Temperature-Aware On-Chip Networks

On-chip networks are becoming increasingly popular as a way to connect high-performance single-chip computer systems, but thermal issues greatly limit network design. This thermal modeling and simulation framework combines with a distributed runtime scheme for thermal management to offer a path to thermally efficient on-chip network design.

Li Shang
Queen's University

Li-Shiuan Peh
Amit Kumar
Niraj K. Jha
Princeton University

•••••• In the design of high-performance computer systems, power and temperature have become dominant constraints. Increased power consumption can raise chip temperature, which in turn can decrease chip reliability and performance and increase cooling cost. Circuit reliability depends exponentially on operating temperature, with temperature variations and hotspots accounting for most electronic failures.[1,2] Thermal variations can also lead to significant timing uncertainty, prompting designers to opt for wider timing margins that degrade performance. According to the International Technology Road Map for Semiconductors,[3] power and thermal issues will still be limiting factors in future system scaling.

High-performance computer systems are moving towards the use of application-specific multiprocessor system-on-a-chip (MPSoC) and general-purpose chip-multiprocessor (CMP). Due to the stringent wire delay constraints and increasing demand for on-chip bandwidth, networks are becoming the pervasive on-chip interconnect fabric.[4-6]

Although networks consume a significant part of the chip's power budget[7,8]—greatly affecting overall chip temperature—there is little research on how they contribute to thermal issues. Unlike centralized microprocessors, networks are distributed, as are on-chip systems, a characteristic that imposes unique requirements on thermal analysis and management. We believe that addressing the thermal issues of an on-chip network can provide valuable insights that would lead to the design of temperature-aware on-chip systems.

To that end, we have developed a complete thermal analysis and management solution that targets thermal efficiency in on-chip network design. Our solution has two components:

- *Sirius* is an integrated thermal modeling and simulation framework that lets designers rapidly evaluate runtime performance, power consumption, and the thermal profile of an on-chip network design.
- *ThermalHerd* is a distributed runtime scheme for thermal management that lets routers collaboratively regulate the network temperature profile and work to

Published by the IEEE Computer Society

0272-1732/06/$20.00 © 2006 IEEE

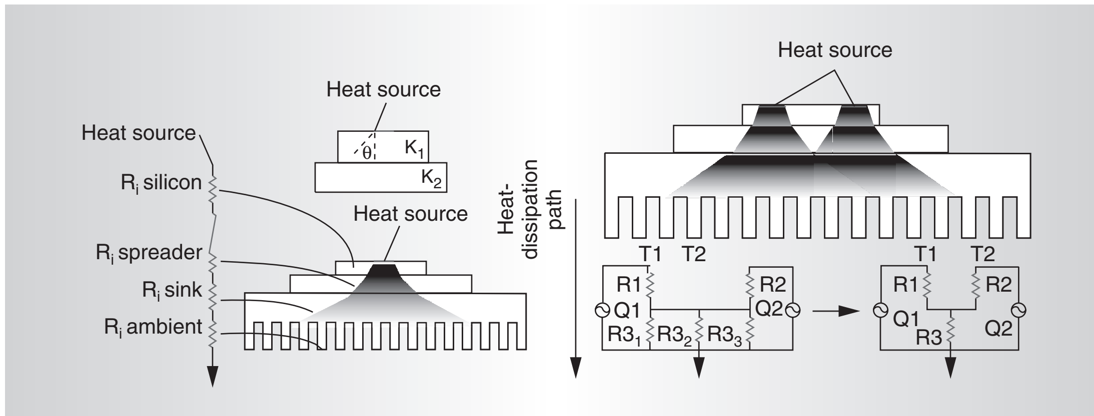

Figure 1. Thermal modeling of an on-chip router. Modeling is based on the idea of heat-spreading angle, which lets designers accurately characterize the package's heat-dissipation path.

avert thermal emergencies while minimizing performance impact.

Performance evaluation demonstrated that ThermalHerd can effectively eliminate runtime network thermal emergencies with a negligible performance penalty. Using MIT's Raw CMP[6] as a case study, we considered the thermal requirements of an entire on-chip system—network, processing, and communication elements. This evaluation provided insights that suggested several extensions to ThermalHerd that might serve as the foundation for a temperature-aware on-chip system.

We believe our work will benefit architects and designers who are seeking a way to quickly explore the design space for the entire on-chip system early on without rigorous circuit-level analysis or joint dynamic thermal-management techniques that target processors and networks.

Compiler writers and application developers can use our results to study and optimize the thermal effects of future application workloads on future CMP platforms. Finally, for operating system developers, our work demonstrates the feasibility of systemwide dynamic thermal management at the operating system level.

Thermal modeling and simulation

Sirius lets designers rapidly explore the performance, power consumption, and thermal profile of on-chip networks. It comprises three models: The *on-chip network* model specifies the technology, topology, and resource configuration of the network design. The *power* model, which we adapted from Orion,[8] characterizes the network's dynamic and leakage power consumption. The *thermal* model, an architectural chip-package model, provides both static and dynamic thermal characterization.

As a trace-driven simulator, Sirius gathers traffic activities and feeds them into the power model to estimate network power consumption. It then periodically feeds the estimated network power distribution to the thermal model to estimate the network temperature profile. Finally, it gathers timing information to monitor network latency and throughput.

Thermal modeling of an on-chip network

The on-chip network model provides a detailed thermal characterization of two main network resources, routers and link circuitry.

Routers. In an on-chip network, each router's power consumption as well as that of neighboring and remote routers affects that router's temperature. Because each router's power and thermal impact could be limited, inter-router thermal correlation plays an important role in shaping the chip's overall temperature profile. Understanding the network's thermal behavior thus requires accurately characterizing spatial thermal correlation. In microelectronic packages, heat flow from the silicon surface to the ambient environment is three-dimensional—heat dissipates both vertically and horizontally, as Figure 1 shows.

Such heat spreading directly affects the on-chip router's thermal resistance and inter-router

thermal correlation. In Sirius, we base the thermal modeling of an on-chip router on the notion of a heat-spreading angle—the angle at which heat dissipates through packaging layers. We estimate the heat-spreading angle as $\theta = \tan^{-1}(k_1/k_2)$, where k_1 is the thermal conductivity of the current packaging layer's material, and k_2 is the thermal conductivity of the underlying packaging layer's material.[9] Using the heat-spreading angle, we can accurately characterize the effective heat dissipation path in microelectronic packages. For example, thermal resistance R of a rectangular heat source on a carrier that includes heat spreading is[10]

$$R = \frac{1}{2k\tan\theta(x-y)} \ln \frac{y+2L\tan\theta}{x+2L\tan\theta} \frac{x}{y} \quad (1)$$

where x and y are the length and width of the heat source, L is the height of the carrier, and k is the thermal conductivity.

As Figure 1 shows, for each on-chip router, the heat generated from the active device layer dissipates through the silicon die, heat spreader, and heat sink to the ambient environment. The thermal resistance of each router i is R_i, the summation of the thermal resistance of each thermal component in the chip package along the heat dissipation path:

$$R_i = R_i \text{ silicon} + R_i \text{ spreader} + R_i \text{ sink} + R_i \text{ ambient} \quad (2)$$

where R_i silicon, R_i spreader, R_i sink, and R_i ambient are the thermal resistances of the on-chip router through the silicon die, heat spreader, heat sink, and ambient environment.

Equation 1 can determine both R_i silicon and R_i spreader. For R_i silicon, the area of the heat source is the size of the on-chip router, and the heat-spreading angle in the silicon die is the thermal conductivity ratio of the silicon die and the heat spreader. For R_i spreader, the area of the heat source is the original router area plus the area expansion from heat spreading in the silicon die—$(x + 2L_{\text{silicon}} \tan\theta_{\text{silicon}})(y + 2L_{\text{silicon}} \tan\theta_{\text{silicon}})$. The thermal conductivity ratio of the heat spreader and heat sink affects the heat-spreading angle.

For R_i sink, we attach the other side of the heat sink to a cooling fan and base the heat dissipation on heat convection. Some researchers have proposed a closed-form thermal equation to address the heat-spreading issue in heat sinks:[11]

$$R_{i\text{sink}} = \frac{1}{\pi k \alpha} \left(\varepsilon\tau + (1-\varepsilon) \frac{\tanh(\lambda\tau) + \frac{\lambda}{B_i}}{1 + \frac{\lambda}{B_i}\tanh(\lambda\tau)} \right) \quad (3)$$

where k is the thermal conductivity of the heat sink; α is the source radius; ε, τ, and λ are dimensionless parameters;[11] and B_i is the Biot number—the ratio of the thermal resistance of the solid by conduction and the fluid by convection.

We then use this equation to estimate thermal resistance along the heat convection path:[12]

$$R_{i\text{ambient}} = \frac{1}{h_c A_s} \quad (4)$$

where h_c is the convection heat-transfer coefficient and A_s is the effective surface area. Sirius characterizes thermal correlation among on-chip routers on the basis of the cooling structure, heat-spreading angle in each cooling package layer, and inter-router distance. A duality exists between heat transfer and electrical phenomena. To analyze the inter-router thermal effect, we extended the linearized superposition principle in electrical circuits to thermal circuits. In Figure 1, $Q1$ and $Q2$ denote the power consumption of two on-chip routers. The corresponding heat dissipation paths of these routers are initially separate but will finally merge because of the heat-spreading effect.

Using the linearized superposition principle, for these two routers, Sirius divides the heat-dissipation paths into two parts from the merged point. Before this point, it models the paths with two thermal resistors, $R1$ and $R2$. After this point, it models the paths with a shared thermal resistor, $R3$. The thermal correlation between two heat sources is the value of $R3$, which is the thermal resistance of the shared heat-dissipation path from where the two heat dissipation paths merge to the ambi-

ent environment. The position of the merged point is based on the heat-spreading angle in each packaging material and the inter-router distance.

We first validated our thermal model for routers against FEMLAB, a commercial multiphysics modeling package. For synthetic test cases, the average error was only 1 percent. We then validated the model against an IBM internal finite-element based thermal simulator. On an actual chip design from IBM, the average error was 5.3 percent.

Links. Increased current densities and associated thermal effects are due in part to aggressive interconnect scaling. In on-chip networks, the high-speed link circuitry that connects routers contributes significantly to chip temperature in both the silicon and metal layers. Because on-chip links are typically long, designers insert buffers to reduce the signal-propagation delay. These buffers affect not only network performance and power consumption, but also temperature. Buffers split on-chip links into multiple segments, with each segment connected to silicon through two vias. In copper processes, vias have much better thermal conductivity than the dielectric and thus serve as efficient heat-dissipation paths. An effective model of on-chip link temperature must thus consider the effects of buffer insertion. Sirius uses a previously derived equation to calculate the optimal interconnect length at which to insert buffers:[13]

$$l_{\text{opt}} = \text{const}\sqrt{\frac{r_0\left(c_0 + c_p\right)}{rc}} \tag{5}$$

where r_0 and c_0 are the effective driver resistance and input capacitance for a minimum-sized driver, c_p is the output parasitic capacitance, and r and c are the interconnect resistance and capacitance per unit length.

Given the length of link segments, the temperature along each segment is[14]

$$T(x) = T_0 + \frac{j^2 \rho L_H^2}{k_M}\left(1 - \frac{\cosh\left(\frac{x}{L_H}\right)}{\cosh\left(\frac{L}{2L_H}\right)}\right) \tag{6}$$

where T_0 is the underlying layer temperature; j is the current density through the link segment; ρ, L, and k_M are the link segment's resistivity, length, and thermal conductivity; and L_H is the thermal characteristic length.

After analyzing the secondary heat-dissipation path from top silicon dioxide and low-k material layers to the printed circuit board, we found that the link circuitry's major thermal contribution is in the silicon (buffers). This is primarily because the self-heating power of metal wires is limited. Thermal hotspots are thus in the silicon, not the metal layers.

Thermal management

Thermal management in ThermalHerd consists of four key functions: temperature and traffic monitoring (including prediction), distributed traffic throttling, and thermal-correlation-based traffic routing.

Temperature and traffic monitoring

At runtime, temperature monitors, such as thermal sensors or online thermal models, periodically report the local temperature to each router, triggering an emergency mode when the local temperature exceeds a thermal threshold.

To monitor and predict local and neighboring traffic, ThermalHerd relies on the two sets of traffic-activity counters embedded in each router. A weighted-average filtering technique eliminates transient traffic fluctuations, which lets ThermalHerd characterize temperature-related long-term traffic trends.

Distributed traffic throttling

When thermal emergencies occur, the network must throttle incoming traffic, reducing power consumption and thus network temperature. The key challenge in designing a distributed traffic-throttling mechanism is to effectively regulate overall network temperature to avert thermal emergencies with as small a performance impact as possible.

Alleviating the thermal emergency without incurring a large performance penalty requires throttling traffic around the strongest thermal hotspots. As we described earlier, inter-router thermal correlation is a function of the cooling structure and inter-router distance. Our detailed thermal analysis revealed that a network's thermal hotspots strongly correlate

with local traffic—that is, neighboring routers have a greater thermal impact than remote ones. By reducing the traffic selectively and targeting spots that contribute the most to the temperature increase, ThermalHerd can effectively minimize the performance impact.

When a router detects a thermal hotspot, it begins to decrease the local workload by throttling the input traffic. ThermalHerd uses exponential factor k and a local traffic estimation to control the traffic-throttling policy; namely,

$$\text{Quota}_i = k \times (N_{\text{his_local}} + N_{\text{his_neighbor}});\ k \leq 1 \quad (7)$$

Where $N_{\text{his_local}}$ is the estimated network traffic workload injected locally. $N_{\text{his_neighbor}}$ is the estimated network traffic workload arriving from the neighborhood. Quota_i is the traffic quota to control the total workload permitted to pass through this router.

At the beginning of each thermal timing window, ThermalHerd reports a new temperature. If the temperature continues to increase in the next timing window, ThermalHerd multiplies the traffic quota by k; otherwise, it maintains the throttling ratio. Thus, the overall traffic-throttling ratio, K, equals k^n, where n is the number of thermal timing windows in which the temperature continuously increases. This procedure continues until the thermal emergency is over or until K reaches predefined lower-bound K_L. Each router also splits Quota_i between the traffic injected locally, $\text{Quota}_{\text{local}}$, and the traffic arriving from the neighborhood, $\text{Quota}_{\text{neighbor}}$:

$$\begin{aligned} \text{Quota}_{\text{local}} &= N_{\text{his_local}} \text{ if } N_{\text{his_local}} \leq \text{Quota}_i \\ \text{Quota}_{\text{local}} &= \text{Quota}_i \text{ if } N_{\text{his_local}} > \text{Quota}_i \end{aligned} \quad (8)$$

$$\text{Quota}_{\text{neighbor}} = \text{Quota}_i - N_{\text{his_local}} \text{ if positive; } 0 \text{ otherwise}$$

This policy is biased toward providing enough traffic quota to the locally generated traffic. The rationale behind the policy is straightforward: Without enough traffic quota, the locally generated traffic will be blocked in the injection buffer, which increases network latency. Traffic from neighboring routers, on the other hand, can be redirected through other paths, thus avoiding a performance penalty.

Thermal-correlation-based traffic routing

Traffic throttling achieves a lower junction temperature by reducing network traffic and power consumption. Distributed throttling is efficient, but it does carry a performance penalty. Network thermal hotspots are often a result of traffic imbalance, so temperature-aware routing algorithms must be able to redirect traffic away from throttled routers to minimize the performance penalty and then shape network traffic patterns suitably. Such algorithms could balance the network temperature profile and avoid or reduce traffic throttling.

In ThermalHerd, each router uses both proactive and reactive routing techniques based on thermal correlation. The *proactive* routing scheme continuously monitors the network's temperature profile. When the maximum chip temperature is below the thermal emergency limit, the proactive routing scheme dynamically adjusts traffic to balance the network temperature profile and reduce peak temperature. When a router receives thermal emergency information, its *reactive routing* protocol replaces its proactive routing scheme and then tries to steer packets away from throttled regions, which helps balance the network's temperature profile to reduce the hotspot temperature and eliminate the need for throttling. To balance the network temperature profile, both routing schemes choose paths that have the least thermal correlation with the hottest regions.

A detailed thermal analysis shows that inter-router thermal correlation dramatically decreases as inter-router distance increases, and that thermal correlation in remote routers is typically very small. However, the paths that the routers choose are not necessarily those farthest from the hottest boundaries. Choosing only the farthest paths significantly reduces path diversity and pushes traffic workload toward the coolest boundaries. The result can be an unbalanced traffic distribution that introduces new thermal hotspots. For these reasons, rather than simply defaulting to the paths with the least thermal correlation, Ther-

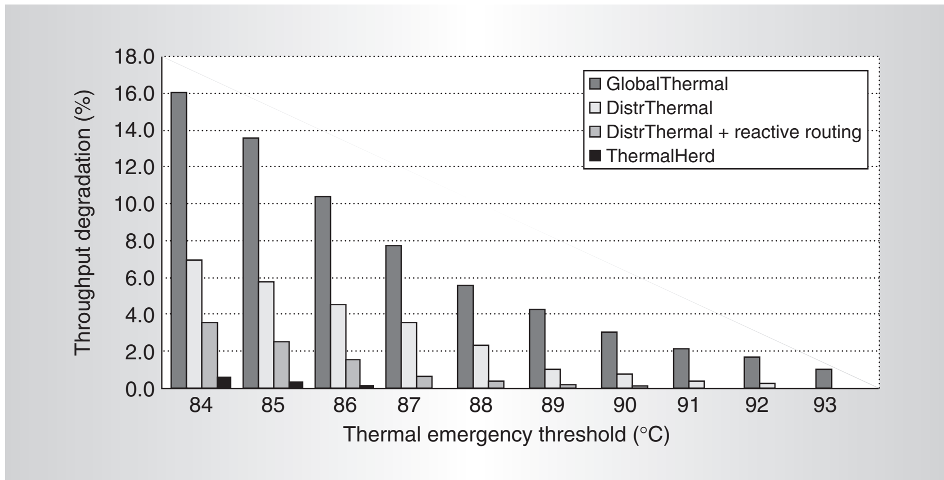

Figure 2. ThermalHerd performance evaluation. ThermalHerd incorporates proactive and reactive routing plus distributed traffic throttling. Throughput degradation is consistently lower than that of the other thermal-management options.

malHerd uses a thermal-correlation threshold, β, as the criterion for selecting routing-path candidates.

Intuitively, ThermalHerd's routing protocol jointly considers thermal correlation and traffic balancing. It aims to eliminate routing paths with a high thermal correlation while leaving enough alternative routing candidates to balance the network traffic. To meet this goal, it picks paths where the thermal correlation between the source and every hop along the path is below β. If no routing candidates meet this criterion, ThermalHerd chooses a candidate routing path with the least thermal correlation.

To strike a good balance between temperature and performance, proactive and reactive routing policies use different thermal-correlation thresholds. When the maximum network temperature is below the thermal emergency limit, to minimize performance penalty, proactive routing uses a less aggressive traffic-redirect policy (a threshold of $2L$ in the work we are describing, in which L is the physical distance between neighboring routers). Reactive routing uses a threshold of $4L$, since during a thermal emergency, reducing the chip temperature is the first-order issue, and most of the performance penalty comes from traffic throttling.

ThermalHerd evaluation

With Sirius as the simulation platform, we evaluated the performance of ThermalHerd using traffic traces that we extracted from the on-chip operand networks of the Tera-Op Reliable Intelligently Adaptive Processing System (TRIPS) CMP by running a suite of 16 SPEC and Mediabench benchmarks.

We designed the evaluation to focus on two major design metrics:

- *Effectiveness of runtime thermal management.* First and foremost, as an online thermal management scheme, ThermalHerd must effectively alleviate thermal emergencies and ensure safe online operation.
- *Impact on network performance.* On-chip networks have very tight latency and throughput requirements, so the performance impact of thermal management must be minimal.

Our evaluation demonstrated that ThermalHerd is both effective and efficient. It regulated network runtime temperature, guaranteed safe online operations, and reduced network peak temperature by 10°C with negligible performance penalty—throughput degradation of less than 1 percent and latency overhead of less than 1.2 percent. We attribute these results to ThermalHerd's proactive and reactive routing schemes in combination with its distributed traffic throttling.

We also compared ThermalHerd to two other thermal management techniques: GlobalThermal, a uniform traffic throttling

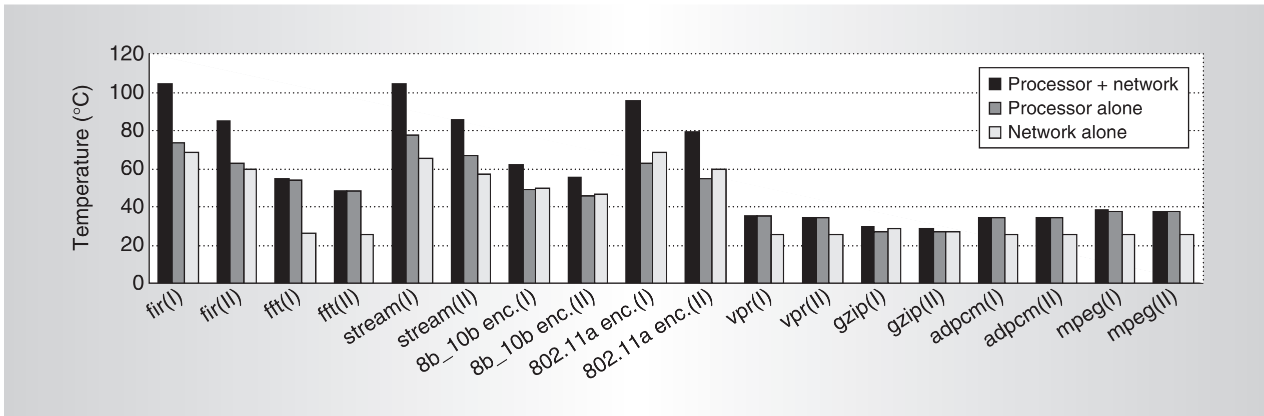

Figure 3. On-chip system thermal characterization based on MIT's Raw chip, which represents an architectural middle ground. The (I) and (II) after the benchmarks represent air-cooling at 300 and 600 linear feet per minute, respectively.

technique, and DistrThermal, a distributed traffic throttling technique which does not take the help of thermal-correlation based routing. As Figure 2 shows, distributed traffic throttling is much more efficient than global uniform traffic throttling. With reactive routing on top of distributed throttling, the temperature profile is further smoothed and throughput degradation reduces by more than 2×. Finally, with proactive routing, throughput degradation is negligible even at the low thermal emergency threshold of 84°C.

Toward temperature-aware design

Research in temperature-aware on-chip system design is still in the early stages. Our work emphasizes the need to account for the on-chip network's thermal contribution when exploring entire on-chip systems, which consist of computation, storage, and communication elements. In such systems, the chip temperature is an accumulated effect of the thermal interactions among *all* these components. By extending our thermal modeling and management techniques to jointly consider processors and memories as well as networks, we can begin to address the thermal issues of entire on-chip systems.

In on-chip systems, a component's relative thermal contribution varies according to chip architecture and application scenario. The chip architecture determines the complexity of processing elements versus storage versus communication—and thus the peak power consumption of all the elements. A chip with complex (wide-issue, multithreaded) processing elements will require larger storage elements (large multilevel caches and register files) as well as sophisticated communication elements, such as multilevel, wide buses; networks with wide link channels; deeply pipelined routers; and significant router buffering. At the other extreme are chip architectures, which have processing elements that are single arithmetic and logic units (ALUs) with a few registers at their input and output ports. Simple single-stage routers that have little buffering connect such processing elements.

The chip's power and thermal profile depends on the application's characteristics. Essentially, the computation and communication per data bit determines the processing, memory, and network elements' relative power and thermal contribution.

Thermal characterization of an on-chip system

To analyze the absolute and relative thermal impact of all chip components, we characterized MIT's Raw CMP. We chose Raw because it is in the middle of two architectural extremes—processing elements that are fat multithreaded cores and single ALUs, such as TRIPS cores. Raw has single-issue processing elements, 32-Kbyte caches and registers per tile, and networks with fairly narrow 32-bit channels, eight-stage pipelines, and limited router buffering.

Figure 3 shows the thermal characterization of Raw using three sets of benchmarks, including SPEC and Mediabench, stream computations, and bit-level computations.

For each benchmark, we considered both typical (300 linear feet per minute) and best (600 lfpm) air-cooling conditions. We further characterized the chip's peak temperature under three power-dissipation scenarios—processor power only, network power only, and processor plus network power.

The results of our analysis gave us three important insights:

- *The combination of components contributes to chip temperature.* As Figure 3 shows, the processors or networks alone never tipped chip temperature over the thermal emergency point. Together, however, the networks and processors pushed the chip peak temperature above 100°C.
- *Chip temperature is the result of thermal correlations among all on-chip components.* Each on-chip component's power consumption as well as its thermal correlation with other components affects its thermal contribution. Power consumption varies with architecture and applications, while the cooling package and physical distance determine the chip component's thermal correlation.
- *Different benchmarks demonstrated different thermal behavior.* Among the three sets of benchmarks, both stream and bit-level computation exhibit excellent scalability; they effectively use on-chip parallel computation and communication resources, leading to a high chip temperature. For SPEC and Mediabench benchmarks, the available compiler (rgcc) cannot efficiently exploit the coarse-grain parallelism. As a result, these benchmarks reflect an inefficient use of on-chip resources and thus have a lower thermal impact. The thermal impact of networks versus processors also varies across benchmarks. In 802.11a enc., 8b_10b enc., and fir, there is a high use of static networks and low access rate of on-chip data caches. Consequently, the network alone yields a comparable or higher temperature than the processor alone. On the other hand, stream reflects a high use of processing resources, and fft uses the dynamic network only partially. In these two cases, the processor's thermal impact is more significant.

Thermal management of an on-chip system

As the previous section implies, coordinating and regulating the behavior of all on-chip components is key to achieving effective thermal management for the entire chip. Because on-chip systems are inherently distributed, ThermalHerd's distributive and collaborative bent make it a good fit for thermal management of an entire chip.

Adapting ThermalHerd to an on-chip system requires extending two of its key features: distributed traffic throttling and routing based on thermal correlation. We have done some preliminary study on both these extensions. The first is to extend ThermalHerd's throttling mechanism to accommodate the distributed joint throttling of processing, storage, and network elements. Thus, when the temperature monitors flag a thermal emergency, the hotspot tile would begin to throttle both the processor (processing and memory elements) and the network by controlling the grant signal of the crossbar, instruction issue logic, and memory disable signals.

The second extension takes into account that thermal emergencies are often due to an unbalanced temperature profile, which means that workload migration can potentially balance the temperature profile and reduce peak temperature. ThermalHerd's thermal-correlation-based routing targets network traffic migration but it could extend to processors. Researchers have already explored computation migration in CMPs to improve performance, with a migration interval in the range of tens of microseconds.[15] Because this interval matches the thermal time constant of on-chip thermal hotspots, an extension might use computation migration to track and balance runtime thermal variations for on-chip computation resources. For example, to balance the chip's runtime thermal profile, an operating system scheduler could dispatch computation jobs on the basis of a thermal-correlation matrix of processing elements.

Our work is the first to address thermal issues in on-chip networks. We hope that it will lead to thermal-efficient network design, enabling network designers to build temperature-aware high-performance network microarchitectures. We have illustrated how the distributed, collaborative nature of on-chip

networks leads to distinct similarities with distributed on-chip systems and showed how extensions of our solution might address entire on-chip systems. This first step opens up an exciting area of performance enhancements and paves the way for studies of complete networked on-chip processing systems. MICRO

Acknowledgments

We thank Kevin Skadron and Wei Huang of the University of Virginia for their help in our understanding of HotSpot; the MIT Raw group, especially Michael B. Taylor, for helping with the Raw simulation platform as well as providing us with critical packaging and power information; Doug Burger of the University of Texas at Austin for supplying us with TRIPS network traffic traces; and Howard Chen of IBM for helping us validate our thermal model.

This work was supported in part by Princeton University's Wallace Memorial Honorific Fellowship, the National Science Foundation under grant CCR-0237540 (CAREER) and CCR-0324891, and the Marco Gigascale Systems Research Center.

References

1. J. Srinivasan et al., "Exploiting Structural Duplication for Lifetime Reliability Enhancement," *Proc. Int'l Symp. Computer Architecture* (ISCA 05), IEEE CS Press, 2005, pp. 520-531.
2. L.T. Yeh and R.C. Chu, *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*, ASME Press, 2002.
3. *2003 International Technology Roadmap for Semiconductors*, Sematech Inc., 2003; http://public.itrs.net.
4. W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proc. 38th Design Automation Conf.* (DAC 01), IEEE CS Press, 2001, pp. 684-689.
5. K. Sankaralingam et al., "Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture," *Proc. Int'l Symp. Computer Architecture* (ISCA 03), IEEE CS Press, 2003, pp. 422-433.
6. M.B. Taylor et al., "The RAW Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs," *IEEE Micro*, vol. 22, no. 2, Mar.-Apr. 2002, pp. 25-35.
7. L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, vol. 35, no. 1, Jan. 2002, pp. 70-78.
8. H.-S. Wang et al., "Orion: A Power-Performance Simulator for Interconnection Networks," *Proc. 35th Int'l Symp. Microarchitecture* (Micro-35), IEEE CS Press, 2002, pp. 294-305.
9. N.B. Nguyen, "Properly Implementing Thermal Spreading Will Cut Cost While Improving Device Reliability," *Proc. Int'l Symp. Microelectronics*, vol. 2920, Int'l Soc. Optical Engineering (SPIE), 1996, pp. 383-386.
10. D. Meeks, "Fundamentals of Heat Transfer in a Multilayer System," *Microwave J.*, vol. 1, no. 1, Jan. 1992, pp. 165-172.
11. S. Song, S. Lee, and V. Au, "Closed-Form Equation for Thermal Constriction/Spreading Resistances with Variable Resistance Boundary Condition," *Proc. Int'l Electronics Packaging Conf.*, Amer. Soc. Mechanical Engineers, 1994, pp. 111-121.
12. J.E. Sergent and A. Krum, *Thermal Management Handbook for Electronic Assemblies*, McGraw-Hill, 1998.
13. R.H.J.M. Otten and R.K. Brayton, "Planning for Performance," *Proc.* 35th *Design Automation Conf.* (DAC 98), IEEE CS Press, 1998, pp. 122-127.
14. T.Y. Chiang, K. Banerjee, and K.C. Saraswat, "Analytical Thermal Model for Multilevel VLSI Interconnects Incorporating Via Effect," *IEEE Electron Device Letters*, vol. 23, no. 1, Jan. 2002, pp. 31-33.
15. K.A. Shaw and W.J. Dally, "Migration in Single-Chip Multiprocessors," *Computer Architecture Letters*, vol. 1, 2002; http://www.cs.virginia.edu/~tcca/2002paps.html.

Li Shang is an assistant professor of electrical and computer engineering at Queen's University. His research interests include computer architecture, computer-aided design of ICs and systems, thermal and power modeling and optimization, and mobile computing. Shang has a PhD in electrical engineering from Princeton University. He is a member of the IEEE and ACM.

Li-Shiuan Peh is an assistant professor of electrical engineering at Princeton University. Her research interests include interconnection networks and parallel computer architectures. Peh has a BS from the National University of

Singapore and a PhD from Stanford University, both in computer science. She is a member of the IEEE and ACM.

Amit Kumar is a PhD student in the Department of Electrical Engineering at Princeton University. His research interests include interconnection networks and power- and thermal-aware system design. Kumar has a BTech from the Indian Institute of Technology and an MA from Princeton University, both in electrical engineering. He is a student member of the IEEE and ACM.

Niraj K. Jha is a professor of electrical engineering at Princeton University. His research interests include low-power hardware and software design, computer-aided design of ICs and systems, digital system testing and distributed computing. Jha has a BTech in electronics and electrical communication engineering from the Indian Institute of Technology, an MS in electrical engineering from State University of New York at Stony Brook, and a PhD in electrical engineering from the University of Illinois. He is a fellow of the IEEE and ACM.

For questions about this article, contact Li Shang, Dept. of Electrical and Computer Engineering, Queen's University, Kingston, ON, K7L 3N6, Canada; li.shang@queensu.ca.

For further information on this or any other computing topic, visit our Digital Library at http://www.computer.org/publications/dlib.

IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING

Learn how others are achieving systems and networks design and development that are dependable and secure to the desired degree, without compromising performance.

This new journal provides original results in research, design, and development of dependable, secure computing methodologies, strategies, and systems including:

- Architecture for secure systems
- Intrusion detection and error tolerance
- Firewall and network technologies
- Modeling and prediction
- Emerging technologies

Learn more about this new publication and become a subscriber today.

www.computer.org/tdsc

Publishing quarterly
Member rate: $31
Institutional rate: $285

IEEE

IEEE Computer Society