. Our approach also builds on ideas previously used for SoC debug and test, such as JTAG boundary scan, however, debug subsystems do not use collected information to influence SoC run-time operation.

Numerous network-on-chip architectures [6] have been proposed for SoCs over the past decade. These interconnects generally require a series of router circuits organized in a mesh-like topology. In contrast to MNoC, most NoC routers are optimized for routing bandwidth and consume considerable chip resources. Often, individual NoC routers require tens of thousands of transistors [6], include datapath widths of 32 to 256 bits, and buffer tens to hundreds of data values. In contrast, our approach attempts to minimize resource count to exactly the bandwidth and buffering required for SoC monitoring.

III. Monitor Network On Chip

Our monitoring subsystem augments conventional system-on-a-chip hardware with additional components for monitoring, verification, and response. Multiple monitors are added to each major component of the SoC. The monitors are linked by a monitor network on-chip (MNoC), a heterogeneous communication substrate, as seen in Fig. 1. In general, the spread among the required bandwidths of different monitors is large. Hence, MNoC supports low-overhead routers and localized connections like buses and multiplexers. High bandwidth monitors are directly connected to routers, while the lower bandwidth monitors are connected via multiplexers or a bus that connects to the network as shown in the Fig 1. The MNoC is interfaced to a monitor executive processor (MEP). The MEP provides a software layer to implement new collaborative monitoring algorithms. MNoC has been designed to incur minimal area and energy overhead compared to a general purpose on- chip interconnect by optimizing its width, access control, arbitration, flexibility, and bandwidth to the monitor data collection task. Specific interconnect challenges include the development of monitor-network and network-MEP interfaces to accommodate different monitor types and the development of interconnection components for irregular topologies and mixed-priority traffic.

On-chip monitors are typically distributed in an unorganized fashion, necessitating an irregular interconnect topology. An irregular mesh topology of routers is needed for MNoC, whose placement is dictated by the distribution of monitors. Two types of monitors are supported by MNoC: (1) *data pull* monitors that put data onto the network at regular intervals and (2) *data push* monitors that report data occasionally. For example, thermal monitors that report temperature periodically can be classified as data pull, while error monitors that report data only in the event of an error are data push. For data pull monitors, data requests are forwarded to the monitors by the associated router interfaces. Interrupts are used to support unexpected events detected at data push monitors. MNoC traffic is entirely monitor data that is communicated to the MEP and no monitor-monitor communication is required. Monitor data in the network is classified into two different priority levels. Messages to the MEP from data push monitors are usually critical in nature and are hence tagged with a higher priority. Messages from data pull monitors are regular priority unless there is an emergency event at the monitor. High priority data is routed through the network using dedicated resources in the routers.

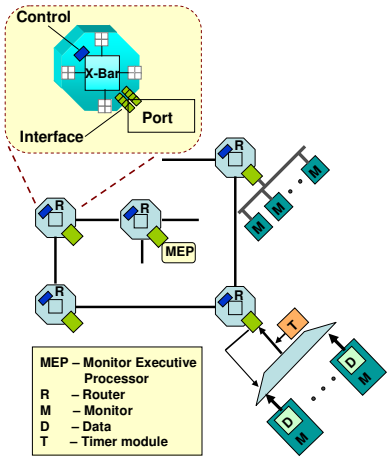

Figure 1: Detailed view of MNOC for multiple cores

Monitor information is transported on the network as packets of data. A network interface appends monitor information with routing information and converts each packet into flits. The packetization module also appends the source monitor's address which is required by the MEP to identify the origin of the monitor data. A priority bit is also included in the packet to enable the routers to differentiate critical data from the regular ones. MNoC flit width is chosen to be the same as the width of the physical channel. MNoC implements wormhole switching which ensures low latency while consuming a minimal amount of buffer space.

The most commonly used adaptive routing protocols involve expensive router implementations [4] and are suitable for very high and unpredictable traffic rates. Instead, for low-overhead MNoC, we use a static distributed routing protocol which involves the use of routing tables at individual routers. Each routing table is a lookup table that can be indexed using the destination address. For every possible destination, the table contains information about the output port that the packet needs to be routed through. The irregular placement of monitors results in an irregular mesh topology leading to concerns regarding deadlock. A fault tolerant mesh routing algorithm [7] is used to generate deadlock free paths that are stored in the routing tables. Since no monitor-to-monitor communication is assumed, the overhead incurred with routing tables is minimal. This non-adaptive routing protocol allows for a very lightweight router implementation because the overhead for adaptive route evaluation is eliminated. Errors in MNoC transmission are handled through a combination of per-packet CRC values and MEP-based requests for retransmission.

Monitors in the system can either have dedicated interfaces to network routers or can interface to the routers through shared buses or multiplexers (Fig. 1). The interfaces need to be generic and should allow for the interfacing of any kind of monitor to the network. The control logic supports both data push and data pull monitors. Also, synchronization issues that result out of different monitor and network frequencies need to be addressed. In our architecture, the monitors and the network router connect through a master-slave interface, the router end being the master and the monitor, a slave. The architecture of the monitor-network interface is shown in Fig 2.

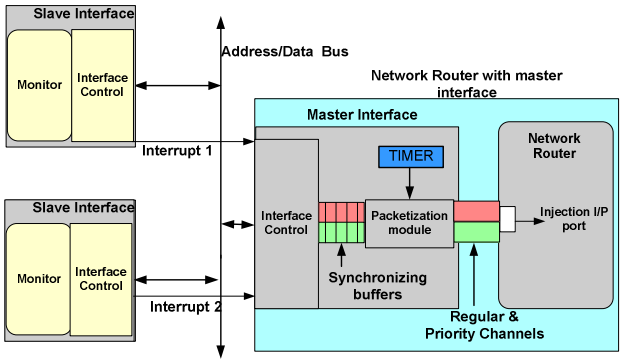

Figure 2: Monitor –bus – network interface

The interface control logic is built to read data at a pre-determined rate from the connected monitors i.e. there is a control state machine at the router interface that generates read addresses for each of the connected data pull monitors according to a pre-set schedule. Any data push type of monitor connected at the interface has a dedicated interrupt line connected to the router interface and has a capability to generate an interrupt indicating that it needs to be read. In the event of an interrupt, the controller breaks away from the original sequence to generate a read address for the interrupting monitor. Any data read from an interrupting monitor is tagged as high priority data. Once the monitor data is read, the controller appends it with information about the originating monitor and priority value. The data is then written into the synchronizing FIFO which is read by the packetization module. The packetization module converts the data into flits, forwards them to the appropriate channel in the network (regular or priority).

The packetization module also appends monitor data with a time stamp from an embedded timer which identifies the time at which data was sampled. The maximum value of the timer is chosen such that any packet injected in the network reaches the MEP before the timer resets twice. This ensures that the MEP accurately identifies the time frame in which the data was sampled. For example, if a monitor generated a temperature value of 20 degrees at time t = 1ms and the data is received at the MEP at time t = 1.5ms, the MEP interprets the current temperature value to be 20.03 degrees using an average temperature gradient of 0.06 deg/ms [1]. A single timer is shared across several interfaces.

The MEP and the network router connect through a master-slave interface, the MEP being the master and the router, a slave. Monitor data received from either of the channels in the router is read by a de-packetization module at the network router-MEP interface. A synchronizing FIFO contains separate queues for regular and priority data. The MEP software reads information from the FIFOs at regular intervals with consideration given to priority data. The FIFO addresses synchronization issues and is sufficiently sized to ensure that no data is dropped. Once data is received, the MEP uses the source information to determine the type and location of the monitor that sent out the data and takes necessary action by affecting system parameters.

The low bandwidth required by most monitors is exploited to minimize MNoC router area. Unlike typical NoC routers, MNoC routers provide sufficient bandwidth and latency with small eight bit data widths and minimal (e.g. 4) buffer sizes. Each router is further optimized by removing unused data ports as a result of the irregular mesh topology. The MNoC router is built to be highly parameterizable. The optimal buffer sizes and widths can be determined based on the required latency and bandwidth for different monitoring systems. The choice of these parameters is ultimately a trade-off between performance (in terms of bandwidth and latency) and overhead (in terms of area and power).

For MNoC, input buffering is used instead of output buffering because of the low overhead that input buffering offers [8]. Head-of-line blocking, a possible drawback of input buffering, is insignificant in the case of MNoC because most MNoC traffic is directed towards the MEP.

Every input channel in the router is multiplexed into two separate virtual channels, a priority channel and the regular channel. The priority channel is used to exclusively transfer critical monitor data. A packet that is injected into a network with a high priority (priority field in the packet header is set to 1) enters the priority channel and travels in the same channel until it reaches the destination. This channel is reserved for critical data and is not used for regular data transfer. Packets remain in the channel determined at packet injection.

MNoC employs a credit based flow control to regulate data traffic and to avoid packet dropping. To facilitate this, every router has buffer slot counters that keep track of the number of empty buffer slots in the regular and the priority channels on the adjacent routers. The availability of a buffer space is communicated by adjacent routers using credit messages. Flits that enter the MNoC router are buffered in the appropriate input channel and subsequently go through three router pipeline stages before reaching the next hop: routing table look up, switch arbitration, and switch traversal. Once switch access is granted by switch arbitration, the flit goes through the final pipeline stage where it traverses the crossbar and enters the same channel (regular or priority) in the next router. The priority channel is given preference in the entire switch arbitration stage to ensure lowest possible latency on that channel. Among requests from the regular channel, the arbiter grants access in a random fashion.

IV. Experimental Approach

In order to validate the MNoC approach and evaluate trade-offs for various design constraints such as area, bandwidth and latency, a series of synthesis and simulation experiments have been performed. The Popnet interconnect simulator [9] has been significantly modified to estimate bandwidth and latency values for the heterogeneous MNoC interconnect. The router pipeline and the routing protocol were modified along with additional support for expanded interfaces. The simulator, in modified form, allows for a complete evaluation of various MNoC topologies and components.

To estimate the overhead of our MNoC approach, we developed a synthesizable hardware model of the MNoC router and MEP. The MNoC hardware model is parameterizable and

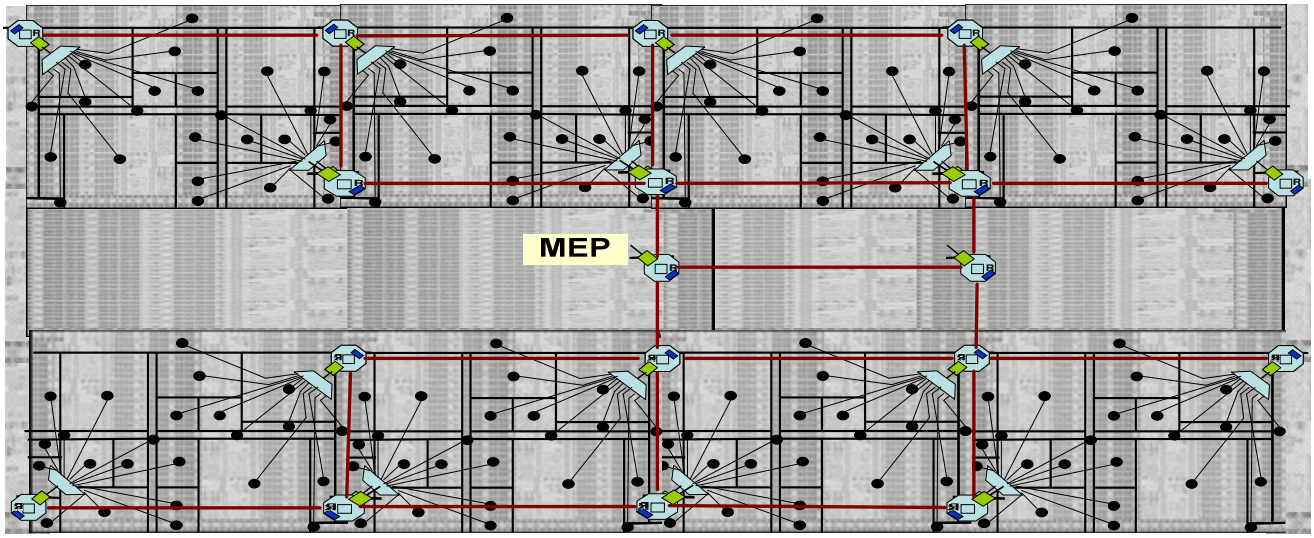

Figure 3: Monitor network on chip layout for thermal monitors on a 8 core processor

allows for evaluation of area for different router widths and buffer sizes. The hardware model, which operates at 500 MHz, was synthesized using Synopsys Design Compiler using a 90nm standard cell library [10]. Architectural simulations were performed using the SESC architectural simulator [11] to quantify the benefits of employing our monitor subsystem at a system level.

In an initial experiment, 24 thermal monitors on each of the 8 processor cores report temperature values from various locations on the chip. The floorplan of each processor core used here for thermal modeling is based on the AMD Athlon 64 processor [12]. The layout of the eight core system is shown

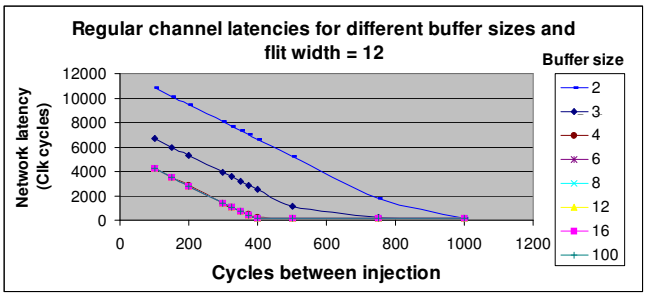

Figure 4: Regular channel latencies for different buffer sizes

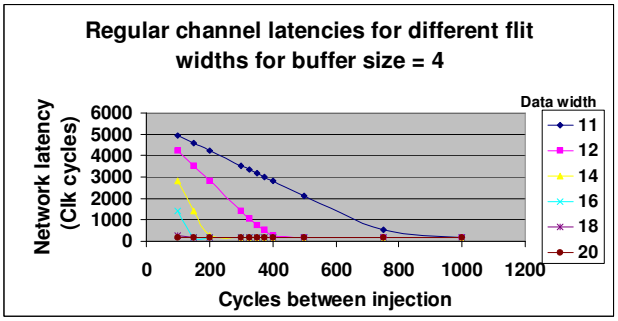

Figure 5: Regular channel latencies for different flit widths

in Fig 3. There are two MNoC routers per core, each of which collects thermal data from 12 thermal monitors using a multiplexer. Thus, 192 thermal monitors from eight cores connect to 16 routers through 16 multiplexers. The MEP is attached to a dedicated router (Fig. 3) at a location central to the routers. The resultant topology is an irregular mesh. A dummy router adjacent to the MEP was added to facilitate routing. With this 18 router setup, deadlock-free routing [7] was used to generate paths from the routers to the MEP. A multiplexer interface was used to make connections between the monitors and the routers. Thermal monitors for DFS [13] can be classified as low bandwidth data pull monitors. Our interconnect simulator was used to evaluate the latency of this network for different network parameters.

Fig 4 shows a plot of network latency versus injection rates for various buffer sizes for regular (non-priority) traffic. A total of 95% of total traffic is assumed to be regular traffic. The value on the X axis, cycles between injections, indicates the number of clock cycles between two sampling points for the thermal monitors. Network latency (the Y axis) indicates the average time required (in clock cycles) for data to travel from a monitor to the MEP. Fig 4 indicates a significant dependence for the regular channel on the input buffer size for sizes less than 4. For buffer sizes greater than and equal to 4, limited latency reduction is achieved by increasing buffer size. For longer delays between injections, the regular channel latency becomes insensitive to buffer sizes. We simulated 5% of the total traffic to be priority traffic to assess the latency on the priority channel. For all cycles per injection rates, network latencies between 16 and 21 cycles were found. The result indicates that the latency on this channel is more or less constant and is ideally suited for low latency critical data transfer. There is practically no impact of buffer sizes on the latency. Fig 5 shows a plot of network latency versus injection rate, for different flit widths. For higher sampling rates, the flit width that gives ideal latency increases with increasing cycles between injections.

Overall, it can be inferred from the results that for higher cycles between injection (lower sampling rates and hence lower bandwidths), the latency values are mostly insensitive to network parameters like buffer size and flit width. At such low sampling rates, close to ideal network latency can be achieved with minimal network resources.

Monitors with higher sampling rates have latencies that are highly network dependent. These monitors usually dictate the choice of network parameters. Table 1 shows area results for the 18 router thermal MNoC system estimated at a 90nm technology node.

In a second experiment, we demonstrate a monitor subsystem that satisfies system design constraints while providing a performance benefit. We use SESC to simulate eight processors and one central MEP, as shown in Fig 3. The eight cores each have a private L1 and L2 cache. In comparison to a commercial 8 core processor [14], the area overhead of MNoC in this configuration is 0.819/378 mm^2 = 0.21%.The power model that is used by SESC for processors is based on Wattch. The cache power model is based on CACTI and the temperature model for both (called SESCSpot) is based on HotSpot [13].

SESCSpot calculates the temperature of processor sub-blocks based on the power trace of the architecture in a post processing fashion. For the DFS implementation we integrated SESCSpot into the core of the SESC simulator to obtain the temperature readings dynamically. This enabled the MEP to sample the temperature readings at a pre-determined interval and execute the DFS algorithm. In this experiment, the 192 thermal monitors on the 8 core chip were sampled every 2ms to provide a resolution of 0.1 degC. This number was determined assuming a maximum temporal temperature gradient of 60degC/sec [1]. To meet this sampling requirement, an MNoC configuration with flit size of 12 bits and an input buffer size of 4 was used. The resulting MNoC area and power, as obtained from Table 1, are 0.819 mm^2 and 244 mW, respectively. The temperature reported by the monitors is collected by MNoC and transported to the MEP which uses the data for dynamic frequency scaling.

Dynamic frequency scaling of a processor system improves system performance by operating cores within power dissipation and temperature limits. Two trials were performed on the 8 core system to demonstrate the benefits of DFS on a benchmark application. A floating point Whetstone benchmark [15] is used to conduct the experiments for a total of 2 billion instructions.

In one scenario, the system was operated at a constant frequency of 1GHz to meet pre-defined power and temperature limits and the run time consumed was noted. In this case, since the predefined temperature threshold is not exceeded, it was not necessary to employ MNoC. In a second scenario, MNoC is employed to transport monitor data which is used by a MEP to perform DFS. In this case, the operating frequency of the system is toggled between 2 GHz and a lower frequency to ensure that the specified power and temperature limits are not violated. The run time was again noted and the resulting performance improvement of 18% for an 8 core system was calculated. To evaluate how the performance benefit using

TABLE 1: MNOC AREA RESULTS

Flit width	Buffer size	Total MNoC area at 90 nm (mm2)	Flit width	Buffer size	Total MNoC area at 90 nm (mm2)
12	2	0.700	12	8	1.084
14	2	0.765	14	8	1.201
16	2	0.825	16	8	1.314
18	2	0.890	18	8	1.420
20	2	0.950	20	8	1.530
12	4	0.819	12	16	1.571
14	4	0.894	14	16	1.751
16	4	0.970	16	16	1.919
18	4	1.043	18	16	2.094
20	4	1.116	20	16	2.262

TABLE 2: RUNTIMES FOR MNOC AND NON-MNOC CASES

Cores	Runtime for Freq = 1 GHz (sec)	Runtime for Freq = 2 GHz (sec)	Performance benefit due to MNoC
4	3.36	2.42	28%
8	2.75	2.25	18%
12	2.27	1.52	33%
16	1.75	1.35	23%

MNoC scales with the number of cores, experiments were also performed for 4, 12 and 16 core systems. The advantage of employing MNoC is consistent as the number of cores is increased, as shown in Table 2.

In a third experiment, the system-level benefits of our monitor subsystem on delay-based voltage control were determined. Real-time delay monitoring (using critical path delay monitors) and control techniques were used to offset voltage droops at system run time. The monitoring setup involves 8 delay monitors per core [5] which report 12 bits of delay data. Monitor data is transported to the MEP through a 9 router MNoC. The delay monitors require high network bandwidth since voltage values can change quickly. In response to a voltage droop event, the MEP increases the voltage of the core to avoid a low voltage condition.

In contrast, in a non-MNoC system the supply voltage is constant and is set conservatively to a value that accounts for the maximum voltage droop. The experiment was conducted for 4, 8 and 16 processor cores (9 routers in all cases). Fig 6 shows the percentage of power savings that MNoC provides in comparison to a non-MNoC system. The cycles between injections, on the X axis, indicates the number of clock cycles between two sampling points for the delay monitors. As seen from the results, all three configurations result in power savings versus the non MNoC case for specific values of

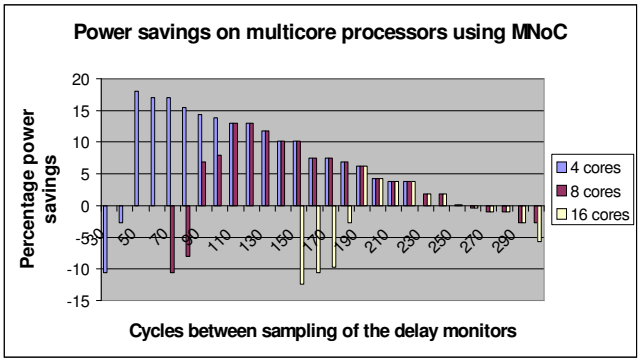

Figure 6: Power savings in multi-core processors using MNoC

sampling rates. As the number of cores increases, the number of monitors increases, requiring more bandwidth from the network. This trend motivates the need for a scalable medium like MNoC versus buses or serial links. Fig 6 shows that certain sampling intervals yield a negative result. In these cases, the sampling or the network delays are so high that the system gains no benefit from run time monitoring. These combinations of sampling intervals and MNoC delays can be determined during system design.

Finally, using the above delay monitoring setup, trials were conducted to assess how the latency of MNoC scales as the number of cores in the system increases. As seen in Fig 7, MNoC delay for 128 cores at a given bandwidth is much

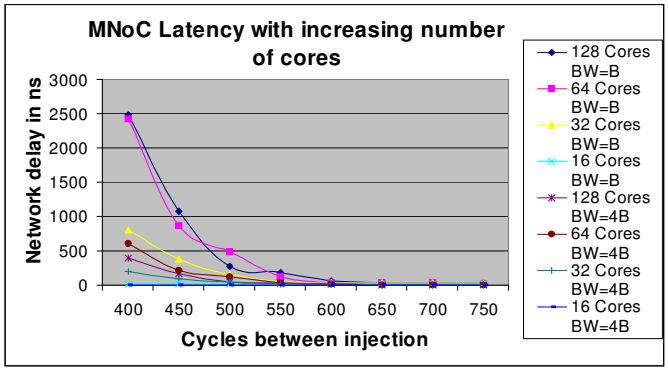

Figure 7: MNoC performance with increasing number of cores

higher than the delay for the 32 core configuration. But the delay values of the 128 core system with 4 times the network bandwidth are comparable to those of the 32 core configuration. This indicates that the network can be scaled to larger number of cores by scaling the network bandwidth, retaining similar network latencies.

V. CONCLUSION

This work presents a scalable and lightweight approach for monitor data collection and processing. System level performance benefits are obtained by using this monitor data to scale processor frequency and voltage values. Experiments show that the interconnect can be sized on a per-application basis to obtain substantial performance benefits. An area overhead of 0.21% was achieved for the monitor interconnect when applied to an eight core system. In the future, we plan to evaluate the collaborative use of data from multiple monitors in controlling multicore behavior. Automating MNoC creation is also a promising area that needs to be addressed.

VI. ACKNOWLEDGEMENTS

This work was funded by Semiconductor Research Corporation under Task 1595.001. The authors would like to acknowledge the suggestions of our SRC liaisons at Intel and AMD.

REFERENCES

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

[2] M. Saen, K. Osada, S. Misaka, T. Yamada, Y. Tsujimoto, Y. Kondoh, T. Kamei, Y. Yoshida, E. Nagahama, Y. Nitta, T. Ito, T. Kameyama, N. Irie, " Embedded SoC Resource Manager to Control Temperature and Data Bandwidth, " IEEE International Solid-State Circuits Conference, pp. 296-604, Feb. 2007.

[3] S. Velusamy, W. Huang, J. Lach, M. R. Stan, K. Skadron, "Monitoring Temperature in FPGA based SoCs," IEEE International Conference on Computer Design, pp. 634-637, Oct. 2005.

[4] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," ACM Computing Surveys, vol. 38, no.1, Mar. 2006

[5] M. S. Floyd, S. Ghiasi, T.W Keller, K. Rajamani, F.L. Rawson, J. C. Rubio, M. S. Ware, "System Power Management Support in the IBM Power6 Microprocessor," IBM Journal of Research and Development, vol. 51, pp. 733-746, Nov 2007

[6] F. Moraes, N. Calazans, A. Mello, L. Möller, L. Ost, "HERMES: An infrastructure for low area overhead packet-switching networks on chip," Integration: The VLSI Journal, pp. 69-93, Oct. 2004

[7] K. H. Chen and G.-M. Chiu, "Fault-Tolerant Routing Algorithm for Meshes without Using Virtual Channels," J. Information Science and Eng., vol. 14, pp. 765-783, Dec. 1998.

[8] Y. Tamir, G. L. Frazier, "High-performance multiqueue buffers for VLSI communication switches," International Symposium on Computer Architecture, pp.343-354, Jun. 1988

[9] L. Shang; L. Peh; N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," International Symposium on High-Performance Computer Architecture, pp. 91-102, Feb. 2003

[10] UMC's 90nm 1P9M Logic/Mixed Mode Low-K SP-HVT process library, http://www.faraday-tech.com

[11] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, K. Strauss, S. Sarangi, P. Sack, P. Montesinos, "SESC Simulator," Jan. 2005, http://sesc.sourceforge.net.

[12] G. M. Link, N. Vijaykrishnan, "Thermal trends in emerging technologies," International Symposium on Quality Electronic Design, Mar. 2006.

[13] K. Skadron , M. R. Stan , K. Sankaranarayanan , W. Huang , S. Velusamy , D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation, " ACM Transactions on Architecture and Code Optimization, vol. 1 no. 1, pp.94-125, Mar. 2004

[14] A. S. Leon, K. W. Tam, J. L. Shin, D. Weisner, F. Schumacher, "A Power-Efficient High-Throughput 32-Thread SPARC Processor," IEEE Journal of Solid-State Circuits, vol. 42, no. 1, pp.7-16, Jan. 2007

[15] H. J. Curnow, B. A. Wichmann, "A Synthetic Benchmark," Computer Journal, vol 19, pp. 43-49, Feb. 1976.