A Dedicated Monitoring Infrastructure For Multicore Processors

Jia Zhao, Sailaja Madduri, Ramakrishna Vadlamani, Wayne Burleson, *Senior Member IEEE*, and Russell Tessier, *Senior Member IEEE*

***Abstract*— On-chip monitoring of environmental information, such as temperature, voltage, and error data, is becoming increasingly important. To address this need, a low-overhead architectural approach to monitor data collection and use in multicore systems is described. A key aspect of our stand-alone monitoring subsystem is a low-complexity, on-chip network designed to transport monitor data with multiple priority levels. Collected monitor information is evaluated by a dedicated processor. Experimental results using architectural and interconnect simulators show that the new low-overhead subsystem facilitates employment of thermal and delay-aware dynamic voltage and frequency scaling. In contrast to using existing on-chip interconnect resources to communicate monitor data, the new subsystem provides necessary bandwidth for monitor data traffic without impacting application data traffic. Synthesis results show that our dedicated monitoring approach consumes about 0.2% of multicore area and power resources for an 8-core system based on AMD Athlon 64 processor cores.**

***Index Terms*—Network on chip, on-chip monitoring, multicore.**

I. Introduction

Computing in the presence of various sources of uncertainty significantly complicates the design and implementation task. Multicore and manycore systems present a particular challenge as large numbers of processor cores are integrated into single-chip platforms. As multicore deployments become more diverse, a static system operating environment can no longer be assumed. These systems are susceptible to a number of reliability, performance, and power constraints that must be carefully addressed during system operation. As the size of multicore systems increase, the importance of using run-time monitoring information to tune system operation becomes critical. Fault tolerance issues are particularly acute for multicore system design in which system elements, such as caches and memory controllers, are shared by many individual cores. Recent multicore processors from Intel (Montecito), AMD (Opteron) and IBM (Cell) use on-chip monitors for run-time estimates of temperature, power, clock jitter, supply noise and performance for a small number of cores. However, an automated, dedicated approach to the collection and use of monitor data in multicores has not been developed. Multicore monitor information represents a substantial data workload that must be analyzed in its own right, separate from the core processing capabilities of the multicore system. This information can then be used to configure multicore resources in conjunction with system and application software.

System critical monitor information, including soft error failures, wear-out data, and voltage droop often require immediate attention at the system level. As demonstrated in this manuscript, the presence of a fast and dedicated, but minimal, interconnect for monitor information allows for effective, dedicated monitor data transfer. Monitor information from multiple multicore monitors is then assessed in real time at one or more processing components. The results of this processing are then used to affect multicore behavior via operations such as per-core frequency and voltage scaling, among others.

The dedicated collection and processing of system-on-chip (SoC) environmental information from on-chip monitors provides an important multicore architecture design dimension. This research presents an integrated approach to address this issue with the development of a complete monitor subsystem for SoCs, including on-chip monitors, a low-overhead on-chip interconnect, which is optimized for monitors, and one or more monitor data processing components. The interconnect has been designed to provide interfaces to a variety of different monitor types and monitor data processing components, from low-complexity thermal monitors to a microcontroller. Although simplified versus typical on-chip interconnects, the monitor network-on-chip's (MNoC) support for irregular topologies, priority-based data transfer, and dead-lock free routing provide a flexible data collection environment. Following monitor data processing, MNoC is used as an interface to control circuitry (e.g. dynamic voltage and frequency scaling (DVFS) control) which affects multicore operation.

The overhead and performance benefit of our monitoring

Manuscript received August 4, 2009; revised February 2, 2010. This work was supported by the Semiconductor Research Corporation under Task 1595.001.

J. Zhao, W. Burleson and R. Tessier are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA. (e-mail: tessier@ecs.umass.edu).

S. Madduri is with Intel Corporation, Hillsboro, OR 97123 USA.

R. Vadlamani is with Qualcomm Inc., Boxborough, MA 01719 USA.

Digital Object Identifier xxxx.

subsystem have been evaluated via hardware synthesis, interconnect simulation and multicore architectural simulation. Multicore systems of up to 16 cores which include thermal and voltage monitors are considered for DVFS while systems of up to 256 cores are considered for performance analysis. Each core is augmented with MNoC hardware which allows for prioritized monitor data transport and subsequent monitor data processing. Appropriate interconnect bitwidth and buffer sizing are assessed through a series of parameter sweeps with an interconnect simulator under realistic monitor data workloads. The size and power consumption of the MNoC hardware is determined via RTL synthesis using a 90nm technology library. This overhead is determined to be less than 0.2% for a system of 8 cores based on AMD Athlon processors. The system-level benefits of monitor collection and evaluation are examined through a series of application experiments in which decisions are made based on collated temperature and voltage information.

The addition of a dedicated monitor interconnect may seem unnecessary given the available high-speed networks-on-chip that are frequently available in multicore systems. To further validate our approach, direct comparisons between a dedicated monitor interconnect and the use of existing multicore interconnect for monitor data transfer are performed. Experimental results show significant latency limitations when existing interconnect is used to transport monitor information.

The remainder of the manuscript is organized as follows. Section II describes existing SoC monitoring subsystems and related interconnect networks. Section III describes our monitoring subsystem including MNoC, dedicated processor for monitoring applications, and system interfaces. Section IV explores the details of per-core monitor allocation as a technique for justifying differing amounts of monitoring interconnect and monitor data processing. The experimental approach for validating our work appears in Section V. Section VI provides an analysis of experimental results and Section VII concludes the paper.

II. Background and Related Work

A. Monitors and Monitor Subsystems

As SoC design has migrated towards the use of multicores, the deployment and use of on-chip monitors has become more widespread. Monitors are important components in many SoCs. Typical examples include processor performance monitors, thermal monitors, delay monitors, and wearout monitors, among others. Most thermal sensors are based on simple ring oscillators or diode-based circuits [1]. Critical path monitors are used to identify the effects of aging, process variation and supply noise on circuit performance. Typical path delay monitors [2] include multi-inverter delay lines with capture latches at each inverter output. The output of the critical path monitor is often a digital code which can require high bandwidth. Hardware performance monitors measure processor performance by establishing a pattern for a certain interval of execution. Hardware monitoring can collect statistics such as instructions per cycle (IPC), resource utilization, and instruction dependencies that can be used to reconfigure processor resources.

In many cases it is desirable to use information from multiple monitors to validate collected information. For example, 90nm Itanium processors [3] use a series of voltage and thermal sensors in conjunction with a controller to evaluate chip environmental conditions. This Foxton technology [3] allows for dynamic voltage and frequency scaling based on sampled monitor data. A similar approach for a Hitachi multiprocessor [4] 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 interconnected sensors and associated control onto a single SoC substrate. Velasumy et al. [5] 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 for small numbers of cores, bus-based interconnect approaches are generally not scalable for large core counts [6]. Additionally, the CoreConnect bus uses far more resources than necessary to implement communication and control for monitor data. The IBM Power6 architecture [7] interconnects multiple sensors and actuators via a high-speed serial bus. This interconnect primarily serves as an external interface to voltage and temperature control via an I^2C bus for a modest number of cores.

MNoC [8] builds on ideas previously used for SoC debug and test. The JTAG boundary scan interface provides a serial scan interconnect which typically operates at 1 MHz. This low bandwidth chain consumes a minimal amount of resources and provides scalability. A recent, enhanced debug system by Dafca, Inc. uses multiplexers to collate debug information to one or more debug control points. Unlike MNoC, debug subsystems do not attempt to use collected information to influence SoC run-time operation.

B. Related On-Chip Interconnects

Numerous network-on-chip architectures [9] 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 network-on-chip (NoC) routers are optimized for routing bandwidth and consume considerable chip resources. Often, individual NoC routers require tens of thousands of transistors [10], include datapath widths of 32 to 256 bits, and buffer tens to hundreds of data values. For example, TILE64 [11][12] interconnects 64 processors with a series of 8×8 single channel interconnect meshes. The TRIPS processor uses a 4×10 mesh on-chip network [13] with prioritized channels to interconnect various SoC resources. In contrast, our approach attempts to minimize resource count to exactly the bandwidth and buffering required for SoC monitoring.

This manuscript significantly extends a previous MNoC conference publication [8]. In this manuscript we consider MNoC-based connections to DVFS control and specialized monitors. Additionally, quantitative comparisons are made between MNoC-transferred monitor data and monitor data transferred with existing NoC interconnect. The impact of both approaches on DVFS and overall system performance are also explored.

III. Integrated Multicore Monitoring

A. *Monitor Network-on-Chip Overview*

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, 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 Fig 1. The MNoC is interfaced to a monitor executive processor (MEP), which provides a software layer to implement 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 challenges of interconnect 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.

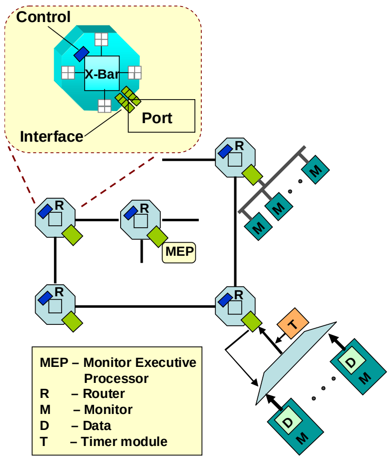

Fig. 1: Detailed view of MNOC for multiple cores

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. Although other topologies could possibly be used with MNoC, a mesh-like topology represents a simple, extensible solution for initial exploration. Two types of monitors are supported by MNoC: (1) monitors that put data into the network at regular intervals and (2) monitors that report data occasionally. For example, thermal monitors generally report temperature periodically, while error monitors only report data in the event of an error. For type 1, data requests are forwarded to the monitors by the associated router interfaces. Interrupts are used to support unexpected events detected at monitors for type 2. 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 that are generated occasionally via interrupts are usually critical in nature and are hence tagged with a higher priority. Periodically-inserted monitor messages are usually 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.

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 included in the packet to enable the routers to differentiate critical data from regular data. 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 [6] 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. Since most traffic is routed to a centralized MEP, the routing table can have a small number of entries (generally less than 8). The irregular placement of monitors results in an irregular mesh topology leading to concerns regarding deadlock. A fault tolerant mesh routing algorithm [14] is used to generate deadlock free paths that are stored in the routing tables. Since no monitor-to-monitor communication is used, 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.

B. Monitor Interfaces to MNoC

Direct, multiplexer and bus monitor interfaces provide a flexible selection of connections between monitors and MNoC routers. The interface control logic is able to support both monitors that inject data periodically and monitors that inject data occasionally. In the MNoC architecture, the monitors and the network router connect through a master-slave interface with the router serving as the master and the monitor as a slave. The high-level architecture of a monitor-network interface which includes a bus is shown in Fig. 2.

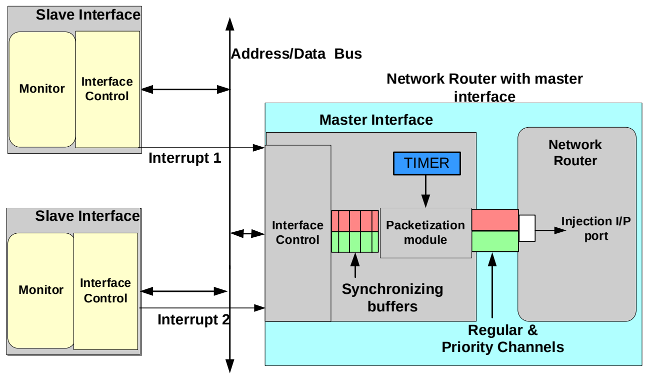

Fig. 2: Monitor –bus – network interface

Typically, interface control logic is built to read data at a pre-determined rate from multiple connected monitors. A control state machine is used at the router interface to sample connected monitors according to a pre-set schedule. Each interrupt-driven monitor has a dedicated interrupt line connected to the router interface that generates an interrupt when a data read is required. In the event of an interrupt, the controller breaks away from the original read sequence to generate a read address for the interrupting monitor. Any data value read from an interrupting monitor is tagged as high priority data. Once the monitor data is read, the controller appends it with the address of the originating monitor and the data's associated priority value. The data value is then written into a synchronizing FIFO, which is read by the packetization module (Fig. 2). The packetization module converts the data into flits and forwards them to the appropriate channel in the network (regular or priority).

To illustrate interface flexibility, an example multiplexer-based interface to thermal monitors is shown in Fig. 3. A contemporary eight-bit thermal monitor [1] is used to collect thermal information. Only one monitor is shown for clarity although the output from several monitors could be connected to the multiplexer. Thermal monitor sampling is triggered by the sample signal from the interface controller state machine. When thermal data is present and the appropriate output first-in, first-out buffer (FIFO) (regular or priority) is not full, the thermal data and routing information is put into a packet and sent to the appropriate channel.

As shown in Figs. 2 and 3, the packetization module can append monitor data with a time stamp from an embedded timer which is used to identify 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 = 1.0005ms, the MEP could interpret the current temperature value to be 20.00003 degrees using a known, average temperature gradient of 0.06 deg/ms [3]. A single timer is shared across several interfaces.

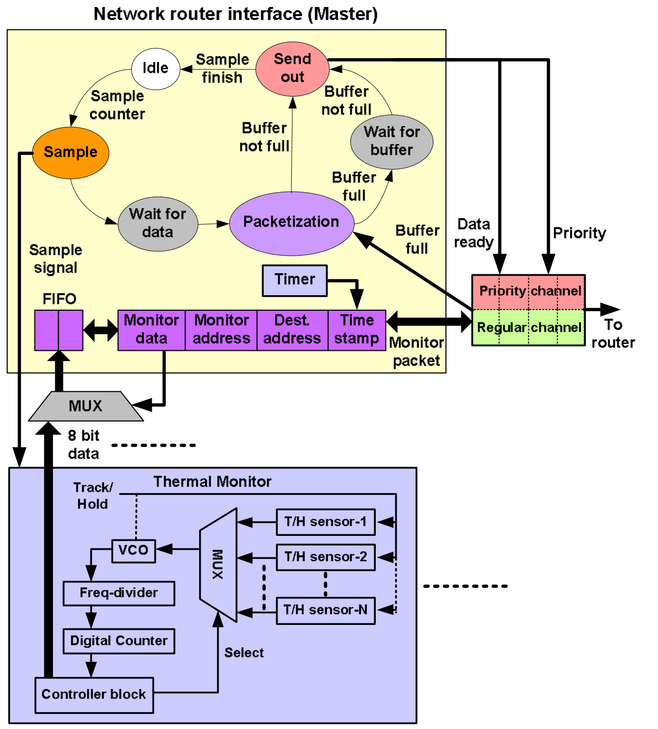

Fig. 3: Thermal monitor and multiplexer router interface controller

The MEP and the network router also connect through a master-slave (MEP-router) interface. Monitor data received from either of the router channels is read from separate FIFOs by a de-packetization module at the router-MEP interface. The MEP software reads information from the FIFOs at regular intervals with consideration given to priority data. Once data is received, the MEP uses the source information to determine the type and location of the monitor that sent the data and takes necessary action by affecting system parameters.

In our monitoring subsystem, MNoC is also used to interconnect the MEP and system controllers, such as voltage controllers and frequency controllers. Current digital interfaces to controllers generally require a small number of bits written to a register [15][16]. For example, the voltage and frequency of each core can be adjusted locally using this type of interface.

C. *MNoC Router Architecture*

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 (e.g. ≤ 24 bit) flit 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 parameterizably instantiated by designers. The optimal buffer sizes and widths can be determined based on the required latency and bandwidth for different monitoring systems. The parameter choices trade off performance (bandwidth and latency) and overhead (area and power).

For MNoC, input buffering is used instead of output buffering due to the low overhead that input buffering offers [17]. 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 separate priority and regular virtual channels. 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) travels in the priority channel until it reaches the destination.

MNoC routers employ a credit-based flow control to regulate data traffic and avoid packet dropping. Each router has buffer slot counters that keep track of the number of empty regular and priority channel buffer slots in adjacent routers. Buffer space availability is communicated by adjacent routers using credit messages. Flits that enter the MNoC router pass through three router pipeline stages: routing table look up, switch arbitration, and switch traversal. Once switch access is granted by switch arbitration, a flit enters the final pipeline stage where it traverses the crossbar and enters the appropriate channel in the next router. The priority channel is given preference during switch arbitration to ensure the lowest possible priority channel latency. The arbiter grants access to the regular data channel in a random fashion.

D. *Monitoring Subsystem Design*

We view the design of the monitoring subsystem as an action that can be performed in concert with the design of main SoC resources. A high-level design flow for creating an MNoC-based monitoring subsystem is shown in Fig. 4. Initially, the designer specifies the parameters of the monitor data including required bandwidth and latency and the permissible area and power of the monitoring subsystem. An initial series of parameters, including MNoC buffer size and bitwidth, number of monitors per core, and number of MEPs are selected. The performance of MNoC is then determined via an interconnect simulator which takes network topology and congestion into account. The results of monitor data use can then be assessed with the use of an architectural simulator. The evaluation of the monitoring subsystem can be performed in an iterative loop until acceptable parameters are located.

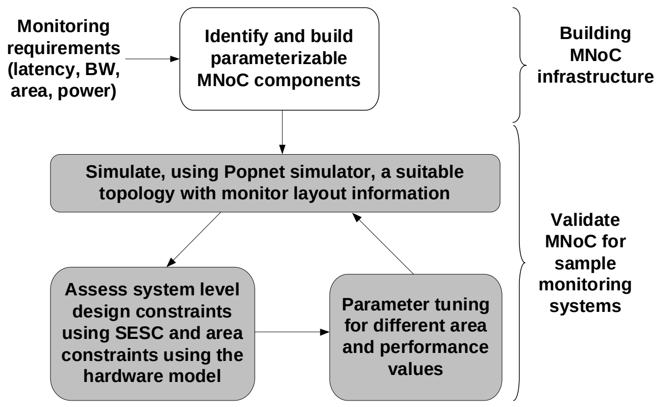
Monitoring requirements (latency, BW, area, power)
Identify and build parameterizable MNoC components
Building MNoC infrastructure
Simulate, using Popnet simulator, a suitable topology with monitor layout information
Validate MNoC for sample monitoring systems
Assess system level design constraints using SESC and area constraints using the hardware model
Parameter tuning for different area and performance values

Fig. 4: System-level monitoring design approach

Although the steps shown in Fig. 4 could eventually be automated, the examples in subsequent sections are enumerated via user-guided experimentation. The parameters used in this analysis are justified in the next section.

IV. Experimental Parameter Assessment

A. *Experimental Approach*

In an initial experiment, a series of simulation and synthesis evaluations similar to the type illustrated in Fig. 4 have been performed. The Popnet interconnect simulator [18] has been significantly modified to estimate bandwidth and latency values for the heterogeneous MNoC interconnect. The router pipeline and the routing protocol were modified and additional support for an expanded set of interfaces (e.g. bus, multiplexer) was provided. 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 allows for evaluation of area for different flit widths and buffer sizes. The hardware model, which operates at 500 MHz, was synthesized by Synopsys Design Compiler using a 90nm standard cell library [19]. Architectural simulations were performed using the SESC architectural simulator [20] to quantify the benefits of employing our monitor subsystem at a system level.

B. *Parameter Evaluation*

In an initial experiment to illustrate multicore MNoC tuning tradeoffs, 64 thermal monitors (8 monitors per processor core) are used to report temperature values from various multicore locations at a variety of monitor data injection rates. The processor core floorplan used for thermal modeling is based on the AMD Athlon 64 processor [21]. A mock-up of a layout of the eight core system including MNoC is shown in Fig 5. One MNoC router per core collects thermal data from the 8 thermal monitors using a multiplexer. Thermal monitors [22] can be classified as low bandwidth monitors. Justification for the per-core thermal monitor and MNoC router count is given in Section V.

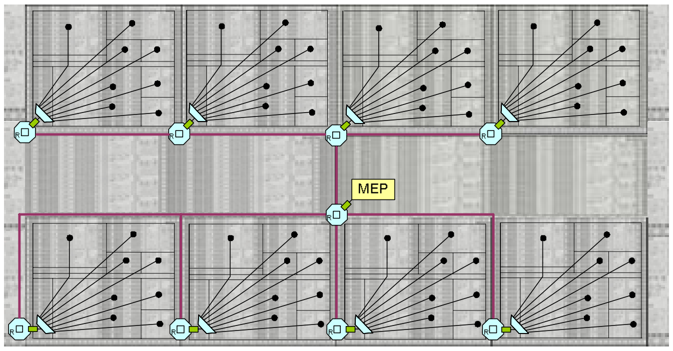

Fig. 5: Monitor network on chip layout for thermal monitors on an 8 core processor

The MEP is attached to a dedicated router (Fig. 5) at a location central to the routers. The resultant topology is an irregular mesh. With this 9 router setup, deadlock-free routing [14] was used to generate paths from the routers to the MEP. Our interconnect simulator was used to evaluate the latency of this network for different network parameters.

Fig. 6 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 [8]. The X-axis, cycles between injections, indicates the number of clock cycles between two sampling points for each thermal monitor. The Y-axis, network latency, indicates the average time required (in clock cycles) for data to travel from a monitor to the MEP. Fig. 6 indicates a significant regular channel dependence on input buffer storage for sizes less than 4. No latency reduction is achieved by increasing buffer size for buffer sizes greater than and equal to 4. For longer delays between injections, the regular channel latency becomes insensitive to buffer size. Although not shown in the figure, latency remains roughly constant for the 5% of total traffic which is priority traffic. Network latencies between 16 and 21 cycles were found for all injection rates.

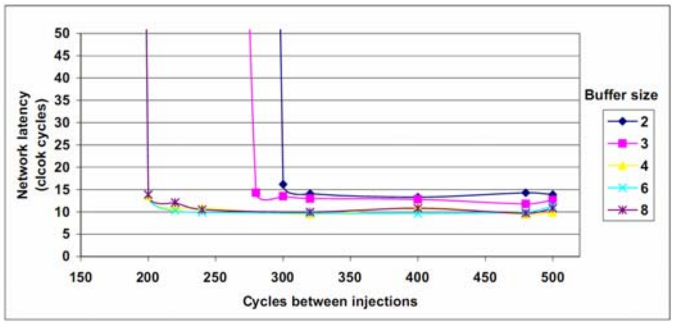

Fig. 6: Regular channel latencies for different buffer sizes for flit width = 24

Fig. 7 shows a plot of network latency versus injection rate for different flit widths for regular traffic. As flit width increases, the sampling rate which saturates the network becomes higher.

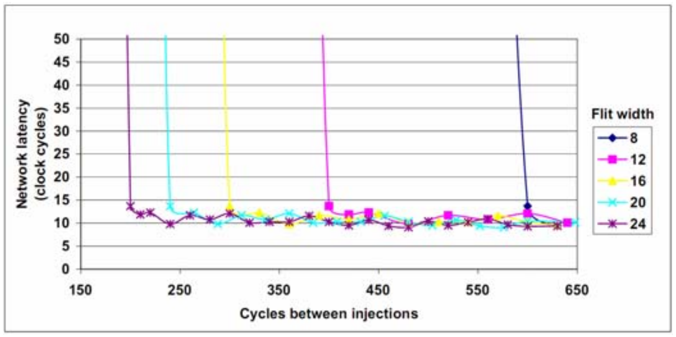

Fig. 7: Regular channel latencies for different flit widths for buffer size = 4

Overall, it can be inferred from the results that for higher cycles between injection (lower sampling rates and lower bandwidth) the latency of a non-saturated network is mostly insensitive to network parameters such as buffer size and flit width. At such low sampling rates, low network latency (less than 20 clock cycles) 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.

Of course, larger flit and buffer size parameters require a larger network area. Table 1 shows area results for the 9 router MNoC system estimated at a 90nm technology node. MNoC router area has been determined via synthesis. MNoC wire area has been estimated using a methodology based on wire pitch and wire length [23].

TABLE 1: MNOC AREA RESULTS FOR A 9 ROUTER SYSTEM

Flit width	Buffer size = 2			Buffer size = 4		
	Router area in mm^2	Wire area in mm^2	Total area in mm^2	Router area in mm^2	Wire area in mm^2	Total area in mm^2
12	0.408	0.035	0.443	0.438	0.035	0.473
14	0.445	0.041	0.486	0.481	0.041	0.522
16	0.482	0.047	0.529	0.523	0.047	0.570
18	0.519	0.053	0.572	0.566	0.053	0.619
20	0.556	0.058	0.614	0.610	0.058	0.668
24	0.602	0.070	0.672	0.693	0.070	0.763
Flit width	Buffer size = 8			Buffer size = 16		
	Router area in mm^2	Wire area in mm^2	Total area in mm^2	Router area in mm^2	Wire area in mm^2	Total area in mm^2
12	0.494	0.035	0.529	0.606	0.035	0.641
14	0.547	0.041	0.588	0.676	0.041	0.717
16	0.598	0.047	0.645	0.743	0.047	0.790
18	0.648	0.053	0.701	0.810	0.053	0.863
20	0.701	0.058	0.759	0.876	0.058	0.934
24	0.801	0.070	0.871	1.013	0.070	1.083

As seen in Fig. 8, as MNoC size increases, the saturation injection rate of a network with 24-bit flits and a buffer size of 4 is reduced. However, network bandwidth increase by 30% leads to a higher saturation injection rate. This result indicates that MNoC can be expanded to a larger number of cores by increasing the network bandwidth via, for example, more aggressive pipelining or signaling to retain a similar saturation injection rate. For this experiment, a new MEP is allocated per 16 cores, reducing the global distribution of monitor data, although some congestion occurs at the borders of different regions. No inter-MEP communication is performed as part of the experiment.

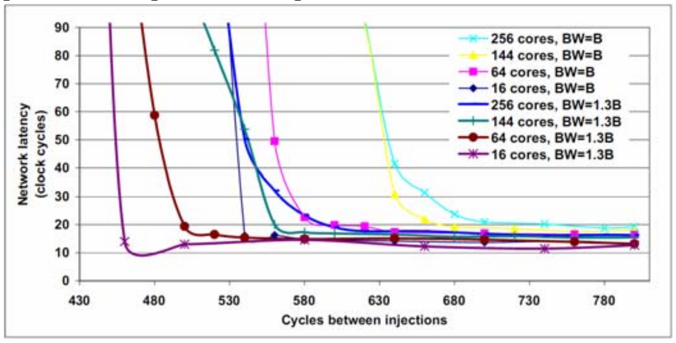

Fig. 8: MNoC performance with increasing number of cores and bandwidth for buffer size 4

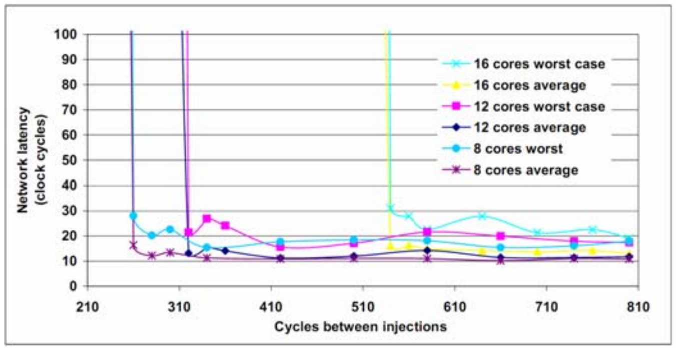

Fig. 9: Average versus worst-case latency for MNoC for buffer size = 4 and flit size = 24

Generally, the worst-case latency for MNoC does not vary substantially from the measured average latency. As shown in Fig. 9, the curve knees for both average and worst-case latencies occur at roughly the same injection rates. Our design methodology allows designers to consider both latencies in developing an appropriate MNoC system.

V. Applications of Dedicated Monitoring

To better justify the use of MNoC, two applications of monitor data, dynamic frequency scaling (DFS) and voltage droop recovery, are examined. MNoC is used to transfer monitor data for each application based on monitoring parameters obtained from previously-published results.

A. *Experimental approach*

These experiments use SESC to simulate multiple processors and a central MEP, similar to the eight core system shown in Fig. 5. Each core has a private L1 and L2 cache. In comparison to a commercial 8 core processor [24], the area overhead of an MNoC with a buffer size of 4 and 24-bit flits in the configuration shown in Fig. 5 is 0.763/378 mm^2 = 0.20%. 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 [22]. 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 dynamically obtain the temperature readings. This approach enabled the MEP to sample the temperature readings at a pre-determined interval and execute the DFS and voltage droop recovery algorithms.

B. *Dynamic Frequency Scaling*

In a DFS experiment, a monitor subsystem that satisfies system design constraints while providing a performance benefit is demonstrated. Dynamic frequency scaling is performed in response to on-chip thermal conditions. Thermal monitor data is forwarded to a MEP which then determines a specific processor's frequency. Although some contemporary DFS approaches support the use of thermal data within a core, several recent efforts [25] have determined the local and global effects of thermal data on DFS.

To successfully evaluate the use of dedicated monitoring, a realistic dispersal of monitors per core and a realistic sampling rate to support DFS is needed. In the following experiments, a total of 8 thermal monitors per processor core are used for the Athlon cores. Previous work [26][27] has shown that for an architecture and per-processor transistor count (approximately 68 million transistors) similar to these cores, 8 thermal monitors per core is appropriate. Numerous contemporary thermal monitors generate eight-bit data [1], the thermal monitor data width used here.

Previous DVFS experiments have used a temperature gradient of finer than 0.1 degC/sec [2] for available monitors to assess thermal activity. For example [22], temperature values used for DVFS have been sampled every 10,000 cycles for a 3GHz core to achieve a temperature resolution of less than 0.1 degC. For the following experiment, a sampling period of 1,600 cycles at 500 MHz is needed to match this rate. To ensure a conservative result, sampling at each thermal monitor is performed every 800 cycles. An MNoC configuration with flit width of 24 bits and an input buffer size of 4 was used to meet the required bandwidth and match subsequent MNoC experiments described in Sections V.C and VI. Data transfer requires a single flit. For this experiment, all data paths shown in Figs. 1, 2, and 3 are 24 bits, unless explicitly marked otherwise in the figures.

Flit size can be justified as follows. The largest analyzed system contains 17 routers and 128 total monitors and each thermal data value includes 8 bits. Thus, the thermal data packet contains 4 source router address bits, 4 destination router address bits, 3 monitor select bits, and an 8-bit data value (19 bits total). The 5 extra bits in the MNoC flit allow for scalability to larger MNoC configurations, as discussed in Section VI.

Dynamic frequency scaling of a processor system improves system performance by operating cores within power dissipation and temperature limits. Two trials were performed on multicore systems to demonstrate the benefits of DFS on a benchmark application. Three floating point benchmarks [28], listed in Table 2, were used to conduct the experiments for a total of at least 2 billion instructions. The temperatures reported by the monitors are collected by MNoC and transported to the MEP which uses the data for dynamic frequency scaling.

TABLE 2: RUNTIMES FOR NON-MNOC (COLUMN 3) AND NON-MNOC (COLUMN 4) CASES. THE MNOC USES DFS TO TOGGLE CORE CLOCK FREQUENCIES BETWEEN 2 AND 1 GHZ. THE NON-MNOC CASE USES A CONSTANT 1 GHZ FREQUENCY

Core number	Test bench	Runtime for Freq = 1 GHz (sec)	Runtime for Freq = 2/1 GHz (sec)	Performance benefit due to DFS
4	Whetstone	3.360	2.420	27.98%
	Water-spatial	0.420	0.326	22.38%
	Water-nsquared	0.424	0.328	22.64%
8	Whetstone	2.750	2.250	18.18%
	Water-spatial	0.420	0.352	16.19%
	Water-nsquared	0.424	0.356	16.04%
12	Whetstone	2.270	1.520	33.04%
	Water-spatial	0.428	0.372	13.08%
	Water-nsquared	0.428	0.374	12.62%
16	Whetstone	1.750	1.350	22.86%
	Water-spatial	0.428	0.348	18.69%
	Water-nsquared	0.432	0.354	18.06%

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 (1 GHz) to ensure that the same power and temperature limits are not violated. The run time was again noted and the resulting performance improvement was calculated. The results of the evaluation for 4, 8, 12, and 16 core systems are shown in Table 2. The performance advantage of employing DFS using MNoC is consistently above 15% as the number of cores is increased, as shown in the table. The resulting MNoC power for Whetstone in an eight-core system was determined to be 122 mW.

For this experiment, MNoC latency does not notably affect the amount of performance benefits that are achieved. The thermal monitor sampling rate of one per 800 cycles is sufficiently low to avoid MNoC congestion. The achieved performance benefit results listed in the fourth column of Table 2 are nearly the same (within 0.2%) for an interconnect which could instantaneously transfer monitor data from monitors to the MEP with zero latency. Experiments with a standard master-slave bus also show similar results to MNoC.

C. Voltage Droop Recovery

In an additional experiment, the system-level benefits of our monitor subsystem were determined for delay-based voltage control. 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 [7] which report 12 bits of delay data [29]. Delay monitors (critical path monitors) are placed next to microprocessor core thermal monitors to maintain consistency with previous work [7]. Generally, regions which experience hotspots and high current flow also experience a higher failure rate due to voltage droops and transistor heating. These regions also tend to contain processor critical paths and increased power density [7]. Monitor data is transported to the MEP through a multi-router MNoC, similar to the one shown in Fig. 5. The delay monitors require higher network bandwidth than thermal monitors since voltage values can change at a rate of over an order of magnitude per second [30], motivating a need for frequent sampling.

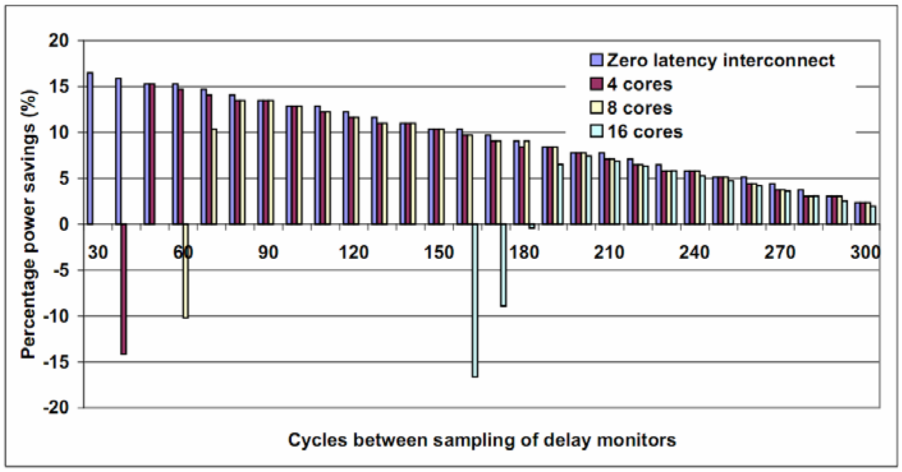

Fig. 10: Power savings in multicore processors using MNoC

TABLE 3: COMPARISON OF MNOC AND BUS VERSUS A ZERO-LATENCY INTERCONNECT

Injection rates	Zero-latency interconnect	Power savings (%)					
		4 cores		8 cores		16 cores	
		Bus	MNoC	Bus	MNoC	Bus	MNoC
190	8.56	8.15	8.21	6.88	8.16	1.61	6.52
200	8.04	7.65	7.68	6.44	7.63	1.41	7.42
210	7.51	7.14	7.17	5.98	7.10	1.18	6.88
220	6.98	6.62	6.63	5.51	6.57	0.91	6.34
230	6.45	6.10	6.09	5.04	6.04	0.62	5.81
240	5.91	5.58	5.55	4.55	5.50	0.30	5.27
250	5.38	5.05	5.01	4.06	4.96	-0.04	4.72
260	4.84	4.52	4.48	3.56	4.41	-0.40	4.18
270	4.29	3.99	3.92	3.06	3.87	-0.77	3.63
280	3.74	3.45	3.37	2.55	3.32	-1.16	3.08
290	3.19	2.91	2.82	2.03	2.77	-1.57	2.53
300	2.64	2.36	2.26	1.51	2.21	-1.98	1.98
Average	5.63	5.29	5.27	4.26	5.21	0.01	4.86
Average. reduction in power savings w.r.t. zero-latency interconnect (%)		6.5%	7.4%	26.4%	8.5%	98.5%	14.6%

In an MNoC-based system, the MEP increases the voltage of a core from 1.2V to 1.4V in response to a voltage droop event at the core. In a non-MNoC system the supply voltage is constant and is set conservatively to a 1.4V value that accounts for the maximum voltage droop. The experiment was conducted for 4, 8 and 16 processor cores. Fig. 10 shows the percentage power savings for MNoC-supported processor cores versus a system without voltage droop recovery. The X-axis indicates the number of clock cycles between two sampling points for each delay monitor. As seen from the results, all three configurations result in power savings versus the non-MNoC case for specific sampling rates. The MNoC power consumption is included in this analysis.

An additional set of data points is also included in Fig. 10 which represents power savings for a 16 core system with a zero-latency monitor interconnect. These numbers represent the power savings if monitor data could immediately be moved from the monitors to the MEP rather than passing through the MNoC. The difference between the MNoC case and zero-latency case is about 15% for 16 cores. In general, as the number of cores and overall monitor count increases, more bandwidth is required from the network. This trend motivates the need for a more distributed medium rather than buses or serial links.

Fig. 10 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. The exact combinations of sampling intervals and MNoC configurations for a proposed multicore can be determined during system design. An additional approach to overcoming this issue for substantially larger collections of processors would be to dedicate one MEP per a fixed number of cores (e.g. 16).

Table 3 illustrates the average power savings for a series of voltage monitor injection rates for a zero-latency interconnect, MNoC and master-slave bus. Overall, MNoC performs within 8% (8 core) and 15% (16 core) of the zero latency case in terms of power savings. Similar bus cases result in a 26% average reduction in power savings versus the ideal case for 8 cores and a reduction of nearly 100% for 16 cores. Fine-grained on-chip dynamic voltage scaling typically uses a voltage sampling resolution of 200-400 ns [15] which fits within the target injection rate range shown in the table.

A. *Error Recovery*

The allocation and placement of error monitors barely affects the performance of MNoC, since the injection rate of error monitors is extremely low. Soft error rate is generally below one per hundred thousand hours [31] and each error requires a single bit error indicator. When a soft error occurs, immediate system remediation is needed to prevent the propagation of the error. In the monitoring system, soft error monitor data has the highest priority and must be transmitted in the priority channel. The expected error rate did not impact the results of DFS or voltage droop experiments.

VI. COMPARISON OF MNOC AND STANDARD NOC FOR MONITOR DATA TRANSFER

The use of MNoC in a multicore system that already includes a network-on-chip may seem redundant. In this section, the benefit of including MNoC as a *separate* network dedicated for monitor data traffic is contrasted with using an *existing* NoC to transfer both monitor and application traffic as a proof-of-concept implementation.

A. *Experimental approach*

In order to demonstrate the benefit of using MNoC as a separate network in NoC-based multicore systems, a model of a shared memory multicore system [32][33][34] based on the TRIPS on-chip network (OCN) [13] architecture and a processor architecture based on Tile64 [12] is used. The shared-memory multicore system model has 16 cores connected by a 4×4 mesh NoC with 4 virtual channels. Each processor has its own L1 data and instruction cache. The shared main memory is connected to a router on one side of the mesh [11].

The NoC performs communication between the processors and the shared memory. The NoC architecture has a 256-bit flit width, 4 virtual channels with different priority levels, and a buffer size of 2 flits [13]. All NoC packets have 2 flit packet sizes. In the case of physical routing channel contention, the crossbar selects the higher priority channel.

As stated in Section V, according to our synthesis results using 90nm technology, MNoC can operate at a speed of 500MHz. The hardware speed of TRIPS using 130nm technology is 500MHz and the 90nm Tile64 processor operates at 750 MHz. Thus, in this experiment, it is assumed that the system model operates at 500MHz using 90nm technology.

1) Monitor and MNoC configurations

To evaluate monitor data transfer in this multicore system fairly, the number, placement and data rate of monitors is needed. In our evaluation, thermal monitors and delay monitors (critical path monitors) are considered. Similar to the configurations described in Section V, every core in the multicore system contains eight thermal monitors and eight delay monitors, which are placed in hotspots [26][27]. The total system monitor count is 240.

2) MNoC configuration and experiment setup

Monitor sampling rate determines the injection rate of monitor data into the monitor interconnect. Similar to the experiment described in Section V.B, a thermal monitor injection rate of 1 value every 800 cycles is used. Since delay values can fluctuate rapidly, delay monitor sampling should be performed about every 400 ns [15]. For an MNoC frequency of 500MHz, a delay monitor sampling rate of 200 cycles is sufficient, leading to a per-monitor MNoC injection rate of the same value.

For this series of multicore experiments, MNoC routers have a 24-bit flit width, 2 virtual channels and buffer sizes of 2 flits. All data paths shown in Figs. 1, 2, and 3 are 24 bits, unless explicitly marked otherwise in the figures. To assess inter-monitor MNoC topology, the monitor data injection rate of each router must be known. For simplicity, each core is assigned one MNoC router and each monitor packet is one 24-bit flit. A total of 12 bits (6-bit destination address and 6-bit source address) are sufficient for router addressing. The thermal monitors and delay monitors use 8-bit and 12-bit data, respectively [1][29]. Since the thermal monitor injection rate is once every 800 cycles and the delay monitor injection rate is once every 200 cycles and there are 8 pairs of thermal and delay monitors per core, the per-router monitor data injection is about (8/800+8/200) = 1 per 20 cycles.

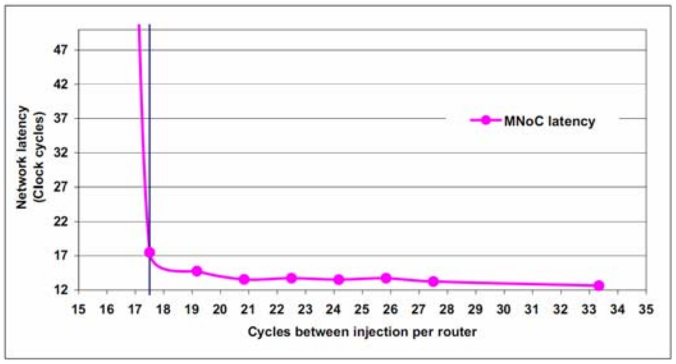

Fig. 11: MNoC latency versus injection rate per core/router. The solid line represents the minimum injection period per core.

Fig. 11 illustrates MNoC latencies for different injection rates determined via simulation for an MNoC with the flit width, buffer size, and core count parameters noted earlier in this section. Latency is less than 20 clock cycles if per-router injection occurs less frequently than once every 17 clock cycles, as shown in Fig. 11.

3) Comparison test cases

To show the benefit of using MNoC as a separate network, monitor data latency is compared in a variety of monitor data collection environments. These specific cases include:

1. **NoC with MNoC** – In this case, all monitor data is transferred via MNoC. All other inter-processor traffic ("application traffic") is transferred using the multicore NoC.
2. **MT-NoC (mixed traffic NoC)** – All monitor data and application traffic is intermixed on the four virtual channel (VC) NoC.
3. **Iso-NoC (isolated channel NoC)** – All monitor data is allocated to one specific VC in the four VC channel NoC. Application traffic uses the remaining three VCs.

Note that cases 2 and 3 do not include an MNoC. The packet size of the monitor traffic is one 24-bit flit. Thermal and delay monitors for these cases are connected to the NoC router through a multiplexer. The processor in position (2, 2) of the mesh performs the functionality of the monitor executive processor (MEP) by processing monitor data. In the Iso-NoC case, monitor data is given priority over application traffic, while all traffic has equal priority for case 2. Cases 1, 2 and 3 use a four virtual channel NoC. In the following section, the latencies and area overheads of the NoC with MNoC and NoC-only implementations are considered.

B. Comparison results

In this experiment, the *Ocean.mips* Splash benchmark is used to generate network traffic traces. The traces are then simulated using an enhanced version of Popnet to calculate both NoC and monitor packet latencies. For case 1 simulations, two simulations are performed, one for MNoC

(MNoC-modified Popnet) and a second for the NoC (standard Popnet). Other cases require only one Popnet simulation.

Fig. 12 shows the latency for application traffic for a range of injection rates for delay monitors. A constant injection rate of one value per 800 cycles is assumed for thermal monitors. The thermal monitor data was included in generating the plots. Fig. 12 shows that the latency of application data in mixed-traffic NoC (MT-NoC) for the 4 virtual channel NoC is higher than the data latency for separated NoC and MNoC (NoC with MNoC). This latency increase is expected since the transfer of monitor data on the NoC increases resource contention. The isolation of monitor data into a separate virtual channel helps latency reduction somewhat, but contention for the router ports eventually causes latency to spike once again. The best channel latency for application traffic is achieved when only application traffic (NoC with MNoC) is transferred on the 4 virtual channel NoC infrastructure.

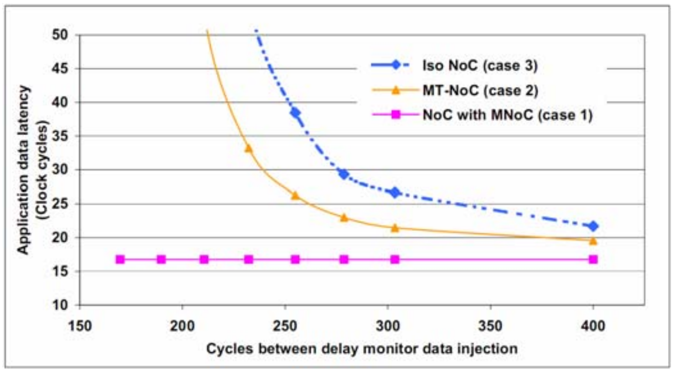

Fig. 12: Latency comparison for application traffic versus delay monitor data injection period. A fixed thermal monitor data injection rate of one value per 800 cycles is also used. Application data is transferred with the lowest latency using the NoC in a system which includes both an MNoC and NoC.

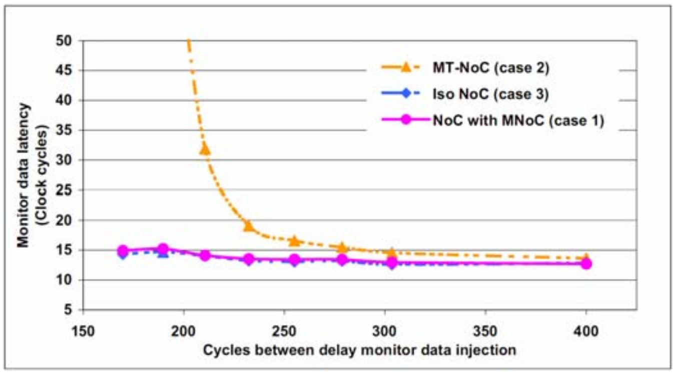

Fig. 13: Latency comparison for monitor traffic versus delay monitor data injection period. A fixed thermal monitor data injection rate of one value per 800 cycles is used. Although MT-NoC and NoC with MNoC cases support the same monitor data transfer latencies, Fig. 12 showed that the application data latency for MT-NoC is higher.

Fig. 13 shows monitor data latencies in the different scenarios which include the thermal monitor data. Since a dedicated, high-priority NoC virtual channel is used to transmit monitor data in the MT-NoC case, its monitor data latency performance matches MNoC, although as noted in Fig. 12, its application data latency is worse.

The case 2 and case 3 plots in Figs. 12 and 13 represent about 80% monitor traffic and 20% application traffic. Table 4 indicates the performance change of the application as NoC latency is increased. Not surprisingly, more congested NoC channels lead to reduced application performance.

TABLE 4: PERFORMANCE COMPARISON OF THE NOC SYSTEM WITH AND WITHOUT MNOC

Delay monitor injection rate (cyc. between inject.)		400	300	250
NoC with MNoC (cycles)		44,247,340		
MT-NoC	NoC latency increase (cycles)	3	5	10
	Performance (cycles)	44,421,683	44,531,612	44,674,131
	Performance degradation (%)	0.39	0.64	0.96
Iso NoC	NoC latency increase (cycles)	5	10	23
	Performance (cycles)	44,531,612	44,674,131	45,068,956
	Performance degradation (%)	0.64	0.96	1.86

Overall, the *Ocean.mips* application has a relatively low amount of inter-processor traffic allowing for a higher fraction of monitor traffic. As shown in Fig. 14, designers could consider the tradeoffs between application and monitor traffic injection rates which maintain a constant, per-router NoC injection rate. MNoC is a reasonable choice for NoC-based systems which cannot meet the application data latencies with mixed traffic, as shown in Figs. 12 and 13. As noted earlier in this section, the per-router monitor data injection rate used for these plots was about one monitor data value every 20 cycles.

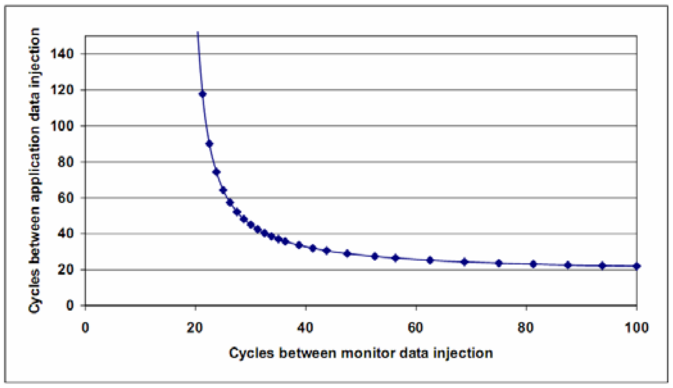

Fig. 14: Tradeoff between application and monitor injection rates for constant NoC router injection rate of 1 value per 18 cycles.

In a final experiment, the area of the monitor data transfer cases was considered. The Synopsys Design Compiler was used to synthesize MNoC and a 4 virtual channel NoC in 90nm technology. The system speed was confirmed to match 500MHz.

TABLE 5: HARDWARE COMPARISON FOR 16 CORE SYSTEM

Name	Flit width	Virtual channel number	Buffer size	Speed (ns)	H/W area in um^2 (per router)	Total H/W area in um^2
MNoC	24	2	2	2	69901	1118416
NoC	256	4	2	2	1064700	17035200
MNoC/4VC NoC (%)						
6.565						

Table 5 shows the hardware cost of MNoC and 4 virtual channel NoC. The entire MNoC router is about 7% of the size of an entire NoC router.

VII. CONCLUSION

This work presents a dedicated 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 0.20% was achieved for the monitor interconnect when applied to an 8-core system based on AMD Athlon processors. The new monitor data transfer approach is directly contrasted with transfer using an existing NoC, which also carries application data traffic. Experimental results show that the use of MNoC lowers both monitor data latency and application traffic in multicores, especially when the monitor sampling rate is high.

ACKNOWLEDGMENT

The authors would like to acknowledge the suggestions of our SRC liaisons at Intel, AMD, and Freescale.

REFERENCES

[1] B. Datta and W. Burleson, “Low-power, process-variation tolerant on-chip thermal monitoring using track and hold based thermal sensors,” in *Proc. ACM/IEEE Great Lakes Symposium on VLSI*, pp. 145-148, May 2009.

[2] O. Khan and S. Kundu, “A framework for predictive dynamic temperature management of microprocessor systems,” in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp. 258-263, Nov. 2008.

[3] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, M. Millican, W. Parks and S. Naffziger, “Power and temperature control on a 90nm Itanium family processor,” in *IEEE Journal on Solid State Circuits* , vol. 41, no 1, pp. 229-237, Jan. 2006.

[4] M. Saen, K. Osada, S. Misaka, T. Yamada, Y. Tsujimoto, Y. Kondoh, T. Kamei, Y. Yoshida, E. Nagahama, Y. Nitta, T. Ito, T. Kameyama and N. Irie, “Embedded SoC resource manager to control temperature and data bandwidth,” in *Proc. IEEE International Solid-State Circuits Conference*, pp. 296-604, Feb. 2007.

[5] S. Velusamy, W. Huang, J. Lach, M. Stan and K. Skadron, “Monitoring temperature in FPGA based SoCs,” in *Proc. IEEE International Conference on Computer Design*, pp. 634-637, Oct. 2005.

[6] T. Bjerregaard and S. Mahadevan, “A survey of research and practices of network-on-chip,” in *ACM Computing Surveys*, vol. 38, no.1, pp. 1-51, Mar. 2006

[7] M. Floyd, S. Ghiasi, T. Keller, K. Rajamani, F. Rawson, J. Rubio and M. Ware, “System power management support in the IBM Power6 microprocessor,” in *IBM Journal of Research and Development*, vol. 51, pp. 733-746, Nov. 2007

[8] S. Madduri, R. Vadlamani, W. Burleson and R. Tessier, “A monitor interconnect and support subsystem for multicore processors,” in *Proc. IEEE/ACM Design Automation and Test in Europe Conference*, pp. 761-766, Apr. 2009.

[9] P. Pande, C. Grecu, A. Ivanov, R. Saleh and G. De Micheli, “Design, synthesis, and test of networks on chips,” in *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 404-413, Sep. 2005.

[10] F. Moraes, N. Calazans, A. Mello, L. Möller and L. Ost, “HERMES: An infrastructure for low area overhead packet-switching networks on chip,” in *Integration: The VLSI Journal*, vol. 38, no. 1, pp. 69-93, Oct. 2004

[11] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J. Brown and A. Agarwal, “On-chip interconnection architecture of the Tile processor,” in *IEEE Micro*, vol. 27, no. 5, pp. 15-31, Sep. 2007.

[12] S. Bell, et. al, “TILE64 processor: A 64-Core SoC with mesh interconnect,” in *Proc. IEEE International Solid-State Circuits Conference*, pp. 88-598, Feb. 2008.

[13] P. Gratz, C. Kim, R. McDonald, S. Keckler and D. Burger, “Implementation and evaluation of on-chip network architectures,” in *Proc. IEEE International Conference on Computer Design*, pp. 477-484, Oct. 2007.

[14] K. Chen and G. Chiu, “Fault-tolerant routing algorithm for meshes without using virtual channels,” in *Journal of Information Science and Engineering.*, vol. 14, no. 4, pp. 765-783, Dec. 1998.

[15] W. Kim, M. Gupta, G. Wei and D. Brooks, “System level analysis of fast, per-core DVFS using on-chip switching regulators,” in *Proc. IEEE International Symposium on High-Performance Computer Architecture*, pp. 123-134, Feb. 2008.

[16] W. Cheng and B. Baas, “Dynamic voltage and frequency scaling circuits with two supply voltages,” in *Proc. IEEE International Symposium on Circuits and Systems*, pp. 1236-1239, May 2008.

[17] Y. Tamir and G. L. Frazier, “High-performance multiqueue buffers for VLSI communication switches,” in *Proc. IEEE International Symposium on Computer Architecture*, pp. 343-354, Jun. 1988

[18] L. Shang, L. Peh and N. K. Jha, “Dynamic voltage scaling with links for power optimization of interconnection networks,” in *Proc. IEEE International Symposium on High-Performance Computer Architecture*, pp. 91-102, Feb. 2003

[19] UMC’s 90nm 1P9M logic/mixed mode low-K SP-HVT process library [Online]. Available: http://www.faraday-tech.com

[20] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, K. Strauss, S. Sarangi, P. Sack and P. Montesinos, “SESC simulator,” [Online]. Available: http://sesc.sourceforge.net.

[21] G. Link and N. Vijaykrishnan, “Thermal trends in emerging technologies,” in *Proc. IEEE International Symposium on Quality Electronic Design*, pp. 625-632, Mar. 2006.

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

[23] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, “Cost considerations in network on chip,” in *Integration, the VLSI journal*, vol. 38, no. 1, pp. 19-42, Oct. 2004.

[24] A. Leon, K. Tam, J. Shin, D. Weisner and F. Schumacher, “A power-efficient high-throughput 32-thread SPARC processor,” in *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 7-16, Jan. 2007.

[25] R. Jayaseelan and T. Mitra, “A hybrid local-global approach for multi-core thermal management,” in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp. 314-320, Nov. 2009.

[26] R. Mukherjee and S. Memik, “Systematic temperature sensor allocation and placement for microprocessors,” in *Proc. ACM/IEEE Design Automation Conference*, pp. 542-547, July 2006.

[27] S. Memik, R. Mukherjee, M. Ni and J. Long, “Optimizing thermal sensor allocation for microprocessors,” in *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 3, pp. 516-527, Mar. 2008.

[28] H. Curnow and B. Wichmann, "A synthetic benchmark," in *Computer Journal*, vol. 19, no. 1, pp. 43-49, Feb. 1976.

[29] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd and V. Pokala, "A distributed critical-path timing monitor for a 65nm high-performance microprocessor," in *Proc. IEEE International Solid-State Circuits Conference*, pp. 398-399, Feb. 2007

[30] R. Joseph, D. Brooks, and M. Martonosi. "Control techniques to eliminate voltage emergencies in high-performance processors," in *Proc. IEEE International Symposium on High-Performance Computer Architecture*, pp. 79-90, Feb. 2003.

[31] P. Shivakumar, M. Kistler, S. Keckler, D. Burger and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 389-398, Jun. 2002.

[32] S. Mukherjee, P. Bannon, S. Lang, A. Spink and D. Webb, "The Alpha 21364 network architecture," in *IEEE Micro*, vol. 22, no. 1, pp. 26-35, Jan. 2002.

[33] M. Monchiero, G. Palermo, C. Silvano and O. Villa, "Exploration of distributed shared memory architectures for NoC-based multiprocessors," in *Proc. IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, pp. 144-151, Jul. 2006.

[34] H. Freitas, D. Colombo, F. Kastensmidt and P. Navaux, "Evaluating network-on-chip for homogeneous embedded multiprocessors in FPGAs," in *Proc. IEEE International Symposium on Circuits and Systems*, pp. 3776-3779, May 2007.

Jia Zhao received the B.E degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2005 and the M.S. degree in microelectronics from Fudan University, Shanghai, China, in 2008. He is currently working toward the Ph.D degree at the University of Massachusetts, Amherst.

His research interests include multiprocessor architecture, VLSI design and reconfigurable computing.

Sailaja Madduri received the B.E degree in electrical and electronics engineering from Birla Institute of Technology and Science, Pilani, India, in 2005 and the M.S degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 2008.

She is currently with Intel Corporation, Hillsboro, OR, where she is involved in the design of next generation microprocessors.

Ramakrishna Vadlamani received the B.E. degree in electronics engineering from Veermata Jijabai Technological Institute, Mumbai, India, in 2004 and the M.S. degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 2010.

He is currently with Qualcomm Inc., Boxborough, MA, where he is involved in the design and verification of next generation basestation modems.

Wayne Burleson (M'84–SM'01) received the B.S. and M.S. degrees from the Massachusetts Institute of Technology, Cambridge, and the Ph.D. degree from the University of Colorado, Boulder, all in electrical engineering.

He is a Professor of electrical and computer engineering with the University of Massachusetts, Amherst. His research interests include VLSI design, reconfigurable computing, content-adaptive signal processing, embedded security, and multimedia instructional technologies.

Russell Tessier (M'00-SM'07) received the B.S. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1989, and the S.M. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, in 1992 and 1999, respectively.

He is an Associate Professor in electrical and computer engineering with the University of Massachusetts, Amherst, where he also leads the Reconfigurable Computing Group. His research interests include computer architecture, field-programmable gate arrays, and system verification.