Advance Access publication on 27 July 2011 doi:10.1093/comjnl/bxr069

SST: A Scalable Parallel Framework for Architecture-Level Performance, Power, Area and Thermal Simulation

MING-YU HSIEH[1,*], ROLF RIESEN[2], KEVIN THOMPSON[3], WILLIAM SONG[4] AND ARUN RODRIGUES[1]

[1]*Sandia National Laboratories, Albuquerque, NM, USA*
[2]*IBM Research, Dublin, Ireland*
[3]*Department of EE, New Mexico State University, Las Cruces, NM, USA*
[4]*Department of ECE, Georgia Institute of Technology, Atlanta, GA, USA*
[*]*Corresponding author: myhsieh@sandia.gov*

In this paper, we describe the integrated power, area and thermal modeling framework in the structural simulation toolkit (SST) for large-scale high performance computer simulation. It integrates various power and thermal modeling tools and computes run-time energy dissipation for core, network on chip, memory controller and shared cache. It also provides functionality to update the leakage power as temperature changes. We illustrate the utilization of the framework by applying it to explore interconnect options in manycore systems with consideration of temperature variation and leakage feedback. We compare power, energy-delay-area product (EDAP) and energy-delay product (EDP) of four manycore configurations-1 core, 2 cores, 4 cores and 8 cores per cluster. Results from simulation with or without consideration of temperature variation both show that the 4-core per cluster configuration has the best EDAP and EDP. Even so, considering that temperature variation increases total power dissipation, we demonstrate the importance of considering temperature variation in the design flow. With this power, area and thermal modeling capability, the SST can be used for hardware/software co-design of future exascale systems.

Keywords: NoC; simulation framework; performance modeling; power consumption

Received 12 December 2010; revised 30 June 2011
Handling editor: Stephen Jarvis

1. INTRODUCTION

With the cost of a megawatt of power for 1 year ranging from $590K to $1.85 Million [1], the power bill for future many-megawatt machines could outweigh their purchase cost. Additionally, increased power consumption often requires expensive new facilities to deliver and manage the vast amount of electricity, as well as facilities to cool the energy-dissipating computers. This is demonstrated by recent data centers constructed by companies like Google and Apple with costs in the hundreds of millions of dollars [2]. The problem is one of worldwide importance as 2010 server power consumption is estimated at $44.5 Billion [3].

Traditional architectural simulators for high performance computing (HPC) narrowly focus on the performance and power dissipation of part of the system. This makes it difficult for system designers to make educated decisions about how to manage the energy for the entire system.

The structural simulation toolkit (SST) [4] aims to address this problem. The SST provides a framework for simulating large-scale HPC systems. This simulator allows parallel simulation of large machines at scale to understand both performance and energy consumption. The SST couples parameterizable models for processors, memory and network subsystems. These models all have access to a uniform interface to power and thermal modeling libraries that estimate power dissipation and update temperature.

This paper introduces the technology interface in the SST, the core of integrated energy, power and temperature simulation. It receives access counts from the SST components and calculates the power dissipation and temperature change. The power and

thermal libraries are architecture modeling tools that provide power and temperature calculation methods to the technology interface. In the current implementation, the power libraries include McPAT [5], IntSim [6, 7] and ORION [8], and HotSpot [9] is used as a thermal library. The technology interface also provides the functionality to dynamically update the leakage power as temperature changes.

Temperature variation has become a major challenge to future HPC architecture design. It has an exponential relationship with leakage, which can cause further exacerbation of power [10]. This problem is becoming more and more critical with the continuous scaling of technology [11]. There have been a few energy management approaches proposed for on-chip networks (NoC), including routing-based energy optimization [12–15] and clustering [5]. In this study, we illustrate the utilization of the modeling framework to study the performance and efficiency tradeoffs in clustering of manycore systems with consideration of temperature variation. The communication pattern generator and the router model components in the SST are used to model the manycore systems and provide component usage counts to the technology interface. The estimated power and performance of the NoC are gathered by the introspection interface for further evaluation.

The remainder of this paper is organized as follows. In Section 2, we discuss prior related works. In Section 3, we describe the structure of the SST and its key interfaces for power/area/thermal modeling and statistics gathering. Section 4 gives more details on the technology models currently supported by the framework, and Section 5 shows preliminary results of validating SST. In Section 6, we use the framework to explore the interconnect options of future manycore processors by varying the degree of clustering, and examine the impact of temperature variation on this. The paper is concluded in Section 7.

2. RELATED WORK

There has been much research in the area of performance and power modeling of HPC architectures. Traditional single-core simulators were designed to model out-of-order superscalar processor pipelines. For instance, SimpleScalar [16] was widely used to evaluate uniprocessor systems. It is extended with several power models, such as Wattch [17] and SimPanalyzer [18], for examining performance/power tradeoffs. However, it runs only user-mode single-threaded workloads, let alone the simulation of multiple processor cores. Full-system simulators are frequently used to evaluate manycore architectures. For example, the M5 simulator [19] supports the execution of the operating system as well as the application code, and is capable of modeling I/O subsystems and multiple networked systems. It has been integrated with Wattch to study the processor lifetime of chip multiprocessors [20]. However, it only supports a shared-bus model to simulate interconnection of manycore processors and the integration with the power model is not available in the standard distribution. GEMS model detailed aspects of cores, cache hierarchy, cache coherence and memory controllers [21] and has an improved network model, GARNET [22]. However, it can only model power dissipation of the network components, not the entire system. The Polaris toolchain provides rapid estimations of power/area/delay of large NoC design space and its power model has been extended to consider temperature variation and leakage feedback [11]. However, it has a lack of detailed power models of the computation and storage components on the chip. The integrated power, area and timing modeling framework, McPAT, has been used together with a manycore performance simulator to explore the interconnect options of future manycore processors, but the simulator is not publicly available. Recently, Lis *et al.* [23] presented HORNET, a parallel, highly configurable, cycle-level manycore simulator with support for power and thermal modeling. However, it does not have a modular design to allow integration of other architectural models not shipped with the package. Moreover, it is integrated with ORION, which only provides power models of network components.

The SST simulation framework builds on a long tradition of architectural and network simulators such as M5 [19], NS-3 [24] and A-SIM [25]. In addition, it builds upon community experiences in modeling power dissipation [5, 17]. The SST often seeks to directly include existing simulators to build a 'best of breed' framework. The novel approach of the SST is to include these individual component models in a parallel, scalable and open-source framework.

3. STRUCTURAL SIMULATION TOOLKIT

The SST is based on a fully modular design and provides a parallel simulation environment using message passing interface (MPI). This enables the SST to extensively explore parameters of an individual system without the need to intrusively change the simulator. It also provides a high level of performance and the capability to look at large systems.

The SST consists of a simulator core, which provides simulation services, and pluggable components (Section 3.4), which constitute individual simulation models. The simulator core provides simulation configuration and start up (Section 3.1), the parallel model of computation (Section 3.2.1), checkpointing (Section 3.3) and a common interface to the technology models (Section 3.5).

3.1. Configuration and job flow

The SST is configured with an XML file which lists the components instantiated in the simulation, any component parameters that must be passed in, the *links* between the components and the latency on the component links.

This configuration is processed into a graph, with the component instances as nodes and the *links* between them as edges, which is then fed to the Zoltan [26] library to find a partition which balances the number of components per host rank and which will maximize the simulated latency between components. Partitioning along high latency *links* means that rank will have to exchange messages less frequently in our conservative optimization.

3.2. Model of computation

The simulation is carried out in a component-based discrete *event* model of computation. Each component can assign a clock to itself, to be triggered at regular intervals. Components can also send *events* to other components along *links*, which have a minimum latency. When an *event* arrives at a component, it triggers an event handler function, in which the component can process and respond to the event. Alternately, the component can poll the *link* to receive and process any outstanding messages.

3.2.1. Parallel implementation

Parallelism is transparent to the component writer. Components interact through sending *events* to each other through *link* objects. All *events* inherit from a common base class, which also includes a time (Section 3.2.2) tag to indicate when it should be delivered. All *events* must be serializable (using the Boost [27] Serialization Library), which transforms the *event* structure into a compact binary representation.

Whenever an *event* is sent, the SST core determines if the destination of the event is local (i.e. on the same MPI rank) or remote. Remote events are queued up for future delivery the next time the given ranks are due to synchronize. This occurs only as often as needed, based upon the latency partitioning given by the Zoltan library; that is, if the components on two ranks are connected by a link with a minimum latency of 1000 ns, those ranks only need to synchronize every 1000 ns of simulated time.

The Boost MPI library is then used to perform the actual communication. When two ranks synchronize, they serialize and send the pending *events* to each other. When the events are received, they are integrated with the local event queues, where they wait for delivery to their target components.

3.2.2. Time

Time in the simulator is represented using a single 64-bit unsigned integer to count the number of atomic timesteps that have passed since the beginning of simulation. The actual atomic timebase (time increment represented by each atomic timestep) is user programmable and has a default of 1 fs (10^{-12}) s, which provides for over 200 days of simulated time. All times used by components and *links* are specified using strings (for example, '1.5 ns' or '1.73 GHz'), and are resolved at build time into a *TimeConverter* object. The *TimeConverter* object essentially represents a component's view of time and provides functions for converting from the component's timebase to the atomic timebase. The *TimeConverter* simply stores the number of atomic timesteps (referred to as its *factor*) in the desired time interval. In the case of a specified clock frequency, the factor represents the number of atomic timesteps in the clock period. For example, a component with a 1 GHz clock would get a *TimeConverter* object with a factor of 1000 (assuming the default atomic timebase of 1 fs), which would also be equal to the factor for 1 ns.

The creation of a *TimeConverter* has two options. The first is to register a clock handler, in which case the handler is called once per clock period. The second is to simply register a timebase with the simulator, which can be used with the event driven interface. In either case the returned *TimeConverter* object is registered with the component's *links*, where it is used to convert latencies from the component's view of time to the atomic timebase. The use of *TimeConverters* insulates the components from both the need to know the value of the atomic timestep, as well as from knowing their own operating frequency. This allows a component to be written with a generic timebase, which can be set at run-time.

3.3. Checkpointing

Because simulations may run for an extended period of time over a number of nodes, the simulator needs the ability to checkpoint and recover its state. To accomplish this, the simulator core uses the Boost Serialization Library to convert the core's state and the state of each component into a binary format. At a user defined interval, this binary state is dumped to a file that can be used to restart the simulation.

3.4. Components

The most important class in the SST is *Component*, the base class from which all simulation components (e.g. core, router, memory, etc.) inherit. Components are connected by *links* to communicate with each other and are partitioned among all ranks to ensure balanced workload and scalability of the simulator.

Components can query the SST's technology interface for power and temperature estimates, and in some cases, chip area and timing information. The *components* can set technology parameters (such as clock frequency and supply voltage), choose the estimation library it wishes to use, and set architecture parameters like cache size or the number of register file ports. Once the estimates are computed, the SST provides *components* a uniform interface (the introspection interface) for reporting power consumption and gathering statistics. The SST includes a variety of processor, network and memory models (*components*) at different scales and levels of accuracy. In this study, we use two components, the communication pattern generator and the router model.

3.4.1. *Communication pattern generator and the Router model*

The communication pattern generator is implemented as a state machine. It simulates compute time by suspending operations until a future event indicates the passing time and the need to transition to another state in the state machine. The communication pattern generators use dimension-ordered routing in the NoC. The only communication pattern implemented at the moment is ghost, which simulates ghost cell exchanges on a five-point stencil operator where each rank communicates only with its east, west, south and north neighbor. Implementations of communication patterns for FFT [28], the NAS parallel benchmark integer sort [29] and master/slave [30] are under way.

The router model component allows for arbitrary topologies. Messages are wormhole routed and use source-based routing. If a path from an input port to an output port is available, the message is forwarded without further delays. The router model also models congestion in the network when the input or the output port is busy. The router model maintains a small number of counters to enable statistics on the number of messages coming in and going out of each port, how often congestion occurs and how much delay is caused.

3.5. Technology interface

The technology interface is the core of power and thermal simulation. It integrates various power and thermal libraries (e.g. McPAT, IntSim, ORION and HotSpot), computes run-time energy dissipation and stores the data traces. Its implementation is independent of the front-end simulator such that the statistical timing model is replaceable.

3.5.1. *The design*

The technology interface is initialized at two levels: the chip level and the component level. The chip-level initialization creates silicon layer floor plans and links the simulation chip with a thermal library. The floor plan may include multiple blocks, and the blocks are assumed to be running with the same clock frequency and voltages, and more importantly within the uniform temperature plane. Therefore, the size of the floor plan determines the modeling granularity of the simulation and can vary from a single block to an SST component (e.g. core or main memory). After floor plans are set up, thermal tiles are created for silicon, interface, spreader and heatsink layers. Thermal tiles are the same size as underlying floor plans but do not need to be on the same layer—only silicon-layer thermal tiles are duplicated in terms of xyz coordination.

Chip-level physical parameters include thermal design parameters, floor plan parameters (e.g. sizing and coordinate information) and technology parameters for each floor plan. Thermal design parameters include chip designs (e.g. chip thickness, thermal RC constants) and the initial temperatures for each thermal tile. Technology parameters include device-level parameters as well as higher-level parameters such as temperature, clock frequency and supply voltage.

The component-level setup reads architectural parameters of each component from the system description language (SDL) file and runs the power model to compute the basic information such as dynamic energy per access, leakage energy and area. Component parameters include core/uncore-level microarchitectural parameters such as number of branch predictors, cache hierarchy, pipeline design, etc.

3.5.2. *Run-time power and temperature estimation with leakage feedback control*

The technology interface specifies the power into five categories: dynamic, short circuit (or switching), gate leakage, subthreshold leakage and peak power. The technology interface gets usage counts of each component at a user-specified period or condition. It then uses the well-known Wattch method [17] to calculate the dynamic energy by multiplying these count values with dynamic energy per access statically calculated by power models. The leakage power consumption is calculated by power models at an initial temperature. The estimated power is then stored in a central power database, which can be accessed for later analysis.

After power dissipation for each component is calculated, the power values for the underlying floor plans are correspondingly updated. Once the power calculation is done for all components in the simulated chip, the technology interface triggers temperature calculation using the thermal library. The thermal library takes instantaneous power values for each floor plan. The new temperature of each floor plan updated by the thermal library is fed back to the blocks grouped within the floor plan. This procedure is called leakage feedback, and the new leakage power is calculated based on the new temperature. These power changes will again affect the temperature profile [31]. After several iterations, the difference between each loop will be small enough and the power and thermal profiles converge [11].

3.6. Introspection interface

The introspection interface is a unified way to report and record simulation data for analysis and display. It provides a standard method of retrieving statistics so that external programs can access the simulator statistics without requiring knowledge of simulator structures. The SST *Introspector* class inherits from the *component* class and can be created to monitor information from all, or a subset of other real components. Like *components*, *introspectors* are created and parameterized by the SDL specifying which components they wish to monitor and how frequent to query these components to retrieve components state (e.g. power). Unlike *components*, *introspectors* are not partitioned and have a copy on every rank. *Introspectors* can exchange components data with introspectors on other ranks

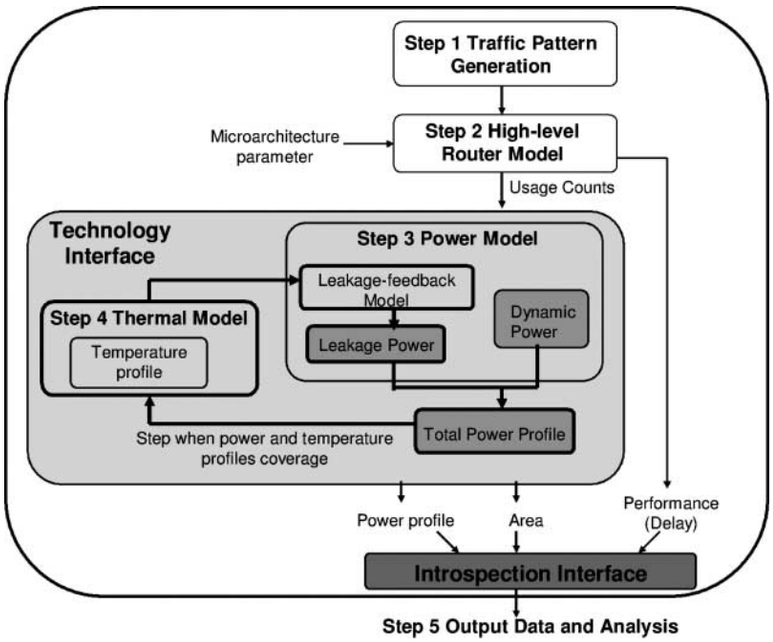

FIGURE 1. Flow chart of NoC power modeling with consideration of leakage feedback.

via MPI collective calls. Via these collective communications, *introspectors* can gather information, such as the highest core temperature in the system and the total power dissipation of the system, etc.

Figure 1 presents the flow chart of NoC power modeling with consideration of the leakage feedback using the modeling framework. The traffic pattern generator component generates a traffic trace, which is then fed into the router model component to capture the usage count of the router and calculate the network delay. The usage count information is then fed into a power library (McPAT or ORION in this study) to calculate the power consumption with consideration of leakage feedback and temperature variations. At this step, within each iteration leakage power is calculated with the simulated temperature until the power and thermal profile have converged. The power library also calculates the area for the NoC. Finally, the introspection interface gathers simulation results including NoC power dissipation, temperature, delay and area from all the router model components. It then uses several metrics to explore interconnect options in manycore processors.

4. TECHNOLOGY MODELS

The following power, area and thermal models are currently integrated with the SST technology interface. They support power/area/thermal analysis of five component types (core, clock, shared cache, memory controller and NoC/router).

4.1. HotSpot

HotSpot is a thermal simulation tool used for estimating chip-wide temperature based on an architectural floor plan and simulated power measurements [9]. The SST technology interface includes the transient thermal simulation kernel of HotSpot, and provides HotSpot with dynamic values for power density and the present temperature of each block. In this paper, we use power densities obtained from McPAT. HotSpot then solves a series of differential equations and returns the new temperature of each block. However, the temperature analysis does not support the case in which power dissipation is dependent on the temperature, which is the situation with leakage [31]. To solve this problem, we extended the thermal analysis such that the power consumption at a timestep is calculated as the sum of two components: (1) the dynamic power return from the power modeling library and (2) the leakage power calculated at the simulated temperature of the previous step.

4.2. McPAT

McPAT is an integrated power, area and timing modeling framework for multithreaded, multicore and manycore processors. It includes power, area and timing models for the processor cores, NoC, shared caches, memory controllers, and clocking. Its power models account for dynamic, subthreshold leakage, gate leakage and short-circuit power. McPAT was validated against Niagara, Niagara2, Alpha 21364 and the Xeon Tulsa

processors. The difference between its models and the reported data was between 10% and 20%.

The original McPAT work [5] assumes uniform temperature within the chip and no leakage feedback. We extended this tool to calculate power consumption with consideration of leakage feedback and temperature variation. Leakage feedback is processed in three steps; updating temperature-dependent technology parameters, reconfiguring McPAT and retrieving the new leakage power.

4.3. IntSim

IntSim is an interconnect-centric CAD tool that optimizes die size and pitches of different wiring levels for circuit blocks or logic cores. It includes a methodology for co-optimization of signal, power and clock interconnects and stochastic wiring distribution [6, 7]. The primary IntSim algorithm is an iterative loop that repeatedly calculates the dynamic and leakage power until the presumed power and the calculated power values converge. This model is useful to evaluate the power of logical blocks such as functional units in the microarchitecture.

4.4. ORION

ORION is a power and area library for different router configurations, which projects potential circuit structures for each configuration at each technology node [8]. ORION was validated to within 7% and 11% error on the Intel 80-core Teraflops and SCC chips, respectively. It takes the SST router model resource utilization information for each router and link as input activity and returns network dynamic and leakage power.

5. VALIDATION

The validation of the SST is composed of three levels: the component level, the component–technology interface level and the component–component level.

The SST often seeks to include existing architectural and network simulators, and provides them a parallel and scalable framework. The simulator developers are responsible for verifying their models (*components*). The SST currently includes cycle-based detailed processor models (e.g. M5), a detailed flit-based router model based on the RedStorm SeaStar router [32] and an abstract message-based router model (Section 3.4.1), a highly detailed memory model (DRAMSim2 [33]) and well-received power/thermal models (McPAT, ORION, HotSpot). Most of these models are validated against real machines.

At the component–technology interface, we verify that components interfacing to power/thermal models are implemented accurately. To validate this, we run two simulations. In one simulation, power and temperature of an NoC are estimated by

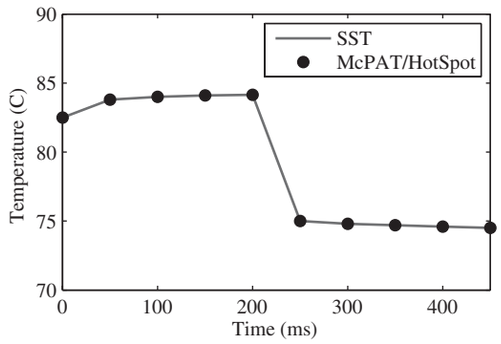

FIGURE 2. Comparison of temperature from the two simulation methodologies.

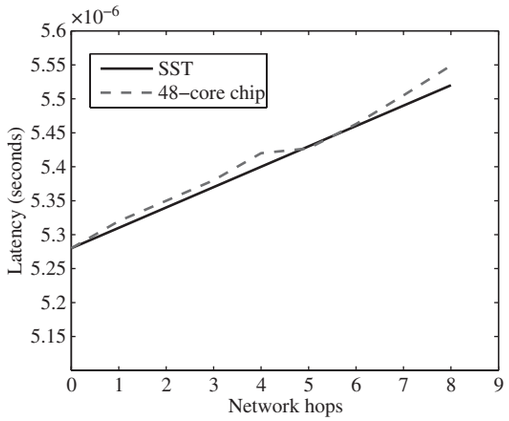

FIGURE 3. Validation against with the Intel 48-core test chip.

McPAT and HotSpot integrated in the SST. In the other simulation, usage information of the NoC is fed into the stand-alone McPAT to compute power. The power trace is then fed into the stand-alone HotSpot to compute temperature. Both simulations are run for 500 ms and power/temperature statistics are gathered every 50 ms. Figure 2 shows that the results from our framework and the direct McPAT/HotSpot output are identical. We do not validate the interface by comparing the results from our framework with power measurements of real machines because McPAT is know to have >50% error in dynamic power modeling.

At the component–component level, we verify the *links* between components, the *events* sending and receiving, and the timing are handled accurately by the SST core. We use the communication pattern generator and the router model to model the Intel 48-core test chip [34] and the Cray Red Storm system installed at Sandia. At the high level, we use the MPI ping-pong latency benchmark to validate the two models. The ping-pong latency is a common network benchmark that helps to insure that basic latencies are correct. As Fig. 3 illustrates, the MPI ping-pong latency matches very well between the simulated and

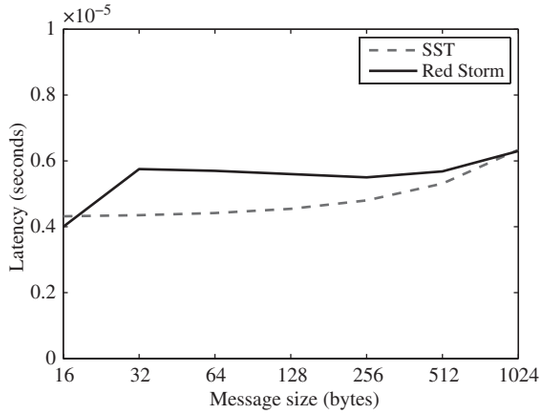

FIGURE 4. Validation against with the Red Storm system.

the 48-core test chip. It is worth noting that the figure shows simple hop delay, and since the ping-pong latency is a very simple benchmark, there is no contention in the network nor randomness in the model, and the result is very deterministic.

Figure 4 shows the ping-pong latency between the simulated and the Red Storm system. The simulation result matches the initial latency of around 4 μs, and the end latency at 1 kB of ∼6 μs. However, it does not show the characteristic bend at 32 bytes. The bend may come from a change in processing the message when it is <32 bytes long, and after it reaches that size. Considering the complexity of the system and our use of generic, coarse-grained components, we consider this validation result to be good. We expect that enhancing the models can mimic the messaging processing behavior.

6. PERFORMANCE ANALYSIS OF VARIOUS INTERCONNECT OPTIONS OF MANYCORE PROCESSORS

We illustrate the utilization of the integrated modeling framework by applying it to evaluate different interconnect options of manycore processors with consideration of the leakage feedback and within-chip temperature variation. We study the power, the energy-delay product (EDP) and the energy-delay-area product (EDAP) for four manycore system configurations. We also examine how temperature variation impacts these metric values.

6.1. Experimental setup

We consider a manycore architecture used in McPAT's work for our baseline system and analysis. The manycore architecture consists of multiple clusters connected by a 2D-mesh on-chip network. A cluster has one or more cores and there is one communication pattern generator on each core. Routers in the network have local ports that connect to a cluster of cores as well as ports that connect to the neighboring routers. The cores attached to the same router in the NoC are assumed to have a shared L2 cache. Messages between them carry a flag indicating that they should be handled at a lower latency and higher bandwidth by the router model.

We adopt parameters from McPAT and our experiments assume 64-core NoC designs and 22 nm processing technology. Our study considers four configurations: 1 core, 2 cores, 4 cores and 8 cores per cluster. Simulations are run with a time step of 150 000 ns and the NoC power is estimated every 100 timesteps. The simulation overhead highly depends on the components used. In this study, we use the high-level communication pattern generator and the router model, which incur little processing and memory overhead. The SST startup takes less than a second. Simulations with power/thermal modeling result in ∼10% overhead, mainly due to the computation performed in McPAT and HotSpot.

6.2. Comparison of router power estimated by McPAT and ORION

The two power models, McPAT and ORION, supported by the modeling framework are popularly used in many power-related architecture research projects. First, we want to compare them

TABLE 1. Parameter values.

Technology	65 nm
Clock frequency	1.0 GHz
Supply voltage	1.2 V
Number of input ports	8
Number of output ports	8
Flit bits	128
Virtual Channels per port	2

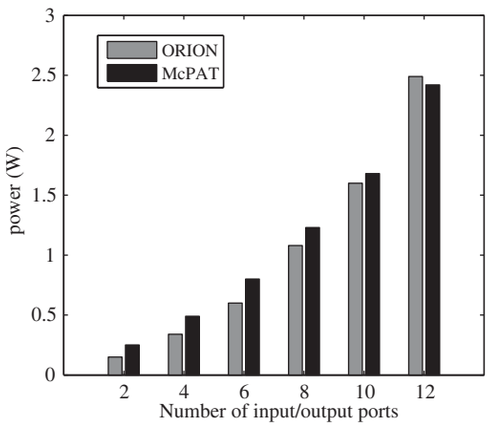

FIGURE 5. Comparison of McPAT and ORION in estimation of router power with varying number of ports.

in terms of NoC power modeling. We use the parameters in Table 1 to model router power dissipation by both McPAT and ORION and the results are shown in Figs 5 and 6.

In Fig. 5, the number of ports is varied from 2 to 12 while the rest of the parameters are kept the same. In Fig. 6, the flit width is varied from 8 bits to 256 bits. The figures show that the discrepancy between McPAT's and ORION's power estimations is small (an average of 10%), and they both grow at about the same rate when the number of ports and flit bits are varied.

Each of the two power modeling tools has its advantage over the other. For example, McPAT also provides power models for core, memory controller and shared cache. One can simply call McPAT to model the overall power dissipation of manycore systems. On the other hand, ORION's NoC parameters are much more customizable than McPAT's and are more flexible to model various router architectures. In addition, ORION is a modeling tool specifically for NoC so it takes less time to initialize while McPAT may have a huge design space to exhaust the search for the best value power and area deviation.

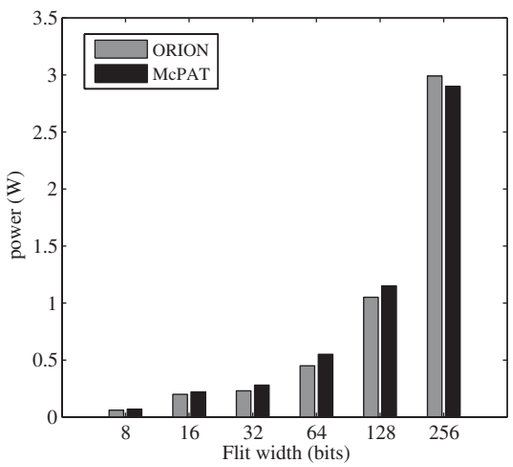

FIGURE 6. Comparison of McPAT and ORION in estimation of router power with varying flit width.

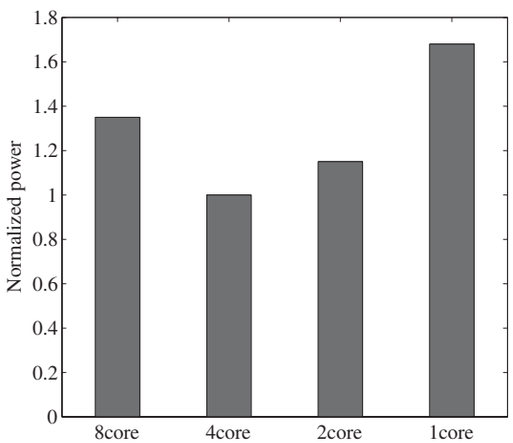

FIGURE 7. Normalized power of the manycore systems.

6.3. Performance and efficiency tradeoff in clustering

Figure 7 shows the total power of the four configurations (1-core, 2-core, 4-core and 8-core per cluster) normalized by the value of the 4-core per cluster configuration. In these simulations, we use McPAT to model the NoC power dissipation

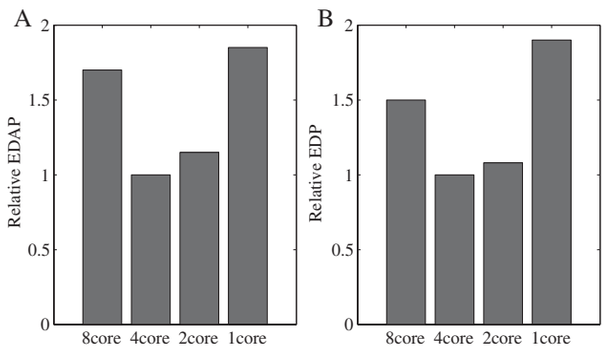

FIGURE 8. Normalized EDP and EDAP.

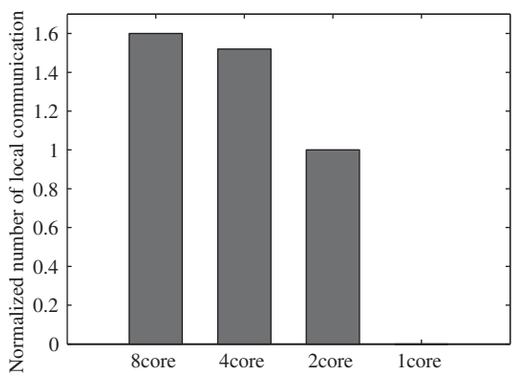

FIGURE 9. Normalized number of intra-cluster communication.

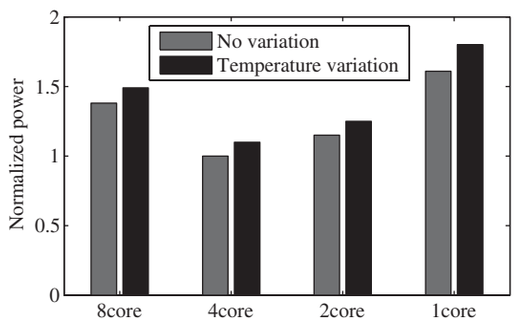

FIGURE 10. Normalized power with consideration of temperature variation.

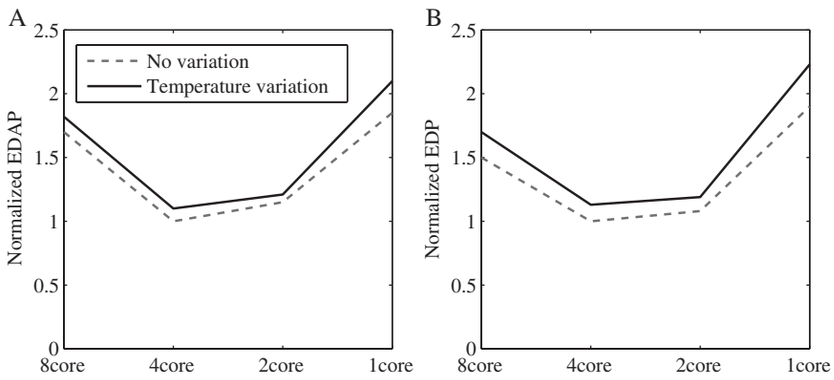

FIGURE 11. Normalized EDP and EDAP with consideration of temperature variation.

with neither leakage feedback nor temperature variation. In general, clustering reduces the system total power except at the 8-core per cluster configuration.

We use two metrics to evaluate the performance and efficiency tradeoffs in clustering. The EDAP is of particular interest because the metric includes both an operational cost element (energy) and a capital cost element (area) [5]. Figure 8 shows EDAP and EDP of the four system configurations normalized by the values of the 4-core per cluster configuration. Figure 8(A) shows that clustering using 4 cores gives the best EDAP. This is consistent with McPAT's conclusion that the 4-core per cluster configuration has the best EDAP on all benchmark suites on average [5]. Figure 8(B) shows the 4-core per cluster configuration also has the best EDP.

It has been shown that, on average, increasing the cluster size improves the system EDP and the effects of clustering on the metric values depend heavily on applications [5]. In our study, the 8-core per cluster configuration has the worse EDP than the 4-core per cluster design. This is because of the characteristics of the communication pattern generator we use. In this study, we assume that the cores in the same cluster have a shared L2 cache. Figure 9 shows that the communication pattern generator results in similar numbers of local communications (intra-cluster communications) for the 4- and the 8-core per cluster configurations. Therefore, clustering 8 cores together does not derive more benefit from cache sharing compared with the 4-core per cluster design. On the other hand, the 2-core per cluster design has ∼50% less local communications and thus consumes more power in routing messages.

6.4. Effect of temperature variation and leakage feedback on performance

In the previous section, we simulate the network on chip with a single uniform temperature and without leakage feedback. We now consider temperature variation and leakage feedback in the model and examine how these affect the metric values for the four system configurations.

Figure 10 shows the total power consumption of the four configurations normalized to the lowest power NoC configuration. The grey bars indicate estimated NoC power with no leakage feedback or temperature variation, while the black bars show the estimated power taking both into consideration. The figure shows that when both leakage feedback and temperature variation are considered, the power consumption for each configuration increases (by ∼10%) compared with the no variation case. Besides, the relative power ranking among the four configurations remains the same as the one with no variation.

Next, we examine the impact of considering both leakage feedback and temperature variation on EDAP and EDP. Figure 11 shows the normalized EDAP and EDP of the four configurations with (solid line) and without variance (dashed line). We can see that considering temperature variation does not change the relative ranking of EDAP or EDP among the four configurations. The figure gives the same conclusion that the 4-core per cluster configuration has the best EDAP and EDP. However, it is still important to consider the temperature variation in NoC power modeling because it can introduce substantial differences in total power dissipation. Even though the relative power ranking does not change, a design may have a power constraint that needs to be met. Therefore, total power should be accurately estimated in order to make a correct design decision.

7. CONCLUSIONS

In this work, we implement a framework for integrated power, area and thermal simulation and data gathering. The framework includes a number of power and thermal models and contains leakage feedback functionality. We illustrate the utilization of the framework on studying performance and efficiency tradeoffs in the clustering of manycore processors with consideration

of within-chip temperature variation and leakage feedback. Integration of this power, area and temperature estimation capability to the SST for HPC architectural simulations makes it possible to use the SST for hardware/software co-design of future exascale systems.

We first compare the two power models, McPAT and ORION, in NoC power estimation. Our result suggests that there is a small discrepancy between the two in terms of power dissipation. We then use a manycore architecture in McPAT's work for our baseline system and analysis. While McPAT uses a full simulator to model a core, the communication pattern generator in the SST allows to generate network traffic without incurring the processing and memory overhead. We arrive at the same conclusion as McPAT's that configuring a cluster with 4 cores has the best EDAP. When temperature variation and leakage feedback are taken into consideration, the conclusion remains the same. Even so, considering the temperature variation in power estimation is still important because it substantially increases the power dissipation, which can change a designer's decision in choosing a network configuration. Moreover, it has been shown that the amount of performance gained by the use of a manycore processor depends very much on the application. When more communication pattern generators are implemented, we expect that our simulation results would demonstrate the impact of communication patterns on power, EDAP and EDP in more detail.

ACKNOWLEDGEMENTS

Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

FUNDING

This project is supported by the DOE NNSA Advanced Simulation and Computing program, and the DOE Office of Advanced Scientific Computing.

REFERENCES

[1] Outlook, A. (2009) *Average retail price of electricity to ultimate customers.* http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_b.html (accessed August 13, 2010).

[2] Register (2009). http://www.theregister.co.uk/2009/05/26/new_apple_data_center/ (accessed August 13, 2010).

[3] Scaramella, J. (2006) *Worldwide server power and cooling expense 2006–2010 forecast.* http://www.sun.com/service/eco/IDCWorldwideServerPowerConsumption.pdf (accessed August 13, 2010).

[4] Rodrigues, A., Hemmert, K., Barrett, B., Kersey, C., Oldfield, R., Weston, M., Risen, R., Cook, J., Rosenfeld, P., CooperBalls, E. *et al.* (2011) The structural simulation toolkit. *ACM SIGMETRICS Perform. Eval. Rev.*, **38**, 37–42.

[5] Li, S., Ahn, J., Strong, R., Brockman, J., Tullsen, D. and Jouppi, N. (2009) Mcpat: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. *Proc. 42nd Annual IEEE/ACM Int. Symp. Microarchitecture*, New York, NY, December 12–16, pp. 469–480. ACM.

[6] Sekar, D. (2008) Optimal signal, power, clock and thermal interconnect networks for high-performance 2d and 3d integrated circuits. Ph. D. Dissertation, School of ECE, Georgia Tech.

[7] Sekar, D., Naeemi, A., Sarvari, R., Davis, J. and Meindl, J. (2007) Intsim: A Cad Tool for Optimization of Multilevel Interconnect Networks. *Proc. 2007 IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, November 5–8, pp. 560–567. IEEE Press.

[8] Singh, S., Mayfield, C., Mittal, S., Prabhakar, S., Hambrusch, S. and Shah, R. (2008) Orion 2.0: Native Support for Uncertain Data. *Proc. 2008 ACM SIGMOD Int. Conf. Management of Data*, Vancouver, BC, Canada, June 10–12, pp. 1239–1242. ACM.

[9] Skadron, K., Stan, M., Sankaranarayanan, K., Huang, W., Velusamy, S. and Tarjan, D. (2004) Temperature-aware microarchitecture: modeling and implementation. *ACM Trans. Archit. Code Optim. (TACO)*, **1**, 94–125.

[10] Mohanty, S., Ranganathan, N., Kougianos, E. and Patra, P. (2008) *Low-Power High-level Synthesis for Nanoscale CMOS Circuits*. Springer.

[11] Li, B., Peh, L. and Patra, P. (2008) Impact of Process and Temperature Variations on Network-on-Chip Design Exploration. *Proc. 2nd ACM/IEEE Int. Symp. Networks-on-Chip*, Newcastle University, UK, April 7–11, pp. 117–126. IEEE Computer Society.

[12] Gelenbe, E. and Morfopoulou, C. (2011) Routing and g-networks to optimise energy and quality of service in packet networks. *Energy-Efficient Comput. Netw.*, **54**, 163–173.

[13] Gelenbe, E. and Mahmoodi, T. (2011) Energy-Aware Routing in the Cognitive Packet Network. *Int. Conf. Smart Grids, Green Communications, and IT Energy-aware Technologies (Energy'11)*, Venice, Italy, May 22–27.

[14] Gelenbe, E. and Morfopoulou, C. (2010) A framework for energy-aware routing in packet networks. *Comput. J.*, **54**, 850.

[15] Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M. and Pentikousis, K. (2010) Energy-efficient cloud computing. *Comput. J.*, **53**, 1045.

[16] Burger, D. and Austin, T. (1997) The simplescalar tool set, version 2.0. *ACM SIGARCH Comput. Archit. News*, **25**, 13–25.

[17] Brooks, D., Tiwari, V. and Martonosi, M. (2000) Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *ACM SIGARCH Computer Architecture News*, *The 27th Annual International Symposium on Computer Architecture*. June 10–14, 2000. Vancouver, British Columbia, Canada. pp. 83–94. ACM.

[18] Kim, N.S. (2003) *Sim-panalyzer library*. http://www.eecs.umich.edu/~panalyzer/ (accessed June 1, 2009).

[19] Binkert, N., Dreslinski, R., Hsu, L., Lim, K., Saidi, A. and Reinhardt, S. (2006) The m5 simulator: modeling networked systems. *Micro, IEEE*, **26**, 52–60.

[20] Coskun, A., Strong, R., Tullsen, D. and Simunic Rosing, T. (2009) Evaluating the Impact of Job Scheduling and Power Management on Processor Lifetime for Chip Multiprocessors. *Proc. 11th Int. Joint Conf. Measurement and Modeling of Computer Systems*, Seattle, WA, June 15–19, pp. 169–180. ACM.

[21] Martin, M., Sorin, D., Beckmann, B., Marty, M., Xu, M., Alameldeen, A., Moore, K., Hill, M. and Wood, D. (2005) Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *ACM SIGARCH Comput. Archit. News*, **33**, 92–99.

[22] Agarwal, N., Krishna, T., Peh, L. and Jha, N. (2009) Garnet: A Detailed On-Chip Network Model Inside a Full-System Simulator. *IEEE Int. Symp. Performance Analysis of Systems and Software, 2009. ISPASS 2009*, Boston, MA, April 26–28, pp. 33–42. IEEE.

[23] Lis, M., Ren, P., Cho, M., Shim, K., Fletcher, C., Khan, O. and Devadas, S. (2011) Scalable, Accurate Multicore Simulation in the 1000-core Era. *Int. Symp. Performance Analysis of Systems and Software (ISPASS 2011)*, Austin, TX, April 10–12, pp. 175–185.

[24] Henderson, T., Roy, S., Floyd, S. and Riley, G. (2006) ns-3 Project Goals. *Proc. from the 2006 Workshop on ns-2: The IP Network Simulator*, Pisa, Italy, October 10, pp. 13–es. ACM.

[25] Nellans, D., Kadaru, V. and Brunvand, E. (2004) Asim-an Asynchronous Architectural Level Simulator. *Proc. GLSVLSI*, Boston, MA, April 26–28, pp. 73–77. Citeseer.

[26] Devine, K., Boman, E., Heaphy, R., Hendrickson, B. and Vaughan, C. (2002) Zoltan data management services for parallel dynamic applications. *Comput. Sci. Eng.*, **4**, 90–96.

[27] Karlsson, B. (2005) *Beyond the C++ Standard Library*. Addison-Wesley Professional.

[28] Swarztrauber, P., Sweet, R., Briggs, W., Henson, V. and Otto, J. (1991) Bluestein's fft for arbitrary n on the hypercube. *Parallel Comput.*, **17**, 607–617.

[29] Prakash, S., Deelman, E. and Bagrodia, R. (2000) Asynchronous parallel simulation of parallel programs. *IEEE Trans. Softw. Eng.*, **26**, 385–400.

[30] Leary, G., Mehta, K. and Chatha, K. (2007) Performance and Resource Optimization of noc Router Architecture for Master and Slave ip Cores. *Proc. 5th IEEE/ACM Int. Conf. Hardware/Software Codesign and System Synthesis*, Salzburg, Austria, September 30–October 5, pp. 155–160. ACM.

[31] Su, H., Liu, F., Devgan, A., Acar, E. and Nassif, S. (2003) Full Chip Leakage Estimation Considering Power Supply and Temperature Variations. *Proc. 2003 Int. Symp. Low Power Electronics and Design*, Seoul, Korea, August 25–27, pp. 78–83. ACM.

[32] Underwood, K., Levenhagen, M. and Rodrigues, A. (2007) Simulating Red Storm: Challenges and Successes in Building a System Simulation. *2007 IEEE Int. Parallel and Distributed Processing Symp.*, Long Beach, CA, March 26–30, p. 45. IEEE.

[33] Rosenfeld, P. (2010) Dramsim2. http://www.ece.umd.edu/dramsim (accessed August 20, 2010).

[34] Mattson, T., Riepen, M., Lehnig, T., Brett, P., Haas, W., Kennedy, P., Howard, J., Vangal, S., Borkar, N., Ruhl, G. *et al.* (2010) The 48-Core scc Processor: The Programmer's View. *Proc. 2010 ACM/IEEE Int. Conf. High Performance Computing, Networking, Storage and Analysis*, New Orleans, LA, November 13–19, pp. 1–11. IEEE Computer Society.