Thermal Modeling, Characterization and Management of On-chip Networks

Li Shang Li-Shiuan Peh Amit Kumar Niraj K. Jha

Department of Electrical Engineering
Princeton University, Princeton, NJ 08544

Abstract

Due to the wire delay constraints in deep submicron technology and increasing demand for on-chip bandwidth, networks are becoming the pervasive interconnect fabric to connect processing elements on chip. With ever-increasing power density and cooling costs, the thermal impact of on-chip networks needs to be urgently addressed.

In this work, we first characterize the thermal profile of the MIT Raw chip. Our study shows networks having comparable thermal impact as the processing elements and contributing significantly to overall chip temperature, thus motivating the need for network thermal management. The characterization is based on an architectural thermal model we developed for on-chip networks that takes into account the thermal correlation between routers across the chip and factors in the thermal contribution of on-chip interconnects. Our thermal model is validated against finite-element based simulators for an actual chip with associated power measurements, with an average error of 5.3%.

We next propose ThermalHerd, a distributed, collaborative run-time thermal management scheme for on-chip networks that uses distributed throttling and thermal-correlation based routing to tackle thermal emergencies. Our simulations show ThermalHerd effectively ensuring thermal safety with little performance impact. With Raw as our platform, we further show how our work can be extended to the analysis and management of entire on-chip systems, jointly considering both processors and networks.

1 Introduction

Chip reliability and performance are increasingly impacted by thermal issues. Circuit reliability depends exponentially upon operating temperature. Thus, temperature variations and hotspots account for over 50% of electronic failures [29]. Thermal variations can also lead to significant timing uncertainty, prompting wider timing margins, and poorer performance. Traditionally, cooling system design is based on worst-case analysis. As heat density and system complexity scale further, worst-case design becomes increasingly difficult and infeasible. This has led to recent processor designs, such as the Intel Pentium4-M [1] and the IBM Power5 [6], moving to average-case thermal design and employing run-time thermal management schemes upon occurrence of thermal emergencies.

As we move towards application-specific multiprocessor systems-on-a-chip (SoCs) [2, 7, 18] and general-purpose chip-multiprocessors (CMPs) [16, 26] where a chip is composed of multiple processing elements interconnected with a network fabric, system chip temperature becomes an accumulated effect of both processing and communication components. With networks consuming a significant portion of the chip power budget [2, 10, 27], and hence having a substantial thermal impact, understanding the joint thermal behavior of all on-chip components, both processors and networks, is the key to achieving efficient thermal design.

Researchers [3, 22, 24] have started addressing thermal issues in microprocessors. Unlike centralized microprocessors, networks are distributed in nature, which imposes unique requirements on thermal modeling and management. Moreover, multiprocessor on-chip systems share the distributed nature of on-chip networks. Therefore, we see addressing of on-chip network thermal issues providing valuable insights for managing entire on-chip systems. For networks, recent studies mainly focus on power issues, including modeling and characterization [10, 15, 28], design [27], and management [11, 20]. Even though both power and temperature are a function of the communication traffic, the network thermal and power consumption profiles exhibit different run-time behavior. Power-oriented optimization techniques cannot fully address network thermal issues. We need to tackle the thermal impact of on-chip networks directly.

In this paper, we first construct an architecture-level thermal model for on-chip networks. We then build a network simulation platform that enables rapid evaluation of network performance, power consumption, and thermal profile within a single run-time environment. Using this model, we characterize the thermal behavior of the on-chip networks in the MIT Raw CMP. We then propose ThermalHerd, a distributed run-time mechanism that dynamically regulates network temperature. ThermalHerd consists of three on-line mechanisms: dynamic traffic prediction, distributed temperature-aware traffic throttling, and proactive/reactive thermal-correlation based routing. The performance of ThermalHerd is evaluated using on-chip network traffic traces from the UT-Austin TRIPS CMP [16]. Our results demonstrate that ThermalHerd can effectively regulate network temperature and eliminate thermal emergencies. Furthermore, ThermalHerd proactively adjusts and balances the network thermal profile to achieve a lower junction temperature, thus minimizing the need for throttling and its impact on network performance. Finally, using Raw as a case study, we extend our work to address the thermal issues of on-chip systems, incorporating on-chip processing elements. A brief summary of our contributions is as follows.

- First work targeting run-time thermal impact of on-chip networks.
- Characterization of the relative thermal impact of computation and communication resources in Raw.

Proceedings of the 37th International Symposium on Microarchitecture (MICRO-37 2004)
1072-4451/04 $20.00 © 2004 **IEEE**

- An architectural thermal model and integrated simulation platform that facilitates trace-driven performance, power and temperature analysis of networks.
- A run-time thermal management technique for on-chip networks that effectively regulates temperature with little performance degradation.
- Extension of network thermal characterization and management to entire on-chip systems, jointly considering both processors and networks.

The rest of this paper is organized as follows. In Section 2, we first use Raw as a motivating case study for demonstrating the thermal impact of on-chip networks. In Section 3, we describe the thermal modeling methods for on-chip networks. In Section 4, we give details of ThermalHerd and evaluate its performance in Section 5. In Section 6, we discuss extending our work from on-chip networks to on-chip systems. In Section 7, we present prior related work and discuss related research issues, and Section 8 concludes the paper.

2 Motivating Case Study: On-chip Network Thermal Impact in Raw

We first motivate the need for run-time thermal management of on-chip networks by studying the thermal impact of the on-chip networks in the MIT Raw CMP.

The Raw chip accommodates 16 tiles connected in a 4x4 mesh topology, clocked at 425MHz frequency and a nominal voltage of 1.8V. The die size is 18.2x18.2mm^2. In each 4x4mm^2 tile, the processor is a single-issue MIPS-style processing element consisting of an eight-stage pipeline equipped with 32KB data and 32KB instruction caches. Tiles are connected by four 32-bit on-chip networks, two statically-scheduled by the compiler through a 64KB switch instruction memory and two dynamically-routed. The chip is manufactured using the IBM CMOS7SF Cu6 0.18μm technology. As shown in Figure 1, it uses the IBM ceramic column grid array (CCGA) package with direct lid attach (DLA) thermal enhancement. The thermal performance of the packaging ranges from 9°C/W to 5°C/W under different air-cooling conditions.

In Raw, the thermal impact of the on-chip network is governed by various design issues. Performance requirements govern the network complexity, and hence the peak power consumption. The chip layout affects the thermal interaction between the network and other computation/storage tiles. The chip temperature is also directly affected by cooling solutions, which vary under different application scenarios and cooling budgets. In this section, we focus on analyzing the thermal impact of just the on-chip network. Complete thermal characterization for the entire chip will be discussed in Section 6.

In Raw, the two static networks contain the following components: switch instruction memory, switch pipeline logic, crossbars, FIFOs, and wires. The two dynamic networks are composed of crossbars, FIFOs, wires, and routing control logic. We use Raw Beetle, a validated cycle-accurate simulator [26], to capture the run-time activities of different network components. Combining the simulation results with the capacitances estimated using an IBM placement tool by Kim [9], we derive network run-time power consumption. Based on the chip layout and packaging information, we construct an architectural thermal model for the on-chip network using the model described in Section 3. Thermal characteristics, including thermal conductivities and capacitances, and geometric information of the package materials including the interface material, DLA, ceramic carrier, etc., and air-cooling conditions are provided by the packaging manual [25]. The ambient temperature is set to 25°C [1]. Figure 2 shows the chip peak temperature contributed by typical-case network power dissipation under different air-cooling conditions (linear feet per minute, or lfpm), where, as explained in [9, 10], the typical-case power assumes the activity factor of 0.25 in each of the four on-chip networks. With poor air-cooling, the network power alone can push the chip temperature up to about 90°C. Even at a typical air cooling of 300lfpm, the network temperature alone can reach a significant 77°C.

Next, we characterize the thermal impact of the on-chip network using three classes of benchmarks provided by the Raw binary distribution. Details of these benchmarks are available in [26]. *fir* and *stream* are stream benchmarks. *8b_10b enc.* and *802.11a enc.* are bit-level computation benchmarks. *gzip* and *mpeg* are ILP computation benchmarks. Each study is based on the typical (300lfpm) and best (600lfpm) air-cooling conditions. Detailed benchmark explanation and complete characterization results are given in Section 6. As shown in Figure 3, the two stream and two bit-level computations benchmarks lead to a high peak temperature. This is due to the heavy utilization of static network resources, which have a higher capacitance than the dynamic network in Raw as they are used as the primary communication fabric in the chip. *gzip* and *mpeg* result in a lower peak temperature due to two reasons. First, the limited parallelism in these benchmarks leads to low resource utilization. Second, limited by the available compiler, the traffic is mainly relayed by the dynamic networks.

The above studies demonstrate that on-chip networks can have a significant thermal impact on overall chip tem-

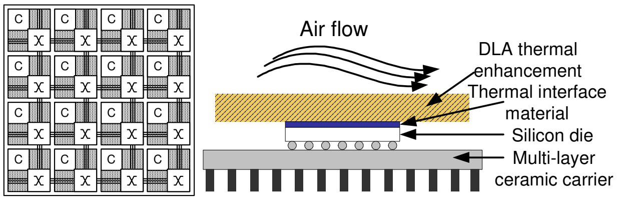


Figure 1. MIT Raw CMP with cooling package.

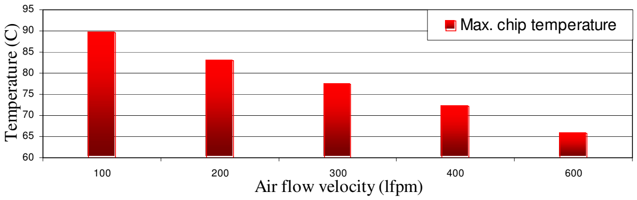


Figure 2. On-chip network thermal analysis I.

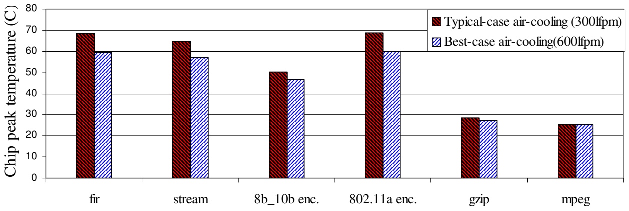


Figure 3. On-chip network thermal analysis II.

[1] Ambient temperature varies across different systems. Relative trends are preserved for other typical settings.

perature. As will be discussed in Section 6, in Raw, the thermal contributions of both processors and networks are comparable, and can be substantial, depending on application characteristics. Based on the benchmarks we used with typical air-cooling conditions, networks or processors alone can reach peak temperatures of 68.6°C and 77.9°C, respectively. When networks and processors are jointly considered, certain stream and bit-level computation benchmarks can push chip peak temperature up to 104.7°C under typical air-cooling conditions. Therefore, more sophisticated cooling solutions and thermal management techniques are required to guarantee safe on-line operation. On the other hand, the SPEC and Mediabench benchmarks result in low power and thermal impact on both networks and processors due to underutilized on-chip resources.

3 Thermal Modeling of On-chip Networks

The cooling package of today's high-performance on-chip systems is complex due to high power density and strict cooling requirements. To facilitate characterization of chip architectures, we need thermal models that can accurately capture the characteristics of these sophisticated thermal packages with only the limited architectural input parameters available at an early-stage design.

Skadron *et al.* first proposed an architectural thermal model for microprocessors, HotSpot [22], which constructs a multi-layer lumped thermal RC network to model the heat dissipation path from the silicon die through the cooling package to the ambient. In HotSpot, the silicon die is partitioned into functional blocks based on the floorplan of the microprocessor, with a thermal RC network connecting the various blocks. HotSpot can be readily used to model on-chip network routers – with each router within the chip floorplan modeled as a block, and a thermal RC network constructed in the same fashion as when the blocks were functional units within a microprocessor. However, certain characteristics of on-chip networks are not accurately captured by HotSpot. First, lateral thermal correlation is modeled using lateral thermal resistors connecting adjacent blocks, and solved using a closed-form thermal equation [23]. This equation was originally proposed to model the spreading thermal resistance for a large symmetric slab area. As the geometric and thermal boundary conditions of a silicon die do not match the above scenario, thermal correlation tends to be underestimated. The temperature of an on-chip router is affected by its own power consumption, and that of its neighborhood and also remote routers. In on-chip networks, the power and thermal impact of each individual router is limited. Inter-router thermal correlation plays an important role in shaping the overall chip temperature profile. Hence, accurately characterizing the spatial thermal correlation is critical for understanding network thermal behavior.

Second, the current release of HotSpot does not model the thermal impact of metal interconnects. Due to the high thermal resistance of the silicon dioxide and low dielectric constant (low-k) materials, the contribution of the interconnects to overall chip temperature is of increasing concern [5] and needs to be considered.

The above issues prompted us to develop an architectural thermal model that handles the thermal characteristics of on-chip networks – one that aptly captures inter-router thermal correlation and models on-chip link circuitry.

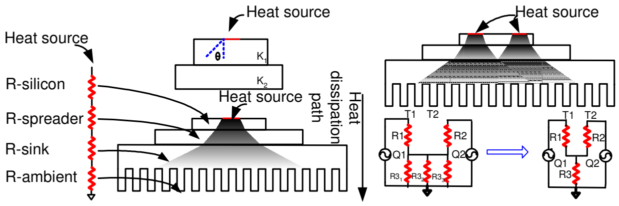


Figure 4. heat spreading angle and inter-router thermal correlation.

3.1 Inter-router Thermal Correlation Modeling

Our model is based on the notion of the heat spreading angle – the angle at which heat dissipates through the different layers of packaging. In microelectronic packages, the heat flow from the silicon surface to the ambient is three-dimensional – heat dissipates in the vertical direction and also spreads along horizontal directions, which can be described by Fourier's law. The heat spreading angle forms the basis of calculations of thermal resistances and thermal correlations of the thermal RC network (that is constructed in the same fashion as in HotSpot).

Heat dissipation path: First, let us analyze the thermal resistance of the heat dissipation path of each on-chip router. The heat spreading angle, θ, (see Figure 4) can be estimated based upon the ratio of the thermal conductivities of the current packaging layer's material, k_1, and the underlying packaging layer's material, k_2 [13]:

$$\theta = \tan^{-1}(k_1/k_2) \tag{1}$$

Then, as shown in Figure 4, the thermal resistance, R, of a rectangular heat source on a carrier, that includes the thermal spreading effect is [12]:

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

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.

For each on-chip router i, the thermal resistance, R_i, is the summation of the thermal resistance of each thermal component along the heat dissipation path, as follows:

$$R_i = R_{i\_silicon} + R_{i\_spreader} + R_{i\_sink} + R_{i\_ambient} \tag{3}$$

where $R_{i\_silicon}$, $R_{i\_spreader}$, R_{i_sink}, and $R_{i\_ambient}$ are the thermal resistance of the on-chip router through silicon die, heat spreader, heat sink and ambient, respectively.

Both $R_{i\_silicon}$ and $R_{i\_spreader}$ can be obtained using Equations (1) and (2). For $R_{i\_silicon}$, the area of the heat source is equal to the size of the on-chip router, and the heat spreading angle in the silicon die can be determined by the thermal conductivity ratio of the silicon die and the heat spreader[2]. For $R_{i\_spreader}$, the area of the heat source is the original router area plus the area expansion due to heat spreading in the silicon die, which equals $(x + 2L_{silicon}\tan\theta_{silicon})(y + 2L_{silicon}\tan\theta_{silicon})$. The heat spreading angle is affected by the thermal conductivity

[2] If an interface material is used, the heat spreading angle can be determined similarly, and Equation (3) should also take thermal resistance in the interface material into account.

ratio of the heat spreader and heat sink. For R_{i_sink}, the other side of the heat sink is attached to a cooling fan, and the heat dissipation is based on heat convection. Previous work has proposed a closed-form thermal equation to address the heat spreading issue in heat sinks. The thermal resistance of a heat sink can be determined by the following equation [23]:

$$R_{i\_sink} = \frac{1}{\pi k a}\left(\epsilon\tau + (1-\epsilon)\frac{\tanh(\lambda\tau) + \frac{\lambda}{B_i}}{1 + \frac{\lambda}{B_i}\tanh(\lambda\tau)}\right) \tag{4}$$

where k is the thermal conductivity of the heat sink, a is the source radius, ϵ, τ, λ are dimensionless parameters defined in [23], and B_i is a Biot number.

The thermal resistance along the heat convection path can be estimated as follows [17]:

$$R_{i\_ambient} = \frac{1}{h_c A_s} \tag{5}$$

where h_c is the convection heat transfer coefficient and A_s is the effective surface area.

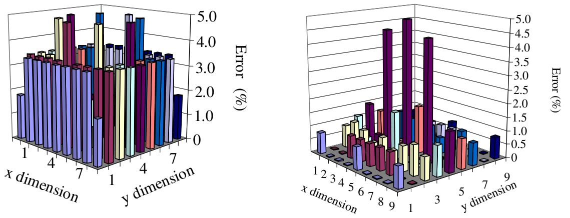
Inter-block thermal correlation: In general, the steady-state temperature of each location across the silicon die is a function of the power consumption of all the on-chip heat sources:

$$T(x,y) = \sum_{k=1}^{N} C_i P_i \tag{6}$$

where $T(x,y)$ is the temperature at location (x,y) on the silicon die, N is the total number of on-chip heat sources, P_i is the power consumption of heat source i, and C_i is the thermal correlation (thermal resistance) between heat source i and location (x,y).

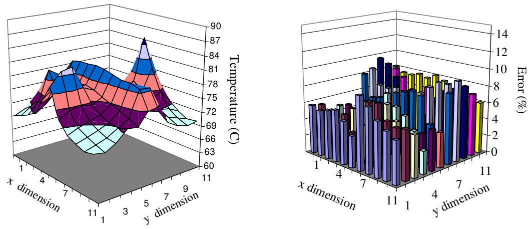
The thermal correlation among on-chip blocks (routers) can be characterized based on the cooling structure, the heat spreading angle in each cooling package layer, and the inter-router distance. There exists a duality between heat transfer and electrical phenomena. The linearized superposition principle in electrical circuits can be extended to thermal circuits to analyze the inter-router thermal effect. As shown in Figure 4, $Q1$ and $Q2$ denote the power consumption of two on-chip routers. The corresponding heat dissipation paths of these two routers are initially separate but will finally merge into one path due to the heat spreading effect. Then, using the linearized superposition principle, for these two routers, the heat dissipation paths can be divided into two parts from the merged point – before this point, the heat dissipation paths can be modeled with two separate thermal resistors, $R1$ and $R2$. After this point, the heat dissipation paths are modeled with a shared thermal resistor $R3$. The thermal correlation between two heat sources is determined by the value of $R3$, which is the thermal resistance of the shared heat dissipation path from the point where the two heat dissipation paths are merged together to the ambient environment. The position of the merged point can be estimated based on the heat spreading angle in each packaging material and the inter-router distance. The junction temperature of each router, $T1$ and $T2$, including inter-router thermal correlation, can then be estimated using thermal superposition:

$$\begin{aligned} T1 &= Q1R1 + (Q1+Q2)R3 \\ T2 &= Q2R2 + (Q1+Q2)R3 \end{aligned} \tag{7}$$



(a) Estimation error. (b) Estimation error.

Figure 5. Thermal model validation against FEMLAB.



(a) Chip thermal profile. (b) Estimation error.

Figure 6. Thermal model validation against IBM in-house finite-element based simulator.

3.1.1 Validation

In this section, we discuss validation of our thermal models.

FEMLAB, a finite-element based simulator: To evaluate the accuracy of our thermal model, we first use a commercial finite-element based simulator, FEMLAB [8]. We assume the silicon die's dimension is $18mm \times 18mm \times 0.6mm$, the thermal conductivity is $100W/mK$, the heat spreader's dimension is $30mm \times 30mm \times 1mm$, and the heat sink's dimension is $60mm \times 60mm \times 6.8mm$. Both the heat spreader and heat sink are assumed to be made of copper, whose thermal conductivity is $400W/mK$. The silicon die is partitioned into 9×9 blocks evenly. We set the ambient temperature to 25°C.

In the first scenario, we assume the central block, (5,5), is the only heat source and has a 2.5W power consumption. We use this setup to evaluate the accuracy of modeling a single heat source. Using our model, the temperature distribution on the silicon surface varies from 32.6°C to 28.2°C. Block (5,5) is the hottest spot, and the four boundary blocks have the lowest temperature. Figure 5(a) shows the modeling error of our thermal model against FEMLAB. The error is consistently less than 5% (the average being 2.9%). To evaluate the accuracy of our model in capturing inter-router thermal correlation, we assume three heat sources situated at (5,5), (3,5) and (7,5), each with a power consumption of 2.5W. Using our model, the temperature varies from 39.8°C to 34.7°C. The estimation error of our thermal model against FEMLAB is shown in Figure 5(b), where the maximum error is less than 5% (the average being 1.0%).

Actual chip with power measurements: Next, we evaluate our thermal model against an actual chip design from IBM [4]. In this design, a $13mm \times 13mm \times 0.625mm$ chip is soldered to a multilayer ceramic carrier. This chip is attached to a heat sink and placed inside a wind tunnel. The power consumption profile across the silicon die is based on physical measurements. We use our thermal model to evaluate the temperature profile of this IBM design. Figure 6(a) shows the temperature simulation result using our thermal model. Comparing it against an in-house finite-element based thermal simulator at IBM, Figure 6(b) shows the estimation errors in the on-chip thermal profile. As shown in this figure, the maximum error is less than 10%, and the average error is 5.3%. Using our thermal model, the junction temperature of the silicon die varies within [70.2, 85.4]°C. The IBM thermal simulator shows the junction temperature varying within [73.2, 87.8]°C.

Both validation studies demonstrate the importance of accurately modeling the spatial variance of temperature across the silicon surface. While our model and the finite-element based simulators show the spatial thermal variance to be around 15°C, HotSpot [22] reports only 8°C.

3.2 Thermal Modeling of Links

The power consumed in the on-chip link circuitry affects the temperature of both silicon and metal layers. As on-chip links are typically fairly long, buffers are inserted to reduce signal propagation delay. The inserted buffers not only affect network performance and power consumption, but also its 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 dielectrics and thus serve as efficient heat dissipation paths. To model on-chip link temperature effectively, buffer insertion effects need to be considered. Previous research work has calculated the optimal length of interconnect at which to insert repeaters as [14]:

$$l_{opt} = const\sqrt{\frac{r_0(c_0 + c_p)}{rc}} \tag{8}$$

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 can be calculated using the following equation [5]:

$$T(x) = T_0 + \frac{j^2 \rho L_H^2}{k_M}\left(1 - \frac{\cosh(\frac{x}{L_H})}{\cosh(\frac{L}{2L_H})}\right) \tag{9}$$

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

Based on our analysis, the major thermal contribution of the link circuitry lies in the silicon (buffers). Due to the limited self-heating power, when the secondary heat dissipation path from top silicon dioxide and low-k material layers to the printed circuit board is considered, thermal hotspots are located in the silicon layer instead of the metal layers.

3.3 *Sirius*: Network Thermal Simulation Environment

The thermal model described above is built into a network simulation environment, called *Sirius*, which provides an architecture-level platform for rapid exploration of the performance, power consumption, and thermal profile of on-chip networks.

Network model: We use a flit-level on-chip network model, which specifies the topology and resource configuration of the network. Currently, two-dimensional direct-network topologies are supported. Each router is specified with a pipeline model. Various routing schemes, including deterministic, oblivious, adaptive, etc., are integrated.

Power model: This model is adapted from Orion [28], an architecture-level network power (dynamic/leakage) model. Network power consumption is determined by the network architecture, implementation technology, and traffic activity. The first two parameters are defined in the network model. The last is captured during run-time simulation.

Thermal model: This is the model presented in this section. It is created, based on the network architecture and cooling structure, during compilation.

Timing-driven simulator: This is built on top of a timing-driven simulation engine. During on-line simulation, traffic activities are gathered and fed into the power model to estimate network power consumption. This is then fed into the thermal model periodically to estimate the network temperature profile. Timing information is also gathered to monitor network latency and throughput.

4 ThermalHerd: Distributed, Collaborative Run-time Thermal Management

The thermal behavior of on-chip networks is inherently distributed in nature. Thermal emergencies can occur in different locations and change dynamically. In addition, on-chip networks are heavily performance-driven. Here, we explore run-time management of network temperature, using ThermalHerd, a distributed scheme where routers collaboratively regulate the network temperature profile and work towards averting thermal emergencies while minimizing performance impact.

4.1 ThermalHerd: Overall Architecture

The microachitecture of a ThermalHerd router consists of five key components:

- *Temperature monitoring:* At run-time, temperature monitors, such as thermal sensors or on-line thermal models, periodically report the local temperature to each router, triggering an emergency mode when the local temperature exceeds a thermal threshold.

- *Traffic monitoring and prediction:* ThermalHerd's throttling and routing relies on traffic activity counters embedded in each router that facilitates prediction of future network workload.

- *Distributed throttling:* Upon a thermal emergency, the routers at the hotspot will throttle incoming traffic in a distributed way, reducing power consumption in the region, thus cooling the hotspot.

- *Reactive routing:* In addition, each router reacts by adapting the routing policy to direct traffic away from the hotspots, relying on the thermal correlation information that is exchanged between routers periodically.

- *Proactive routing:* During normal operation, routers will proactively shape their routing decisions to reduce traffic to potential thermal hotspots, based on thermal correlation information.

4.2 Run-time Thermal Monitoring

There are two different mechanisms that can be used to monitor network temperature: thermal sensors and on-line thermal estimation. Thermal sensors have been widely used in high-performance systems. For instance, Power5 [6] employs 24 digital temperature sensors to obtain the chip temperature profile. Another approach is dynamic thermal modeling and estimation. In [22], an architecture-level thermal model is used to estimate and monitor the temperature profile of microprocessors. Since thermal transitions are fairly slow, the computation overhead introduced by on-line thermal estimation is tolerable.

In this work, we focus on dynamic thermal management. Either of the above temperature monitoring techniques can be used. For thermal sensors, if one sensor per router is not affordable for large on-chip networks, the network can be partitioned into regions and multiple adjacent routers within a region could share the same sensor. For the on-line thermal modeling based approach, on-line power and temperature models are required. An efficient on-line power estimation mechanism for networks is proposed in [20]. This can then be fed into thermal models implemented with dedicated hardware or executed on on-chip processing elements for temperature estimation.

4.3 Dynamic Traffic Estimation and Prediction

In ThermalHerd, each router is equipped with two sets of traffic counters to dynamically estimate the traffic workload. Traffic counter, cnt_{local}, is integrated with the injection buffer to monitor locally-generated traffic. Traffic counter, $cnt_{neighbor}$, is used to monitor the traffic arriving from the neighborhood. Two other counters, cnt_{his_local} and $cnt_{his\_neighbor}$, are used for traffic bookkeeping. Both traffic and bookkeeping counters are updated based on a predefined timing window, T_{traff}. Within each T_{traff}, traffic counters, cnt_{local} and $cnt_{neighbor}$, count the total amount of incoming flits from the injection and input ports, respectively. At the end of each T_{traff}, the traffic information in the traffic and bookkeeping counters are combined using weighted average filtering to eliminate transient traffic fluctuations.

4.4 Distributed Traffic Throttling

The key challenges faced in designing a distributed traffic throttling mechanism is that it has to effectively regulate overall network temperature, averting thermal emergencies, while minimizing performance impact.

Exploring the design space for distributed throttling: As shown in Equation (6), the temperature contributed by each heat source is affected by the thermal correlation, which is determined by the cooling structure and distance. Neighboring heat sources have more thermal impact than remote ones. Therefore, to effectively alleviate the thermal emergency, traffic throttling around the hotspot locations requires less throughput reduction, and hence less performance penalty.

Traffic throttling within a router also affects the power consumption of neighboring routers – throttling the router injection port decreases the traffic through neighboring routers; throttling the router input and output ports decreases the available network bandwidth. We study the power throttling effect on a 9×9 on-chip network using uniform random traffic. Router $R_{4,4}$ is chosen to be the only traffic regulation point. Figure 7 shows the power savings of router $R_{4,4}$ and the whole network as router $R_{4,4}$ throttles an increasing percentage of incoming traffic (we call this the traffic throttling ratio). The black and gray bars show the power savings of router $R_{4,4}$ and the whole network, respectively. The line shows the ratio of the power savings of router $R_{4,4}$ to that of the total network. It shows that when $R_{4,4}$ is throttling only a small percentage of its arriving traffic, most of the power reduction comes from the throttled router, and the power reduction in the remaining part of the network is negligible. As $R_{4,4}$ throttles more and more of its arriving traffic, the power savings of the local router increases, while at the same time, the throttling effect spreads from the local router to its neighborhood, much like traffic congestion on a local street spreading beyond its initial location to other roads feeding into this street. Thus, power reduction at other routers also increases and begins to dominate the total power savings, with neighboring routers seeing a sharper power reduction than remote ones.

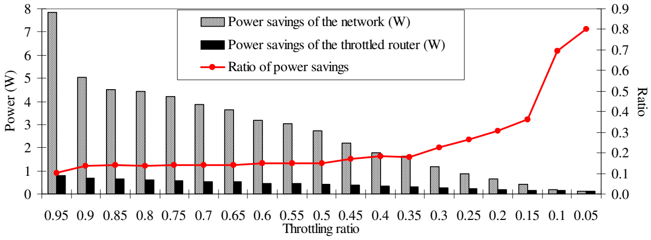


Figure 7. Power impact of localized throttling.

Distributed throttling in ThermalHerd: Based on the above observations, we propose the following distributed throttling mechanism in ThermalHerd.

When a thermal hotspot is detected at the local router R_i, this router begins to decrease the local workload by throttling the input traffic. The policy of traffic throttling is controlled by an exponential factor k and local traffic estimation, as follows:

$$Quota_i = k \times (N_{his\_local} + N_{his\_neighbor}), \quad k \leq 1 \quad (10)$$

where $Quota_i$ is the traffic quota used to control the total amount of workload that is allowed to pass through this router. At the beginning of each thermal timing window, a new temperature is reported. If the temperature still increases, in the next timing window, the traffic quota is further multiplied by k, otherwise, the throttling ratio is kept the same. 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 till the thermal emergency is removed. Furthermore, each router uses the following policy to split the quota, $Quota_i$, between the traffic injected locally, $Quota_{local}$, and the traffic arriving from the neighborhood, $Quota_{neighbor}$.

$$Quota_{local} = \begin{cases} N_{his\_local} & \text{if } N_{his\_local} \leq Quota_i \\ Quota_i & \text{if } N_{his\_local} > Quota_i \end{cases}$$

$$Quota_{neighbor} = \begin{cases} Quota_i - N_{his\_local} & \text{if} > 0 \\ 0 & \text{if} \leq 0 \end{cases} \quad (11)$$

This policy is biased towards providing enough traffic quota to the traffic generated locally. The rationale behind the policy is as follows. Without enough traffic quota, the locally-generated traffic will be blocked in the injection buffer, which increases network latency. Traffic from neighboring nodes, on the other hand, can be redirected through other paths, thus avoiding a performance penalty. Hence, the above-mentioned bias towards local traffic reduces the performance penalty.

4.5 Thermal-Correlation Based Routing Algorithm

Traffic throttling achieves a lower junction temperature by reducing network traffic and power consumption. While distributed throttling is efficient, it is not without a performance penalty. Since network thermal hotspots are often a result of imbalanced traffic, thermal-aware routing algorithms that can redirect traffic from throttled routers to minimize performance penalty and then shape network traffic patterns suitably will potentially balance the network temperature profile and avoid or reduce traffic throttling.

Our routing protocols rely on the thermal information exchanged within the network. At run-time, when thermal hotspots are detected, routers located at hotspot regions are marked as *hotspots*, and send special packets across the network. Since temperature transition is a very slow process, a noticeable temperature variation takes at least hundreds of microseconds. The power consumption and delay overhead introduced by these messages are thus negligible. For instance, in a 4×4 network, encoding the location of each hotspot takes four bits. Using a 100μs temperature report interval, the communication overhead for each hotspot is about 1bits/μs. When no thermal emergency occurs, the location information with the highest chip temperature is relayed through the network.

We discuss both proactive and reactive routing protocols next.

Proactive routing protocol: The proactive routing scheme continuously monitors the network temperature profile. When the maximum chip temperature is below the thermal emergency limit, it dynamically adjusts traffic to balance the network temperature profile and reduces the peak temperature.

Reactive routing protocol: Upon receiving the thermal emergency information, the reactive routing protocol replaces the proactive routing protocol. It tries to steer packets away from throttled regions to minimize the performance penalty due to throttling. In addition, reactive routing tries to balance the network temperature profile to reduce the hotspot temperature, hence the need for throttling.

Both routing schemes use traffic reduction to achieve their goals. Since non-minimal path routing results in more hops and links being traversed, and thus higher power consumption and hence potentially higher junction temperatures, we use minimal-path adaptive routing functions.

For both routing schemes, in order to balance the network temperature profile, they should choose the routing paths which have minimal thermal correlation with the regions with the highest temperature. Since the inter-router thermal effect is based on distance, among all of the minimal-path routing candidates, the routing path with the farthest thermal distance should be chosen. However, this approach significantly reduces path diversity and pushes traffic workload towards the coolest boundaries, which can result in an unbalanced traffic distribution and may also generate new thermal hotspots in the future.

Detailed thermal analysis shows that inter-router thermal correlation dramatically decreases with increasing inter-router distance. Furthermore, the thermal correlation with remote routers is very small. We thus use a thermal correlation threshold, α, to select routing candidates. The thermal correlation of each routing candidate is determined as in Equation (7). Intuitively, ThermalHerd's routing protocol jointly considers thermal correlation and traffic balancing. It tries to eliminate routing paths with a high thermal correlation while leaving enough alternative routing candidates to balance the network traffic. It does so by picking paths where the thermal correlation between the source and every hop along the path is below the thermal correlation threshold α. Since we use minimal-path routing, routing candidates satisfying such a thermal correlation threshold criterion may not be available. If so, the candidate routing path with the least thermal correlation is chosen.

To strike a good balance between temperature and performance, in our implementation, we use different thermal correlation thresholds between proactive and reactive routing policies. When the maximum network temperature is below the thermal emergency limit, to minimize performance penalty, a less aggressive traffic redirection policy is used for proactive routing (in this work, we set the thermal correlation threshold to be equivalent to $2L$, in which L is the physical distance between neighboring routers). When a thermal emergency occurs, we set the threshold to be equivalent to $4L$ for reactive routing, since at that time, reducing the chip temperature is the first-order issue, and also most of the performance penalty is due to traffic throttling.

5 ThermalHerd Evaluation

In this section, we evaluate the performance of ThermalHerd using CMP traffic traces generated from the TRIPS CMP. We use *Sirius* as the simulation platform, which was described in Section 3.3. Performance evaluation focuses on the following two major design metrics.

Effectiveness of run-time thermal management: First and foremost, as an on-line thermal management scheme, ThermalHerd should effectively alleviate thermal emergencies and ensure safe on-line operation.

Two temperature-related parameters are introduced here – network thermal emergency threshold and network thermal trigger threshold. The former is a hard temperature upperbound, which is the maximum allowable network temperature depending on various factors, such as thermal budget and cooling solutions, and timing/reliability issues. For different systems, the chip temperature typically varies from 60°C to 95°C. In mobile applications or application scenarios with tight cooling budgets and space, the thermal budget could reach 105°C. In other applications, such as supercomputing, where cooling cost is not critical, and the mean time to failure of hundreds of parallel computation nodes needs to be maximized, the thermal budget could be lower than 60°C. Here, we set the thermal emergency threshold across a wide temperature range. The latter parameter, thermal trigger threshold, is used to activate ThermalHerd. When the chip peak temperature exceeds the thermal trigger threshold, distributed traffic throttling is enabled and the proactive routing scheme is replaced by the reactive routing scheme. We set the thermal trigger threshold to 1°C lower than the thermal emergency threshold.

Network performance impact: On-chip networks have very tight performance requirements in terms of latency and throughput. Hence, the performance impact of thermal management should be minimal.

The network performance is evaluated as follows. Latency spans the creation of the first flit of the packet to ejection of its last flit at the destination router, including source queuing time and assuming immediate ejection. Network throughput is the total number of packets relayed through the network per unit time.

For comparison purposes, we also introduce the following alternative run-time thermal management techniques.

- **GlobalThermal:** When the temperature exceeds the thermal trigger threshold, all the routers throttle the same percentage of incoming traffic. GlobalThermal is an approximation of the chip-level thermal management technique in Pentium4-M [1], in which, as the processor temperature reaches the temperature limits, the thermal control circuit throttles the processor clock.
- **DistrThermal:** This scheme is equipped with the same distributed traffic throttling technique as ThermalHerd. However, the thermal-correlation based routing algorithms are not enabled. This scheme is used here to differentiate the performance impact between the throttling and routing techniques. Throttling individual functional units has also been proposed for microprocessors – Power5 uses a dual-stage thermal management scheme [6], where in its second stage, temperature reduction is achieved by throttling the processor throughput via individual functions.
- **PowerHerd:** It is the only available architecture-level run-time power management scheme targeting network peak power [20]. Unlike ThermalHerd, PowerHerd is power-aware instead of being thermal-aware. It controls network peak power based on a global power budget that mimics the thermal threshold.

5.1 Evaluation Using TRIPS On-chip CMP Traffic Traces

We evaluate the performance of ThermalHerd using traffic traces extracted from the on-chip operand networks of TRIPS CMP by running a suite of 16 SPEC and Mediabench benchmarks. Since thermal transition is a slow process, to study the accumulated impact of ThermalHerd on both performance and chip temperature, traffic traces generated by different benchmarks are concatenated together, forming a 200ms test trace.

Using Raw [26], TRIPS [16] and the on-chip network proposed in [7], we define a 5×5 on-chip mesh network, as the TRIPS network architecture is currently in design stage. We assume a typical cooling solution here – the silicon die uses flip-chip packaging and is attached to a 15mmx15mm heat spreader and a 30mmx30mm heat sink. The ambient temperature is 25°C. The initial temperature is determined by the average network power consumption over the whole simulation trace. Figure 8 shows the network peak temperature without any thermal management. As different benchmarks have different network workload and traffic patterns, we can see that network peak temperature varies from 70.1°C to 94.0°C along the 200ms simulation.

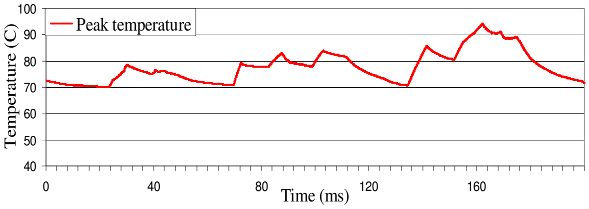


Figure 8. Network peak temperature using the TRIPS traffic trace.

Table 1. Temperature management.

T_T (°C)	89	87	85	83	82	79	77	75
T_A (°C)	86.2	86.1	84.1	82.5	81.3	78.8	76.5	74.7

To evaluate the effectiveness of ThermalHerd in maintaining the chip temperature below the emergency point, we choose eight thermal emergency thresholds, T_T. The simulation results are shown in Table 1. In this table, the first row shows these eight thermal emergency thresholds, the second row shows the actual network peak temperature, T_A, regulated by ThermalHerd. Comparing these two rows, we can see that using ThermalHerd, network run-time temperature is always below the corresponding thermal constraint, which means ThermalHerd can guarantee safe on-line operation.

As shown in Table 1, when T_T is at or below 87°C, the difference between T_T and T_A is always less than 1°C, which implies that the network peak temperature has exceeded the corresponding thermal trigger threshold. Hence, both distributed throttling and reactive routing are enabled. At $T_T = 89$°C, the network peak temperature is below the thermal trigger threshold. Therefore, only proactive routing is enabled in this case. As compared to the peak temperature (94.0°C) when ThermalHerd is disabled, proactive routing alone can reduce the peak temperature by 7.8°C through balancing of the chip thermal profile.

Figure 9 shows network throughput degradation introduced by ThermalHerd under different thermal constraints. The throughput degradation, Thr_{deg}, is defined as follows.

$$Thr_{deg}(t) = (Thr_{init}(t) - Thr_{Thermal}(t))/Thr_{init}(t) \tag{12}$$

where $Thr_{Thermal}(t)$ and $Thr_{init}(t)$ are network run-time throughputs with and without ThermalHerd. From this figure, we have two observations. First, when the thermal threshold is higher than 84.0°C, throughput degradation introduced by ThermalHerd is less than 1%. Therefore, compared to 94.0°C network peak temperature, ThermalHerd reduces network peak temperature by 10°C with negligible performance penalty. This significant improvement is achieved by effective proactive and reactive routing schemes in collaboration with efficient distributed traffic throttling. Second, as the thermal threshold decreases, throughput degradation increases. This indicates that proactive routing alone cannot sufficiently avert the need for throttling which impacts performance.

As we can see, the proactive routing scheme is an effective mechanism to balance the network temperature profile and reduce the network peak temperature, which reduces the need for thermally-induced throttling and hence network throughput degradation. As discussed in Section 4.5, this routing scheme affects network latency. Here, latency overhead, Lat_{ovd}, is defined as follows.

$$Lat_{ovd}(t) = (Lat_{Thermal}(t) - Lat_{init}(t))/Lat_{init}(t) \tag{13}$$

where $Lat_{Thermal}(t)$ and $Lat_{init}(t)$ are network run-time latencies with and without ThermalHerd.

Figure 10 shows the run-time latency impact of ThermalHerd. It shows that the latency overhead introduced by the proactive routing scheme is consistently less than 1.2%.

5.2 Comparison of ThermalHerd Against Alternative Thermal Management Schemes

Next, we seek to isolate the impact of the various features of ThermalHerd – distributed throttling, reactive routing, and proactive routing. Figure 11 shows network throughput degradation under different peak thermal constraints. We compare four schemes: GlobalThermal, DistrThermal (ThermalHerd with only distributed throttling), DistrThermal plus reactive routing, and ThermalHerd (Distributed throttling, reactive and proactive routing). The results show that under the same peak thermal constraints, DistrThermal is much more efficient than GlobalThermal by selectively reducing the traffic with high thermal contribution to thermal hotspots. With reactive routing on top of distributed throttling, the temperature profile is further smoothed and throughput degradation reduces by more than 2X. Finally, with proactive routing, throughput degradation is negligible even at a thermal emergency threshold of 84°C.

We next compare ThermalHerd with PowerHerd. PowerHerd is power-aware instead of thermal-aware. It controls network power consumption based on a global power budget that mimics the thermal threshold. However, under the same global power budget, different traffic can result in very different network temperature profiles. In order to guarantee safe on-line operation, PowerHerd has to assume worst-case power distribution that results in the maximum chip temperature. To obtain such a worst-case power distribution, we partition the TRIPS network trace into slots of 10μs and calculate the average network power consumption for each time slot. With that, we derive the worst-case average power distribution which results in the maximum chip temperature. Figure 12 shows the maximum temperature under the worst-case power distribution as compared to the actual network peak temperature. We can see that worst-case estimation overestimates network peak temperature by about 5°C, which implies PowerHerd will begin to throttle network traffic when the network peak temperature is 5°C lower than the thermal trigger threshold. Therefore, by targeting the temperature directly, ThermalHerd is much more efficient than PowerHerd.

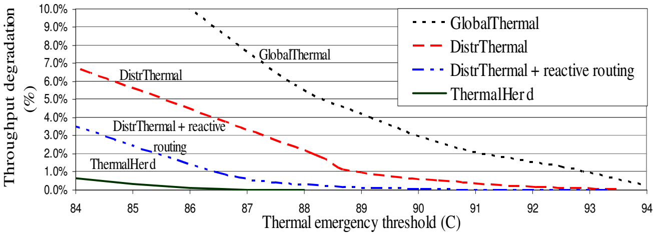


Figure 11. Performance impact of throttling.

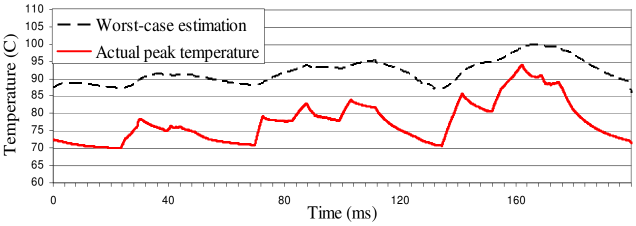


Figure 12. Worst-case temperature estimation by PowerHerd vs. actual temperature profile.

6 From On-chip Networks to Entire On-chip Systems

On-chip systems, such as SoCs and CMPs, consist of computation and storage elements interconnected by on-chip networks. Therefore, the chip temperature is an accumulated effect of the thermal interactions across all on-chip components. The relative thermal contribution of the different components varies depending on the chip architecture as well as application scenarios.

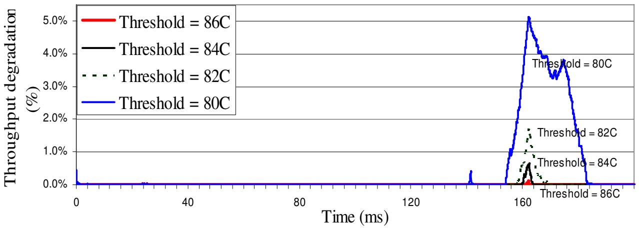


Figure 9. Throughput degradation.

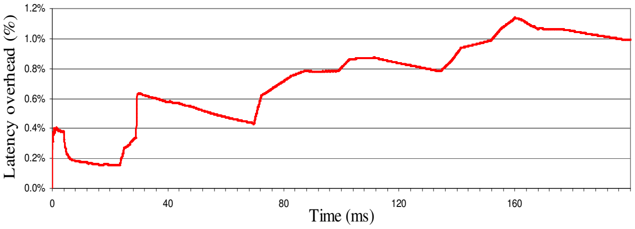


Figure 10. Latency overhead.

The chip architecture determines the complexity of processing vs. storage vs. communication elements and thus the peak power consumption of these elements. A chip with complex processing elements (e.g., wide-issue, multi-threaded) will require larger storage elements (e.g., large multi-level caches, register files) as well as sophisticated communication elements (e.g., multi-level, wide buses, networks with wide link channels, deeply-pipelined routers and significant router buffering). On the other extreme, there are chip architectures where processing elements are single ALUs serviced by a few registers at ALU input/output ports, interconnected with simple single-stage routers with little buffering.

Application characteristics dictate how the above elements are used, thus influencing the power and thermal profile of the chip. Essentially, the amount of computation and communication per data bit affects the relative power and thermal contribution of processing, memory and network elements. Here, we use the MIT Raw chip as a platform for analyzing the absolute and relative thermal impact of all components of a chip. The Raw chip, with its single-issue processing elements, 32KB caches and registers per tile, and networks with fairly narrow 32-bit channels, 8-stage pipelines, and limited router buffering sits in the middle of two possible architectural extremes – a chip where processing elements are fat multi-threaded cores vs. one where processing elements are single ALUs, such as TRIPS cores.

6.1 Thermal Characterization of the Raw Chip

In Raw, the chip temperature is affected by both on-chip processors and networks. In each tile, the processor power is mainly due to instruction and data caches, ALU, register file, fetch, and control logic. For each of these components, the capacitance is obtained from the estimates from an IBM placement tool [9]. Its run-time activities are captured using the Raw Beetle simulator. Based on the chip layout, we extend our network thermal model to on-chip processors by representing each functional component as a thermal block and construct a lumped thermal RC network covering all the power-consuming components in the Raw chip. In Raw, switch memories, and instruction/data caches are synchronous SRAM modules that initiate read operations every clock cycle. Raw proposed a power-aware technique allowing individual SRAM lines to be disabled when not in use. Here, we characterize the chip temperature in both the power-aware and non-power-aware modes.

As shown in Table 2, the current Raw binary distribution contains three sets of benchmarks – SPEC and Mediabench, stream computations, and bit-level computations.

We choose benchmarks from all the three categories and study the thermal behavior of Raw [3]. In order to reveal the thermal impact of the run-time activities of the benchmarks themselves, we first assume the power-aware feature is supported – switch memories are only active when static networks are used; instruction caches are only active during processor execution; data caches are accessed by both load and store operations. Figure 13 shows the thermal characterization of Raw using these benchmarks. For each benchmark, we consider both typical (300lfpm) and best (600lfpm) air-cooling conditions. The chip peak temperature is further characterized under three different power dissipation scenarios – *processor power only*, *network power only*, *and processor plus network power*. These results highlight the following observations:

- First and foremost, the chip temperature is the joint contribution of all on-chip components. As we can see, among all the benchmarks, the processors or networks alone may not tip chip temperature over the thermal emergency point. However, together, the networks and processors can push the chip peak temperature to higher than 100°C.
- The chip temperature is the result of thermal correlations between all on-chip components. For each on-chip component, its thermal contribution is affected by its own power consumption as well as thermal correlation with other components. The former varies with architecture and applications. The latter is determined by the cooling package and physical distance.
- Different benchmarks demonstrate different thermal behavior. Among the three sets of benchmarks, both the stream and bit-level computation benchmarks exhibit excellent scalability – they can effectively utilize on-chip parallel computation and communication resources, leading to a high chip temperature. Due to the limitation of the available compiler (rgcc) and the available ILP in the programs, the ILP in the SPEC and Mediabench benchmarks cannot be explored efficiently. Therefore, these benchmarks result in the on-chip resources being underutilized, thus having a lower thermal impact. The thermal impact of networks vs. processors also varies for the different benchmarks. For instance, in *802.11a enc.*, *8b_10b enc.*, and *fir*, due to the high utilization of the static networks and low access rate of on-chip data caches, the networks alone result in a comparable or higher temperature than the processors. On the other hand, *stream* results in high utilization of its processing resources; *fft* only uses the dynamic network partially. In these two cases, the processor thermal impact is more significant.

Table 2. Benchmarks provided by the Raw binary distribution.

Benchmark set	Description of benchmarks
SPEC & Mediabench	Targets instruction-level parallelism (ILP)
Stream	Targets real-time I/O
Bit-level computations	Targets comparisons with FPGAs and ASICs

Figure 14 shows the chip peak temperature without power-awareness in the SRAM modules. Here, on-chip memories result in a significant power and thermal overhead.

6.2 Extending ThermalHerd to Thermal Management of Entire On-Chip Systems

The observations from the previous section highlight that coordinating and regulating the behavior of all on-chip components is the key to achieving effective thermal management for the entire chip. Since on-chip systems are distributed in nature, the distributed, collaborative nature of ThermalHerd lends readily to the thermal management of the entire chip. We discuss the extension of the two key features of ThermalHerd next – distributed throttling and thermal-correlation based routing.

Distributed joint throttling: For Raw, effective thermal regulation requires jointly considering both the networks and processors. We extend the distributed traffic throttling mechanism in ThermalHerd to distributed joint throttling of processing, storage and network elements in Raw. When the temperature monitors flag a thermal emergency, the hotspot tile begins to throttle both the processor (processing and memory elements) and the network by controlling the grant signal of crossbar, instruction issue logic, and memory disable signals.

Thermal-correlation based placement: Thermal emergencies are often a result of an unbalanced temperature profile. Therefore, workload migration can potentially balance the temperature profile and reduce the peak temperature. The thermal-correlation based routing scheme proposed in Section 4.5 targets network traffic migration. For

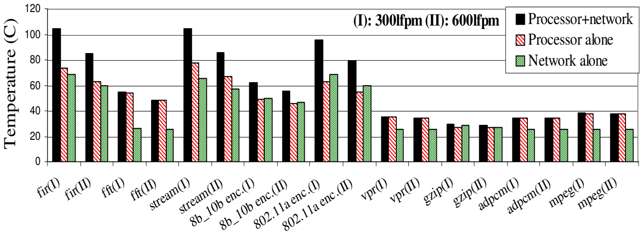


Figure 13. Thermal characterization of Raw.

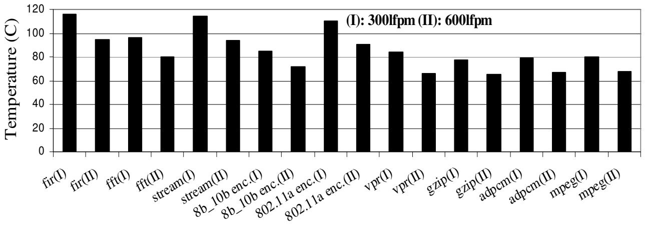


Figure 14. Peak temperature of Raw for various benchmarks, without power-awareness in SRAMs.

[3] As thermal behavior is similar within a benchmark group, we arbitrarily picked just 2-3 benchmarks in each group to highlight differences across groups.

processors, previous work has explored computation migration in CMPs to improve the performance with a migration interval in the range of tens of microseconds [21], which matches the thermal time constant of on-chip thermal hotspots. Therefore, computation migration can be used to track and balance run-time thermal variations for on-chip computation resources – scheduler/OS dispatches computation jobs based on the thermal correlation matrix of processing elements to balance the run-time chip thermal profile.

Preliminary investigations: Concurrent tasks running on a parallel architecture, such as Raw, are more or less logically correlated. The inter-tile program correlation has a significant impact on run-time thermal management. First, distributed throttling minimizes performance degradation by only throttling those tiles that have the highest thermal impact locally. However, the inter-tile program correlation spreads the localized throttling effect to other tiles, thus degrading the throttling efficiency and overall performance. Second, to balance the chip thermal profile, task placement needs to avoid adjacent tiles to minimize the inter-tile thermal impact. This increases not only the latency but also the power and thermal overhead for supporting data communication among correlated tasks.

We explore the advantages and limitations of the above thermal management techniques. We design synthetic benchmarks to emulate two typical CMP applications:

- *Benchmark I:* Tasks running on different tiles are independent, which emulates server-like workloads – on-chip resources support multiple independent applications for different users.
- *Benchmark II:* All tiles form a tightly coupled computation pipeline stream, which emulates multimedia streaming applications.

As we can see, these two benchmarks are the two extreme cases in terms of inter-tile program correlation.

First, we evaluate the performance of distributed joint throttling (DJT). We compare it with global throttling (GT), in which when a thermal emergency occurs, all the tiles are throttled in the same fashion. Figure 15 shows the performance degradation for these two techniques. For GT, each tile throttles 5% to 30% of system throughput, which is defined as the overall finished workload. The x-axis shows the corresponding chip peak temperature reduction. As shown in the figure, for Benchmark I, to achieve the same temperature reduction, DJT results in much lower performance degradation by more efficiently throttling traffic on those tiles that have a higher thermal impact on thermal hotspots. As the required temperature reduction increases further, more tiles need to be throttled in order to achieve enough temperature reduction. Hence, the gain of DJT reduces. For Benchmark II, since all the tiles are tightly correlated, throttling any individual tile results in the same performance impact on all the other tiles. Therefore, DJT has the same performance impact as GT.

To evaluate the impact of thermal-correlation based placement on inter-tile program correlation, for each benchmark, we reduce the number of parallel tasks to four. As shown in Figure 16, for both benchmarks, initially, four tasks are placed in the four center tiles in Raw. To balance the thermal profile, these four tasks are moved to the corner tiles. For Benchmark I, task reallocation effectively balances the chip thermal profile and reduces the chip peak temperature by 13.8%. For Benchmark II, task reallocation also reduces the chip thermal gradient. However, the extra communication power results in a significant power and thermal overhead, and increases the overall chip temperature. As a result, the chip peak temperature increases by 6.9%. In this case, placing the tasks as in Figure 16 achieves a temperature reduction of 3.5% by reducing the inter-task thermal correlation and avoiding the extra communication overhead.

7 Discussion and Related Work

We next present discussions and related work.

Interconnection networks: Substantial research has explored power consumption issues in interconnection networks. Patel *et al.* [15] first developed power modeling techniques for routers and links. Wang *et al.* [28] developed an architecture-level power model, called Orion, for interconnection networks. In our work, we have integrated Orion into our network evaluation platform. For design optimization, most prior works used circuit-level techniques to improve the power efficiency of the link circuitry, such as low-swing on-chip links. Recently, power-efficient on-chip network design has also been addressed at the microarchitecture level [27]. Power-aware techniques, such as dynamic voltage scaling and dynamic power management, have been proposed [11, 19] to minimize the power consumption in the link circuitry. All the above techniques target average power, not peak power.

Recently, a dynamic power management scheme, called PowerHerd [20], has been proposed to address network peak power issues. However, PowerHerd is power-aware instead of being thermal-aware. Even though power and temperature are correlated, they are still fundamentally different in nature. Thus, as demonstrated in the experimental results section, PowerHerd cannot address network thermal issues efficiently.

Microprocessors: Power and thermal concerns in modern processors have led to significant research efforts in power-aware and temperature-aware computing. Brooks *et al.* [3] first proposed a dynamic thermal management scheme. Skadron *et al.* [22] further explored control-theoretic techniques for this purpose. They also developed an architecture-level thermal model, called HotSpot. In our work, we constructed a thermal model for on-chip networks, which is an improvement over HotSpot. Recently, Srinivasan *et al.* [24] used a predictive dynamic thermal management scheme targeting multimedia applications.

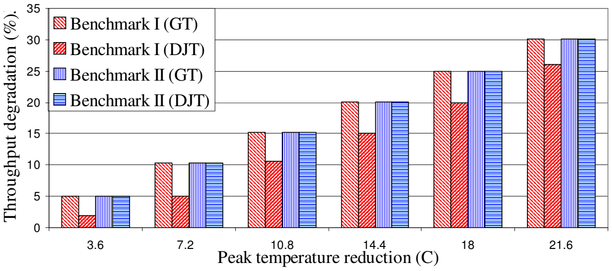


Figure 15. Distributed joint throttling vs. global throttling.

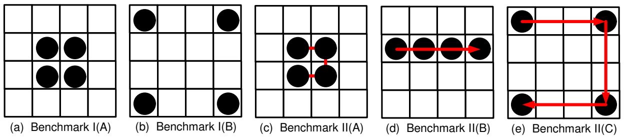

(a) Benchmark I(A) (b) Benchmark I(B) (c) Benchmark II(A) (d) Benchmark II(B) (e) Benchmark II(C)

Figure 16. Thermal-correlation based placement.

8 Conclusions

Power and cooling costs are critical constraints in high-performance on-chip systems. With networks replacing on-chip buses and becoming the pervasive on-chip interconnection fabric in SoCs and CMPs, we need to seriously address network thermal issues to guide on-chip network design and improve its thermal efficiency.

In this work, we built an architecture-level thermal model, and constructed an architecture-level platform for jointly evaluating the network performance, power, and thermal profile. We then characterized the computation and communication thermal impact in the MIT Raw CMP and revealed the importance of jointly considering both networks and processors for efficient thermal design of on-chip systems. To overcome the deficiencies of designing for the worst-case thermal signature, we proposed a distributed on-line thermal management scheme, called ThermalHerd, which can dynamically regulate the network temperature profile and guarantee safe on-line operation. Finally, using Raw as a testbed, we explored extensions of our work to address the thermal issues for entire on-chip systems.

This work is the first study addressing thermal issues in on-chip networks. We hope this work will lead to thermal-efficient network design, enabling network designers to build temperature-aware high-performance network microarchitectures. In this paper, we also illustrated how the distributed, collaborative nature of on-chip networks leads to distinct similarities with distributed on-chip systems and showed how thermal management of on-chip networks can be extended to entire on-chip systems. We see this work forming the foundation for studies of complete networked on-chip processing systems in the future.

Acknowledgments

The authors would like to thank Kevin Skadron and Wei Huang of University of Virginia for their help in our understanding of HotSpot. We wish to thank the MIT Raw group, especially Michael B. Taylor, for the help with the Raw simulation platform as well as providing us with critical packaging and power information of their chip. We also wish to thank Doug Burger of University of Texas at Austin for supplying us with TRIPS network traffic traces. In addition, we wish to thank Howard Chen of IBM for helping us validate our thermal model. This work was supported in part by Princeton University's Wallace Memorial Honorific, National Science Foundation under Grant No. CCR-0237540 (CAREER) and CCR-0324891, as well as the MARCO Gigascale Systems Research Center.

References

[1] Mobile Intel Pentium 4 processor – M datasheet. `http://www.intel.com`.

[2] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–78, Jan. 2002.

[3] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. Int. Symp. High Performance Computer Architecture*, pages 171–182, Jan. 2001.

[4] H. Chen. IBM, personal communication.

[5] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat. Analytical thermal model for multilevel VLSI interconnects incorporating via effect. *IEEE Electron Device Letters*, 23(1):31–33, Jan. 2002.

[6] J. Clabes *et al.* Design and implementation of the POWER5 microprocessor. In *Proc. Int. Solid-State Circuits Conf.*, pages 56–57, Jan. 2004.

[7] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. Design Automation Conf.*, pages 684–689, June 2001.

[8] FEMLAB User Manual. `http://www.femlab.com`.

[9] J. S. Kim. Power consumption of Raw networks. Technical report, CSAL/LCS, MIT, 2003.

[10] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff. Energy characterization of a tiled architecture processor with on-chip networks. In *Proc. Int. Symp. Low Power Electronics and Design*, pages 424–427, Aug. 2003.

[11] E. J. Kim *et al.* Energy optimization techniques in cluster interconnects. In *Proc. Int. Symp. Low Power Electronics and Design*, pages 459–464, Aug. 2003.

[12] D. Meeks. Fundamentals of heat transfer in a multilayer system. *Microwave Journal*, 1(1):165–172, Jan. 1992.

[13] N. B. Nguyen. Properly implementing thermal spreading will cut cost while improving device reliability. In *Proc. Int. Symp. Microelectronics*, pages 385–386, Oct. 1996.

[14] R. H. Otten and R. K. Brayton. Planning for performance. In *Proc. Design Automation Conf.*, pages 122–127, June 1998.

[15] C. Patel, S. Chai, S. Yalamanchili, and D. Schimmel. Power-constrained design of multiprocessor interconnection networks. In *Proc. Int. Conf. Computer Design*, pages 408–416, Oct. 1997.

[16] K. Sankaralingam *et al.* Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture. In *Proc. Int. Symp. Computer Architecture*, pages 422–433, June 2003.

[17] J. E. Sergent and A. Krum. *Thermal Management Handbook for Electronic Assemblies*. McGraw-Hill Press, 1998.

[18] M. Sgroi *et al.* Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proc. Design Automation Conf.*, pages 667–772, June 2001.

[19] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proc. Int. Symp. High Performance Computer Architecture*, pages 79–90, Feb. 2003.

[20] L. Shang, L.-S. Peh, and N. K. Jha. PowerHerd: Dynamically satisfying peak power constraints in interconnection networks. In *Proc. Int. Conf. Supercomputing*, pages 98–108, June 2003.

[21] K. A. Shaw and W. J. Dally. Migration in single chip multiprocessors. *Computer Architecture Letters*, 1, 2002.

[22] K. Skadron *et al.* Temperature-aware microarchitecture. In *Proc. Int. Symp. Computer Architecture*, pages 1–12, June 2003.

[23] S. Song, S. Lee, and V. Au. Closed-form equation for thermal constriction/spreading resistances with variable resistance boundary condition. In *Proc. Int. Electronics Packaging Conf.*, pages 111–121, May 1994.

[24] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *Proc. Int. Conf. Supercomputing*, pages 109–120, June 2003.

[25] M. B. Taylor. MIT, personal communication.

[26] M. B. Taylor *et al.* Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams. In *Proc. Int. Symp. Computer Architecture*, June 2004.

[27] H.-S. Wang, L.-S. Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proc. Int. Symp. Microarchitecture*, pages 105–116, Nov. 2003.

[28] H.-S. Wang, X.-P. Zhu, L.-S. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *Proc. Int. Symp. Microarchitecture*, pages 294–305, Nov. 2002.

[29] L.-T. Yeh and R. C. Chu. *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*. ASME Press, New York, NY, 2002.