Thermal Modeling and Management of Liquid-Cooled 3D Stacked Architectures

Ayşe Kıvılcım Coşkun[1], José L. Ayala[2],
David Atienza[3], and Tajana Simunic Rosing[4]

[1] Boston University, Boston, MA 02215, USA,
acoskun@bu.edu
[2] Complutense University of Madrid, Spain
jayala@fdi.ucm.es
[3] Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
david.atienza@epfl.ch
[4] University of California, San Diego, CA 92039, USA
tajana@ucsd.edu

Abstract. 3D stacked architectures are getting increasingly attractive as they improve yield, reduce interconnect power and latency, and enable integrating layers manufactured with different technologies on the same chip. However, 3D integration results in higher temperatures following the increase in thermal resistances. This chapter discusses thermal modeling and management of 3D systems with a particular focus on liquid cooling, which has emerged as a promising solution for addressing the high temperatures in 3D systems. We first introduce a framework that is capable of detailed thermal modeling of the interlayer structure containing microchannels and through-silicon-vias (TSVs). For energy-efficient liquid cooling, we describe a controller to adjust the liquid flow rate to meet the current chip temperature. We also discuss job scheduling techniques for balancing the temperature across the 3D system to maximize the cooling efficiency and to improve reliability.

1 Introduction

3D integration is a recently proposed design method for overcoming the limitations regarding the delay and power consumption of the interconnects. However, this increased level of integration also results in new limitations and design challenges, including the challenges related to higher temperatures. A *k-tier* 3D chip could potentially use k times as much current as a single 2D chip of the same footprint, while utilizing similar packaging technology due to cooling cost limitations. The implications of this observation are:

– The 3D stacked systems will likely consume more power than their 2D counterparts, and the heat generated as a result of the power consumption must be removed from the system. Unless the 3D chip design has been optimized with thermally-aware techniques, considering that the package characteristics of the 3D system are similar to those of 2D chips, on-chip temperatures for 3D chips will be higher than temperatures on 2D chips.

– Stacking layers vertically increase the thermal resistances on a given unit. Therefore, it is more difficult to remove heat from the chip, especially for the layers that are further away from the cooling infrastructures. This situation further escalates the temperature-induced challenges.
– Elevated temperatures and large thermal gradients degrade performance and reliability of chips. Reliability issues in 3D stacks will also be aggravated because of the higher temperatures and presence of mechanical stress. Therefore, on-chip thermal management is a critical issue in 3D design.

Liquid cooling is a potential solution to address the high temperatures in 3D chips, due to the higher heat removal capability of liquids in comparison to air. Liquid cooling is performed by attaching a cold plate with built-in microchannels, and/or by fabricating microchannels between the layers of the 3D architecture. Then, a coolant fluid (i.e., water or other fluids) is pumped through the microchannels to remove the heat. The heat removal performance of this approach, called interlayer cooling [1], scales with the number of tiers. The flow rate of the pump can be altered dynamically, but as there is a single pump connected to the system, the flow rates among the channels are the same—assuming identical channel dimensions. One obvious way to set the flow rate is by matching it with the worst-case temperature. However, the pump power increases quadratically with the increase in flow rate [1], and its contribution to the overall system power is significant. Also, over-cooling may cause dynamic fluctuations in temperature, which degrade reliability and cooling efficiency. Through runtime system analysis and intelligent control of the flow rate, it is possible to determine the minimum flow rate to remove the heat and maintain a safe system temperature. In addition, by maintaining a target temperature value throughout the execution, we can minimize the temperature variations. Note that, while reducing the coolant flow rate, it is necessary to maintain the temperature at a level where the temperature-dependent leakage power does not revert the benefits achieved with lower-power pumping.

Current technology enables fabricating the infrastructures required for interlayer liquid cooling. IBM Zurich Research Laboratory has built a 3D chip with multiple microchannels that allow water flow (see Figure 1). The 50 μm channels between individual chip layers are able to cool with a rate of 180 $watt/cm^2$ per layer for a stack with a footprint of $4cm^2$ [2].

3D systems have an inherent temperature imbalance among the various processing units, due to the change in thermal resistance that is a function of the location of the unit. Cores located at different layers or at different coordinates across a layer may have significantly different rates for heating and cooling [3]. Therefore, even when we select the appropriate energy-efficient flow rate for the coolant in a 3D liquid-cooled system, large temperature gradients across the system may still exist. Conventional multicore schedulers, e.g., dynamic load balancing, do not consider such thermal imbalances. To address this issue, we discuss temperature-aware load balancing, which weighs each core's workload with the core's thermal properties and uses this weighted computation to balance the temperature. The highlights of this chapter are the following:

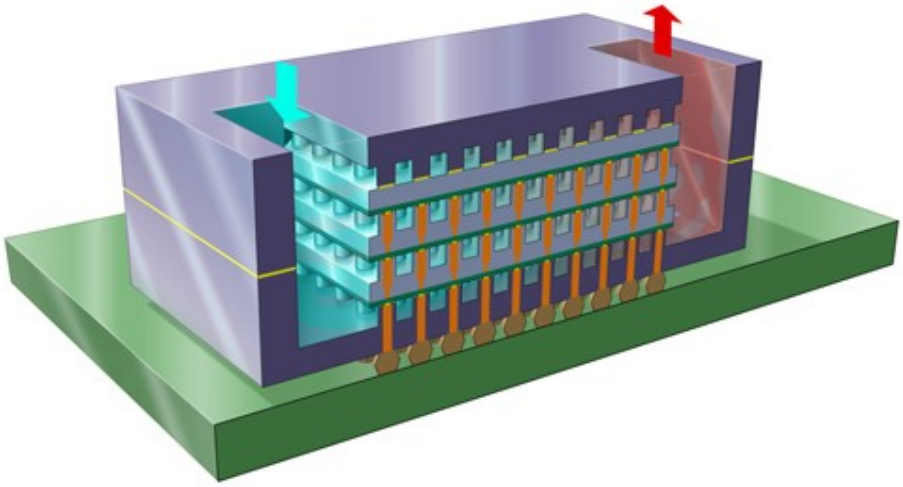

Fig. 1. 3D chip with microchannels for liquid cooling (by IBM)

- We show in detail how to model the effects of the liquid flow on temperature. Our model is based on the liquid cooling work of IBM [1]. The liquid cooling model (introduced in [4, 5]) includes a fine-grained computation of the heat spread and takes into account the effects of TSVs and microchannels. We use model parameters verified by finite element simulation, and integrate our modeling infrastructure in HotSpot [6] for ease of use.
- We describe a controller for adjusting the liquid flow rate dynamically to maintain a target temperature while minimizing the pump power consumption. Our controller forecasts maximum system temperature, and uses this forecast to proactively set the flow rate. This way, we avoid over- or under-cooling due to delays in reacting to the temperature changes.
- We integrate the controller with a job scheduler that computes the current workload of each core as a function of the core's thermal properties. The scheduler addresses the inherent thermal imbalances in multicore 3D systems and reduces the frequency of large thermal gradients.
- On the 2- and 4-layered 3D systems that we simulate, we see that our method achieves up to 30% reduction in cooling energy, and 12% reduction in system-level energy in comparison to setting the flow rate at the maximum value, while we maintain the target temperature. We also show that temperature-aware load balancing reduces the hot spots and gradients significantly better than load balancing or reactive thread migration.

The rest of this chapter starts with an overview of the prior art. Section 3 describes the thermal model for 3D systems with liquid cooling. In Section 4, we provide the details of the flow rate controller and job scheduler. The experimental results are in Section 5, and Section 6 concludes the chapter.

2 Related Work

2.1 Thermal Modeling and Management

Accurate thermal modeling is critical in the design and evaluation of systems and policies in 3D systems. There has been abundant work on design-time full-chip thermal models [7, 6, 8, 9]. However, existing studies do not provide flexibility on thermal package modeling. HotSpot [6] is an automated thermal model, which calculates transient temperature response given the physical and power consumption characteristics of the chip. To reduce simulation time even for large multicore systems, a thermal emulation framework for FPGAs is proposed in [10]. In such thermal models the typical packaging configuration is forced air convection with a heat sink and/or spreader.

In addition to simulation frameworks for thermal modeling, there are existing studies on runtime thermal characterization methods. For example [11, 12] provide insights into using on-chip temperature sensors in processors that contain integrated sensors, such as IBM POWER series processors. However, many of the hot spots can be missed as the number of sensors is limited. Another runtime thermal characterization method is IR thermal imaging [13, 14]. While this technique can capture the detailed thermal map in real-time, the limited sampling rate of the IR camera may filter out high-frequency transient thermal fluctuations.

Dynamic thermal management in response to thermal measurements in microprocessors has been first introduced by Brooks et al. [15], where the authors explore performance trade-offs among various dynamic thermal management mechanisms. Activity migration [16] and fetch toggling [6] are other examples of dynamic management techniques. Kumar et al. propose a hybrid method that combines clock gating and software thermal management [17]. The multicore thermal management method introduced by Donald et al. [18] combines distributed DVS with process migration; while Chaparro et al. [19] investigate thermal management techniques for multicore systems. For multicore systems, temperature-aware task scheduling [20] shows a lot of potential to achieve desirable thermal profiles at low performance cost. Li et al. [21] and Monchiero et al. [22] consider the thermal constraints in multicore systems at a detailed microarchitecture level with comprehensive architecture simulations for multi-programmed and multi-threaded workloads, respectively. For manycore architectures, Huang et al. [23] look at a heat-spreading floorplanning approach to increase the power envelope of symmetric manycore chips without running into thermal violations.

2.2 Design, Modeling, and Management of 3D Systems

The fabrication technology for manufacturing 3D systems determines many of the electrical, architectural, and thermal characteristics of the final stack. Various 3D fabrication technologies have been proposed in the recent years. For example, prior work [24, 25, 26] proposes diverse fabrication technologies. Two commonly

used fabrication technologies are: die-bonding and Multi-Layer Buried Structures (MLBS). Die-bonding process employs conventional 2D fabrication processes and metal vias to bond the planar die vertically [27]. Figure 2 (a) shows a conventional planar IC modeled as five layers: metal layers (A), active silicon (B), bulk silicon (C), heat spreader (D) and heat sink (E). The heat spreader is attached to the bulk silicon with a thermal interface material. Figure 2 (b) shows a 2-die 3D IC built with two planar dies stacked with their metal layers face-to-face (F2F). In MLBS technology [28] it is possible to stack many heterogeneous dies to mix dissimilar process technologies such as high-speed CMOS with high-density DRAM [29, 30]. The MLBS approach (shown schematically in Figure 3) combines dual Damascene process for in-plane and out-plane interconnects, chemical-mechanical polishing of bondable roughness, and a critical low temperature layering step in order to achieve the three-dimensional structures.

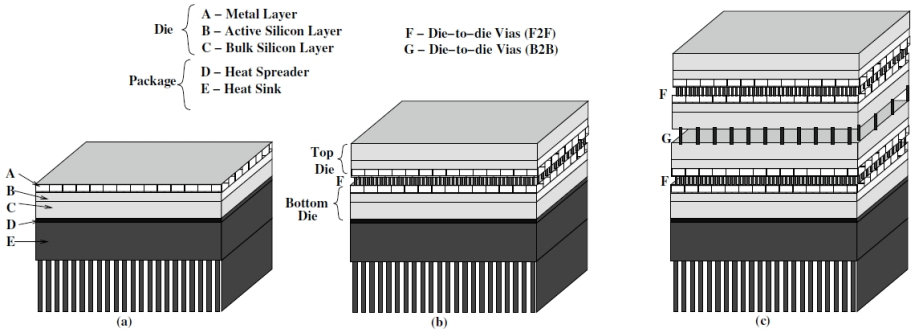

Fig. 2. Die-bonding process [31].

There are three different stacking topologies for interfacing multiple planar dies: face-to-face (F2F), face-to-back (F2B) and back-to-back (B2B). These topologies have different quality and pitch of the die-to-die (D2D) vias at the interfaces and thus influence the benefits obtained from building a 3D IC. For the 2-die 3D IC with the F2F topology shown in Figure 2 (b), D2D vias are etched and deposited on top of the metal layer of each of the planar dies using conventional metal etching technology. Therefore, the via pitch can be as dense as regular on-die interconnects, and the realizable pitch is only limited by the accuracy of aligning the two dies. The die-to-die via interface is densely populated since the vias are required as the physical bonding mechanism independent of whether they actually carry a signal. The bulk silicon layer of the top die is usually thinned down with chemical-mechanical polishing down allowing low impedance backside vias to be etched through, which provide I/O and power/ground connections.

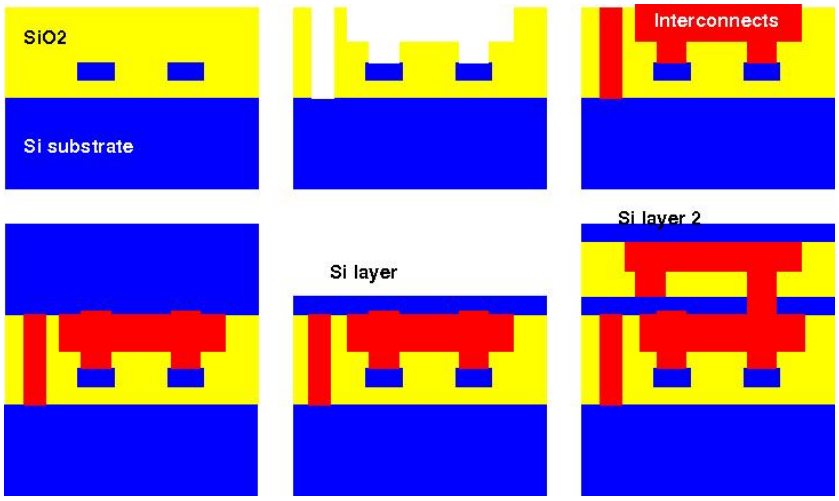

Fig. 3. MLBS process. [32]

Most of the prior work in thermal management of 3D systems addresses design stage optimization, such as thermal-aware floorplanning (e.g. [33]) and integrating thermal via planning in the 3D floorplanning process [34]. In [35], the authors evaluate several policies for task migration and DVS. A recent paper proposes a temperature-aware scheduling method specifically for air-cooled 3D systems [3]. This method takes into account the thermal heterogeneity among the different layers of the system.

The use of convection in microchannels to cool down high power density chips has been an active area of research since the initial work by Tuckerman and Pease [36]. Their liquid cooling system can remove 1000 W/cm^2; however, the volumetric flow rate and the pressure drop are large. More recent work shows how back-side liquid cold plates, such as staggered microchannel and distributed return jet plates, can handle up to 400 W/cm^2 in single-chip applications [37]. The heat removal capability of interlayer heat-transfer with pin-fin in-line structures for 3D chips is investigated in [1]. At a chip size of 1 cm^2 and a $\Delta T_{jmax-in}$ of 60 K, the heat-removal performance is shown to be more than 200 W/cm^2 at interconnect pitches bigger than 50 μm. Previous work in [38, 39] describe how to achieve variable flow rate for the coolant. Finally, in a recent work by Jang et al. [40], the authors evaluate the architectural effects (temperature, leakage, and reliability) of the direct interlayer cooling method for 3D integrated processors, where the dielectric coolant flows in-between individual dies. The evaluation shows that this liquid cooling scheme significantly reduces on-chip temperature under 350K, which completely eliminates thermal emergencies. The temperature reduction also leads to more than 10% leakage reduction of the 3D integrated processor.

Prior work on liquid-cooled 3D systems [4] evaluates existing thermal management policies on a 3D system with a fixed-flow rate setting, and also investigates the benefits of variable flow using a policy to increment/decrement the flow rate based on temperature measurements, without considering energy consumption. The follow-up work in [5] proposes a controller design to provide sufficient coolant flow to the system with minimal cooling energy. The runtime management policy combines this controller with a job scheduler to reduce thermal gradients, and further improves the cooling efficiency without affecting performance.

3 Modeling Framework for Liquid-Cooled 3D Systems

Modeling the temperature dynamics of 3D stacked architectures with liquid cooling consist of: (A) Forming the grid-level thermal R-C network, (B) Detailed modeling of the interlayer material between the tiers, including the through-silicon-vias (TSVs) and the microchannels, and (C) Modeling the pump and the coolant flow rate. We assume forced convective interlayer cooling with water [1] in this chapter, but the model can be extended to other coolants as well.

Figure 4 shows the 3D systems targeted in this chapter. A target system consists of two or more stacked layers (with cores, L2 caches, crossbar, and other units for memory control, buffering, etc.), and microchannels are built in between the vertical layers for liquid flow. The crossbar contains the TSVs that provide the connection between the layers. The microchannels, which are connected to an impeller pump (such as [41]), are distributed uniformly, and fluid flows through each channel at the same flow rate. The liquid flow rate provided by the pump can be dynamically altered at runtime. In the rest of this section, we provide the details of the thermal modeling infrastructure that we developed for the 3D system.

3.1 Grid-Level Thermal Model for 3D Systems with Liquid Cooling

Similar to thermal modeling in 2D chips, 3D thermal modeling is performed using an automated model that forms the R-C circuit for given grid dimensions. In this work, we utilize HotSpot v.4.2. [6], which includes 3D modeling capabilities. The existing model in HotSpot considers the interlayer material between two stacked layers as a layer with homogeneous thermal characteristics, represented by a thermal resistivity and a specific heat capacity value. The extension we have developed for the multi-layered thermal modeling provides a new interlayer material model to include the TSVs and the microchannels.

In a typical automated thermal model, the thermal resistance and capacitance values of the blocks or grid cells are computed initially at the start of the simulation, assuming that the system properties do not vary at runtime. To model the heterogeneous characteristics of the interlayer material including the TSVs and microchannels, we introduce two novelties: (1) As opposed to having a uniform thermal resistivity value of the layer, our infrastructure enables

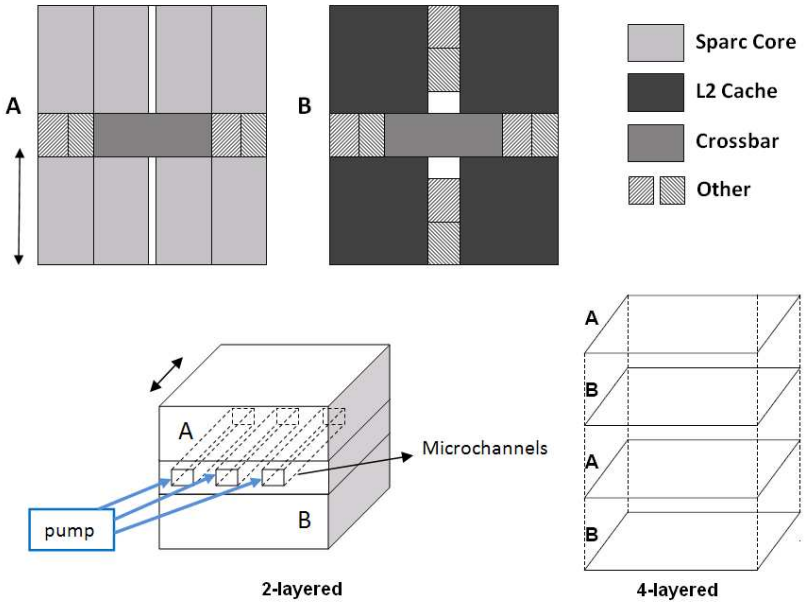

Fig. 4. Floorplans of the 3D Systems.

having various resistivity values for each grid cell, (2) The resistivity value of the cell can vary at runtime. Item (1) enables distinctly modeling TSVs, the microchannels, and the interlayer material, while item (2) enables modeling the liquid coolant and dynamically changing flow rate. Thus, the interlayer material is divided into a grid, where each grid cell except for the cells of the microchannels has a fixed thermal resistance value depending on the characteristics of the interface material and TSVs. The thermal resistivity of the microchannel cells is computed based on the liquid flow rate through the cell, and the characteristics of the liquid at runtime. We use grid cells of $100\mu m$ x $100\mu m$ in our experiments.

In a 3D system with liquid cooling, we compute the local junction temperature using a resistive network, as shown in Figure 5. In this figure, the thermal resistance of the wiring layers (R_{BEOL}), the thermal resistance of the silicon slab (R_{slab}), and the convective thermal resistance (R_{conv}) are combined to model the 3D stack. In the figure, the heat flux values ($\dot{q}$) represent the heat sources. This R-network can be solved to get the junction temperature (T_j). Note that the figure shows the heat sources and the resistances of only one layer, and heat will be dissipated to both opposing vertical directions (i.e., up and down) from the heat sources. For example, if there is another layer above the two heat-dissipating layers shown in the figure, $\dot{q}_1$ will also be dissipating heat towards the upper stack. Also, the network in Figure 5 is a simplification and it assumes isothermal channel walls; i.e., top and bottom of the microchannel have the same temperature.

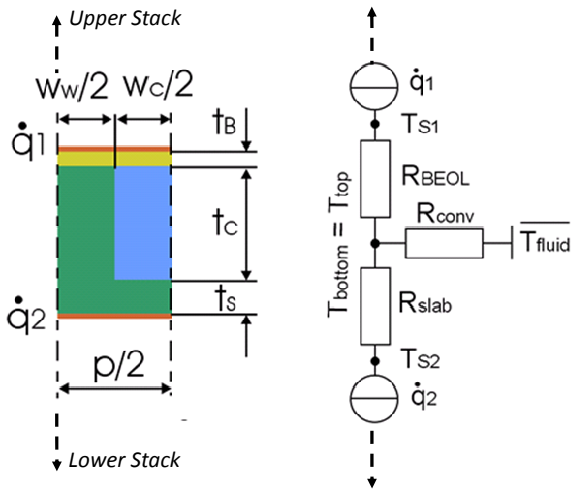

Fig. 5. Cross section of the 3D layers and the resistive network.

The typical junction temperature (T_j) response at uniform chip heat flux and convective cooling is a sum of the following three components: (1) The thermal gradient due to conduction (ΔT_{cond}); (2) the coolant temperature, which increases along the channel due to the absorption of sensible heat (ΔT_{heat}); and (3) the convective (ΔT_{conv}) portion, which increases until fully developed hydrodynamic and thermal boundary layers have been reached [1]. The total temperature rise on the junction, ΔT_j, is computed as the following:

$$\Delta T_j = \Delta T_{cond} + \Delta T_{heat} + \Delta T_{conv} \tag{1}$$

Thermal gradient due to heat conduction through the BEOL layer, ΔT_{cond} is computed with Equations 2 and 3. Note that ΔT_{cond} is independent of the flow rate. Figure 5 demonstrates t_B, and k_{BEOL} is the conductivity of the wiring layer.

$$\Delta T_{cond} = R_{th-BEOL} \cdot \dot{q}_1 \tag{2}$$

$$R_{th-BEOL} = \frac{t_B}{k_{BEOL}} \tag{3}$$

Temperature change due to absorption of sensible heat is computed using Equations 4 and 5. A_{heater} is the area of the heater (i.e., total area consuming power), c_p is the heat capacity of the coolant, ρ is the density of the coolant, and $\dot{V}$ is the volumetric flow rate in the microchannel (in l/min).

Equations 4 and 5 are valid for uniform power dissipation. For the general case, heat absorption in the fluid is calculated iteratively along the channel: $\Delta T_{heat}(n+1) = \sum_{i=1}^{n} \Delta T_{heat}(i)$, where n is the position along the channel.

$$\Delta T_{heat} = (\dot{q_1} + \dot{q_2}) \cdot R_{th-heat} \tag{4}$$

$$R_{th-heat} = \frac{A_{heater}}{c_p \cdot \rho \cdot \dot{V}} \tag{5}$$

Finally, Equation 7 shows how to calculate ΔT_{conv}. Note that ΔT_{conv} is independent of flow rate in case of developed boundary layers. h is dependent on hydraulic diameter, Nusselt number, and conductivity of the fluid [1]. As ΔT_{conv} is not affected by the change in flow rate, we compute this parameter prior to simulation and use a constant value during experiments. Figure 5 demonstrates w_c, t_c, and p parameters on the cross-section of the 3D system.

$$\Delta T_{conv} = (\dot{q_1} + \dot{q_2}) \cdot h_{eff} \tag{6}$$

$$h_{eff} = h\frac{2 \cdot (w_c + t_c)}{p} \tag{7}$$

The equations above give the ΔT_j for the unit cell shown in Figure 5; thus, we extend the computation to model multiple layers and multiple cells as well.

Table 1 lists the parameters used in the computations, and provides the values for the constants, which are taken from prior liquid cooling work [1]. Note that the flow rate ($\dot{V}$) range provided in the table is per cavity (i.e., the interlayer cavity consisting of all the microchannels in one layer), and this flow is further divided into the microchannels.

We compute the flow rate dependent components whenever the flow rate changes. Heat flux, $\dot{q}$ (W/cm^2), values change as the power consumption changes. Instead of reacting to every instant change in power consumption of the cores, we re-compute the $\dot{q}$ values periodically to reduce the simulation overhead.

Considering the dimensions and pitch requirements of microchannels and TSVs, we assume there are 65 microchannels in between each two layers (in each cavity), and there are cooling layers on the very top and the bottom of the stacks. Thus, there are 195 and 325 microchannels in the 2- and 4-layered systems, respectively.

In our target systems shown in Figure 4, we assume the TSVs are located within the crossbar. Placing the TSVs in the central section of the die provides an advantage on the thermal design as well, as TSVs reduce the temperature due to the low thermal resistivity of Cu. We assume there are **128 TSVs** within the crossbar block connecting each two layers. Feasible TSVs for microchannels of $100\mu m$ height and $100\mu m$ pitch have a minimal pitch of $100\mu m$ as well due to aspect ratio limits. We assume each TSV occupies a space of $50\mu m$x$50\mu m$, and the TSVs have a minimum spacing requirement of $100\mu m$.

Previous work has studied granularity and accuracy of TSV modeling [4]. The study shows that using a block-level granularity for TSVs, i.e., assigning

Table 1. Parameters for computing Equation 1

Parameter	Definition	Value
$R_{th-BEOL}$	Thermal resistance of wiring levels	Eqn.(3) $5.333\ (K \cdot mm^2)/W$
t_B	See Figure 5	$12\mu m$
k_{BEOL}	Conductivity of wiring levels	$2.25W/(m \cdot K)$
$R_{th-heat}$	Effective thermal resistance	Eqn.(5)
A_{heater}	Heater area	Area of grid cell
c_p	Coolant heat capacity	$4183J/(kg \cdot K)$
ρ	Coolant density	$998kg/m^3$
$\dot{V}$	Volumetric flow rate	0.1-1 l/min per cavity
h	Heat transfer coefficient	$37132W/(m^2 \cdot K)$
w_c	See Figure 5	$50\mu m$
t_c	See Figure 5	$100\mu m$
t_s	See Figure 5	$50\mu m$
p	See Figure 5	$100\mu m$

a TSV density to each block based on the functionality of the unit, constitutes a reasonable trade-off between accuracy and simulation time. Thus, based on the TSV density of the crossbar, we compute the joint resistivity of that area combining the resistivity values of interlayer material and Cu. We do not alter the thermal resistivity values for the regions without TSVs or microchannels. We assume that the effect of the TSV insertion to the heat capacity of the interface material is negligible, which is a reasonable assumption considering the total area of TSVs constitutes a very small percentage of the total area of the material.

3.2 Modeling the Pump and Liquid Flow Rate

All the microchannels are connected to a pump to receive the coolant. We assume a 12V DC-pump, *Laing DDC* [41], which has suitable dimensions, flow rates, and power consumption for this type of liquid cooling. The power consumption of the pump across the *five* flow rate settings we use is shown in Figure 6 (right y-axis). The pressure drop for these flow rates changes between 300-600 mbar [41]. We assume that the total flow rate of the pump is equally distributed among the cavities, and among the microchannels. DC pumps typically have low efficiency. Also, the flow rate in the microchannels further decreases because the the pressure drop in the small microchannels is larger than its value in the pump output channel. In this work, we assume a global reduction in the flow rate by 50% to account for the loss due to all of these factors. In Figure 6, we show the per cavity flow rates for the 2- and 4-layered 3D systems after applying the reduction factor.

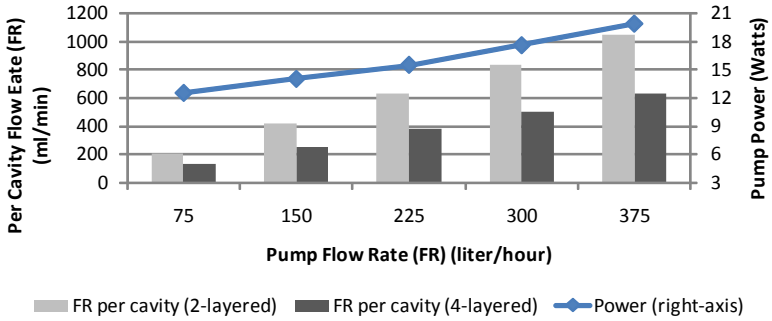

Fig. 6. Power consumption and flow rates of the pump (based on [41]). Per cavity flow rates reflect 50% efficiency assumption.

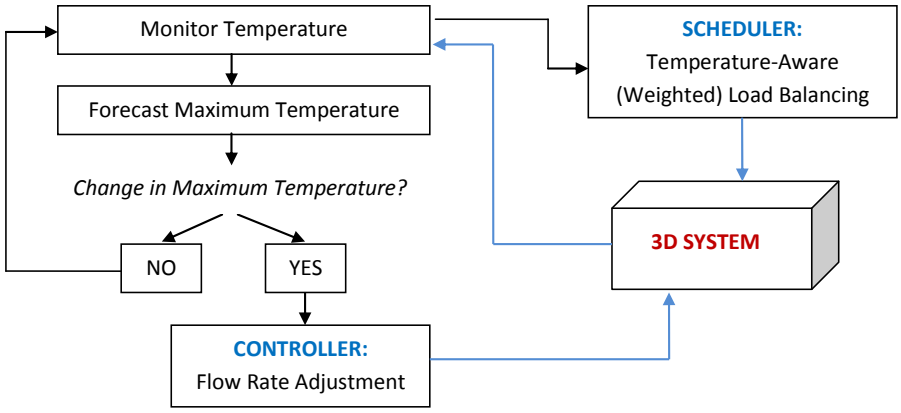

Fig. 7. Overview of the technique.

4 Joint Flow Rate Control and Job Scheduling

This section provides the details of our energy-efficient thermal management technique for 3D systems with liquid cooling. The goals of our technique are: (1) Tuning the liquid flow rate to meet the heat removal demand of the current workload and reducing the energy consumption; (2) Minimizing the thermal imbalances across the chip to reduce the adverse effects of variations on reliability, performance, and cooling efficiency. To achieve these goals, we combine joint flow rate control with job scheduling. Figure 7 provides a flow chart of our method. We monitor the temperature at regular intervals for all the cores in the 3D system. Based on the forecasted change in maximum temperature, the controller is responsible for adjusting the coolant flow rate. The scheduler performs temperature-aware load balancing to also reduce the thermal gradients.

4.1 Temperature Monitoring and Forecasting

Monitoring temperature provides our technique with the ability to adapt the controller and job scheduler decisions. We assume each core has a thermal sensor. One way to utilize the thermal feedback is to react to temperature changes. A typical impeller pump like the one we use ([41]) takes around 250-300ms to complete the transition to a new flow rate. Due to the time delay in adjusting the flow rate, a reactive policy is likely to result in over-/under-cooling—the thermal time constant on a 3D system like ours is typically less than 100ms. Thus, for the liquid flow rate controller, we forecast temperature into the near future, and adjust the flow rate control on time to meet the heat removal requirement.

We use autoregressive moving average (ARMA) [42] to predict the *maximum temperature* for the next interval. Predicting maximum temperature is sufficient to select the suitable liquid flow rate to apply, as the flow rate is fixed among the channels. Note that our job scheduler balances the temperature, therefore the temperature difference among cores is minimized. ARMA forecasts the future value of the time-series signal based on the recent history (i.e., maximum temperature history in this work), therefore we do not require an offline analysis. An ARMA model is described by Equation 8. In the equation, y_t is the value of the series at time t (i.e., predicted temperature value), a_i is the lag-i auto-regressive coefficient, c_i is the moving average coefficient and e_t is called the noise, error or the residual. p and q represent the orders of the auto-regressive (AR) and the moving average (MA) parts of the model, respectively. ARMA prediction is highly accurate for temperature forecasting, and runtime adaptation methods can also be integrated with ARMA as discussed in [42].

$$y_t + \sum_{i=1}^{p}(a_i\ y_{t-i}) = e_t + \sum_{i=1}^{q}(c_i\ e_{t-i}) \tag{8}$$

The prediction is highly accurate because of the serial correlation within most workloads and the slow change in temperature due to the thermal time constants. Furthermore, the rate of change of maximum temperature is typically even slower, resulting in easier prediction. In our experiments, we use a sampling rate of 100ms, and predict 500ms into the future.

If the trend of the maximum temperature signal changes and the predictor cannot forecast accurately, we reconstruct the ARMA predictor, and use the existing model until the new one is ready. Such cases occur when the workload dramatically changes (e.g., day-time and night-time workload patterns for a server). To achieve fast and easy detection, we apply the sequential probability ratio test (SPRT) [43]. SPRT is a logarithmic likelihood test to decide whether the error between the predicted series and measured series is diverging from zero [43, 42]—i.e., if the predictor is no longer fitting the workload, the difference function of the two time series would increase. As the maximum temperature profile changes slowly, we need to update the ARMA predictor very infrequently.

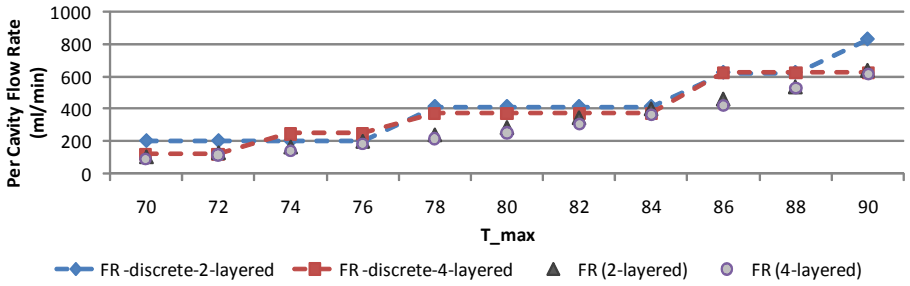

Fig. 8. Flow rate requirements to cool a given T_{max}.

4.2 Liquid Flow Rate Control

The input to the controller is the predicted maximum temperature, and the output is the flow rate for the next interval. Then, considering that we have discrete flow rate settings for the pump, we first analyze the effect of each flow rate for both 3D systems (2- and 4-layered).

Figure 8 shows which flow rate (per cavity) should be applied when the maximum temperature is T_{max} so that the temperature is guaranteed to cool below the target operating temperature of 80^oC. In this figure, the dashed lines show the discrete flow rate settings, while the triangular and circular shaped data points refer to minimum rate to maintain the desired temperatures.

Based on this analysis, we see that for a given system and maximum temperature, we already know which flow rate setting is able to cool the system to the target temperature level. We set-up a look-up table indexed by temperature values, and each line holds a flow rate value. At runtime, depending on the maximum temperature prediction, we select the appropriate flow rate from the table. As the maximum temperature prediction is highly accurate (well below 1^oC), this way we can adjust the cooling system to meet the changes in the heat removal demand on time. To avoid rapid oscillations, once we switch to a higher flow rate setting, we do not decrease the flow rate until the predicted T_{max} is at least 2^oC lower than the boundary temperature between two flow rate settings. The runtime overhead of using a look-up table based controller is negligible, considering that the cost is only limited to a look-up from a small-sized table.

4.3 Job Scheduling

Our job scheduler is a temperature-aware version of load balancing. Dynamic load balancing is a common policy used in multicore schedulers today. While frequent load balancing eliminates contention and long thread waiting times in the queues, it does not consider the location of the cores. However, a core's thermal behavior is strongly correlated with where it is placed on the chip, and the power consumption of the neighboring units.

We assume short threads, which is a common scenario in server workloads running on multiprocessor systems [18, 20]. For instance, in real-life workloads running on the UltraSPARC T1, the thread length (i.e., continuous execution time without any interrupt) has been reported to vary between a few to several hundred milliseconds [20]. Thus, since we consider threads with short lengths and similar execution time, we use number of threads for computing the job queue length of each core. Note that, depending on the available information, our approach can be extended for other workload metrics such as instruction count per thread.

To address the thermal asymmetries of cores in a 3D system, we run *Weighted Load Balancing* [5]. Weighted load balancing does not change the priority and performance aware features of the load balancing algorithm, but only modifies how the queue lengths are computed. Each core has a queue to hold the incoming threads, and the weighted queue length of a core is computed as:

$$l^i{}_{weighted} = l^i{}_{queue} \cdot w^i{}_{thermal}(T(k)) \tag{9}$$

In the equation, $l^i{}_{queue}$ is the number of threads currently waiting in the queue of core i, and $w^i{}_{thermal}(T(k))$ is the thermal weight factor. This weight factor is a function of the current maximum temperature of the system. For a given set of temperature ranges, the weight factors for all the cores are computed in a pre-processing step and stored in the look-up table. For example, consider a 4-core system, where the average power values for the cores to achieve a balanced 75^oC are p_1, p_2, p_3, and p_4, and $p_1 = p_4 > p_2 = p_3$. This means cores 2 and 3 should run fewer number of threads per unit time to maintain a balanced temperature. Thus, we take the multiplicative inverse of the power values, normalize them, and use them as weight factors to balance temperature.

5 Experimental Results

The 3D multicore systems we use in our experiments are based on the 90nm UltraSPARC T1 (i.e., Niagara-1) processor [44]. The power consumption, area, and the floorplan of UltraSPARC T1 are available in [44]. UltraSPARC T1 has 8 multi-threaded cores, and a shared L2-cache for every two cores. Our simulations are carried out with 2-, and 4-layered stack architectures. We place cores and L2 caches of the UltraSPARC T1 on separate layers (see Figure 4). Separating core and memory layers is a preferred design scenario for shortening interconnections between the cores and their caches and achieving higher performance.

First, we gather workload characteristics of real applications on an UltraSPARC T1. We sample the utilization percentage for each hardware thread at every second using `mpstat`, and record half an hour long traces for each benchmark. Also, the length of user and kernel threads were recorded using `DTrace` [45]. We use various real-life benchmarks including web server, database management, and multimedia processing. The web server workload is generated by SLAMD [46] with 20 and 40 threads per client to achieve medium and high

utilization, respectively. For database applications, we experiment with MySQL using `sysbench` for a table with 1 million rows and 100 threads. We also run the `gcc` compiler and the `gzip` compression/decompression benchmarks as samples of SPEC-like benchmarks. Finally, we run several instances of the `mplayer` (integer) benchmark with 640x272 video files as typical examples of multimedia processing. A detailed summary of the benchmarks workloads is shown in Table 2. The utilization ratios are averaged over all cores throughout the execution. We also record the cache misses and floating point (FP) instructions per 100K instructions using `cpustat`. The workload statistics collected on the UltraSPARC T1 are replicated for the 4-layered 16-core system.

Table 2. Workload characteristics

	Benchmark	Avg Util (%)	L2 I-Miss	L2 D-Miss	FP instr
1	Web-med	53.12	12.9	167.7	31.2
2	Web-high	92.87	67.6	288.7	31.2
3	Database	17.75	6.5	102.3	5.9
4	Web & DB	75.12	21.5	115.3	24.1
5	gcc	15.25	31.7	96.2	18.1
6	gzip	9	2	57	0.2
7	MPlayer	6.5	9.6	136	1
8	MPlayer&Web	26.62	9.1	66.8	29.9

The peak power consumption of SPARC is close to its average value [44]. Thus, we assume that the instantaneous dynamic power consumption is equal to the average power at each state (active, idle, sleep). The active state power is taken as 3 Watts [44]. The cache power consumption is 1.28W per each L2, as computed by CACTI [47] and verified by the values in [44]. We model the crossbar power consumption by scaling the average power value according to the number of active cores and the memory accesses. To account for the temperature effects on leakage power, we used the second-order polynomial model proposed in [48].

Many systems have power management capabilities to reduce the energy consumption. We implement Dynamic Power Management (DPM), especially to investigate the effect on thermal variations. We utilize a fixed timeout policy, which puts a core to sleep state if it has been idle longer than the timeout period (i.e., 200ms in our experiments). We set a sleep state power of 0.02 Watts, which is estimated based on sleep power of similar cores.

We use HotSpot Version 4.2 [6] as the thermal modeling tool. We use a sampling interval of 100 ms, and all simulations are initialized with steady state temperature values. The model parameters are provided in Table 3. Modeling methodology for the interlayer material to include TSVs and the microchannels has been described in Section 3. In our experiments, we compare air-cooled

Table 3. Thermal Model and Floorplan Parameters

Parameter	**Value**
Die Thickness (one stack)	$0.15mm$
Area per Core	$10mm^2$
Area per L2 Cache	$19mm^2$
Total Area of Each Layer	$115mm^2$
Convection Capacitance	140 J/K
Convection Resistance	0.1 K/W
Interlayer Material Thickness	0.02 mm
Interlayer Material Thickness (with channels)	0.4 mm
Interlayer Material Resistivity (without TSVs)	0.25 mK/W

and liquid-cooled 3D systems. For the conventional system, we use the default characteristics of a modern CPU package in HotSpot.

We assume that each core has a temperature sensor, which is able to provide temperature readings at regular intervals (e.g., 100ms). Modern OSes have a multi-queue structure, where each CPU core is associated with a dispatch queue, and the job scheduler allocates the jobs to the cores according to the current policy. In our simulator, we implement a similar infrastructure, where the queues maintain the threads allocated to cores and execute them.

We compare our technique to other well-known policies in terms of temperature, energy, and performance. **Dynamic Load Balancing (LB)** balances the workload by moving threads from a core's queue to another if the difference in queue lengths is over a threshold. LB does not have any thermal management features. **Reactive Migration** initially performs load balancing, but upon reaching a threshold temperature, which is set to 85^oC in this work, it moves the currently running thread from the hot core to a cool core. Our novel temperature-aware weighted load balancing method is denoted as **TALB**. We also compare liquid cooling systems with air cooling systems (denoted with *(Air)*). In the plots *Var* refers to variable flow rate and *Max* refers to with using a maximum (worst-case) flow rate.

Figure 9 shows the average percentage of time spent above the threshold across all the workloads, percentage of time spent above threshold for the hottest workload, and energy for the 2-layered 3D system. We demonstrate both the pump energy and the total chip energy in the plot. Note that, for the air-cooled system, there is also an additional energy cost due to the fans, which is beyond the focus of this work and not included in the plot. The energy consumption values are normalized with respect to the load balancing policy on a system with air cooling. We see that temperature-aware load balancing combined with liquid flow control achieves 10% energy savings on average in comparison to setting the worst-case flow rate. For low utilization workloads, such as `gzip` and `MPlayer`, the total energy savings (including both chip and pump energy) reach 12%, and the reduction in cooling energy exceeds 30%.

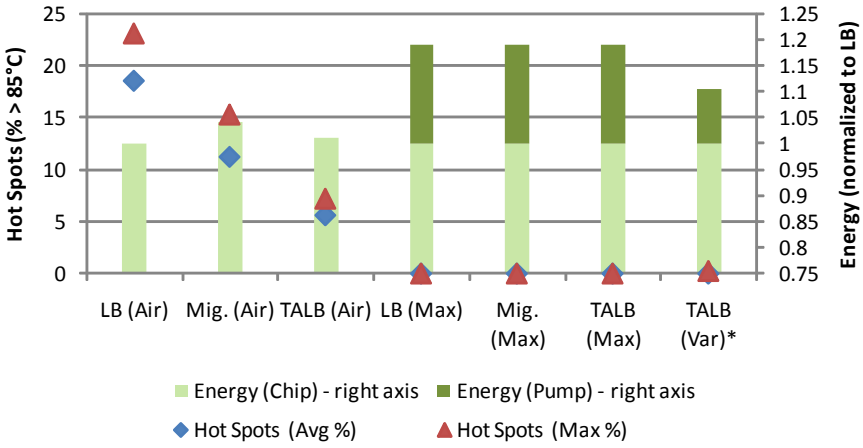

Fig. 9. Hot spots (left-axis) and energy (right-axis) for all the policies. (*) denotes our novel policy.

Figure 10 shows the average and maximum frequency of spatial and temporal variations in temperature, respectively, for all the policies. We evaluate the spatial gradients by computing the maximum difference in temperature among all the units at every sampling interval. Similarly, for thermal cycles, we keep a sliding history window for each core, and compute the cycles with magnitude larger than 20^oC. In the experiments in Figure 10, we run DPM in addition to the thermal management policy. Our weighed load balancing technique (TALB) is able to minimize both temporal and spatial thermal variations much more effectively than other policies.

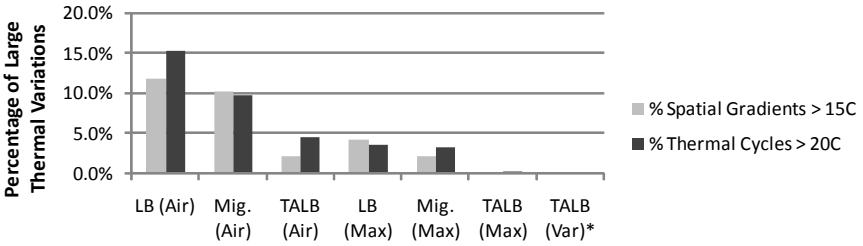

Fig. 10. Thermal variations (with DPM). (*) denotes our novel policy.

Figure 11 compares the policies in terms of energy and performance, both for the air and liquid cooling systems. For the multicore 3D systems, we compute throughput as the performance metric. We define throughput as the number of threads completed per given time. As we run the same workloads in all experi-

ments, when a policy delays execution of threads, the resulting throughput drops. Most policies we have run in this work have a similar throughput in comparison to default load balancing. Thread migration, however, reduces the throughput especially for high-utilization workloads because of the performance overhead of frequent temperature-triggered migrations. The overhead of migration disappears for the liquid cooled system, as the coolant flowing at the maximum rate is able to prevent all the hot spots, and therefore no temperature-triggered migrations occur. The figure show that for 3D systems with liquid cooling, our technique is able to improve the energy savings without degrading performance.

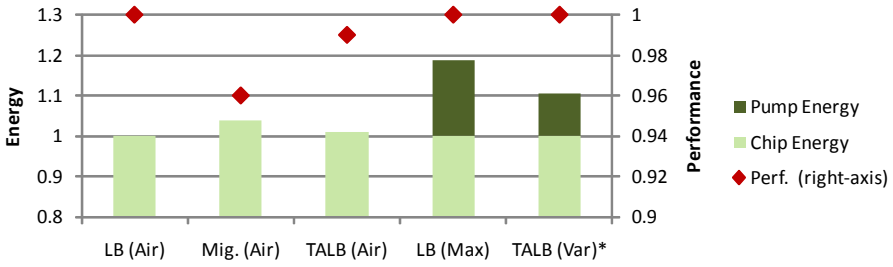

Fig. 11. Performance and Energy. (*) denotes our novel policy.

6 Conclusion

Liquid cooling is a promising solution to overcome the elevated thermal problems of 3D chips, but intelligent control of the coolant flow rate is needed to achieve energy-efficiency. In this chapter we have presented a novel controller that is able to select the minimum the coolant injection rate to guarantee a bounded maximum temperature in 3D MPSoCs under variable workload conditions. Our method minimizes the energy consumption of the liquid cooling subsystem. The controller is integrated with a novel job scheduler which balances the temperature across the system to prevent the thermal variations and to improve cooling efficiency. Our experimental results show that the joint flow rate control and job scheduling technique maintains the temperature below the desired levels, while reducing cooling energy by up to 30% and achieving overall energy savings up to 12%.

7 Acknowledgements

The authors would like to thank Thomas Brunschwiler and Bruno Michel at IBM Research GmbH, Zurich, Switzerland for their valuable contributions to the research that forms the basis of this chapter.

This research has been partially funded by the Nano-Tera.ch NTF Project CMOSAIC (ref. 123618), which is financed by the Swiss Confederation and scientifically evaluated by SNSF. This research has also been partially funded by Sun Microsystems, UC MICRO, Center for Networked Systems at UCSD, MARCO/DARPA GSRC, and NSF Greenlight.

References

[1] Brunschwiler, T., et al.: Interlayer cooling potential in vertically integrated packages. Microsyst. Technol. (2008)
[2] Gruener, W.: IBM Cools 3D Chips With Integrated Water Channels. http://www.tomshardware.com/news/IBm-research,5604.html
[3] Coskun, A.K., Rosing, T.S., Ayala, J., Atienza, D., Leblebici, Y.: Dynamic thermal management in 3D multicore architectures. In: Design Automation and Test in Europe (DATE). (2009)
[4] Coskun, A.K., Ayala, J., Atienza, D., Rosing, T.S.: Modeling and dynamic management of 3D multicore systems with liquid cooling. In: IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC). (2009)
[5] Coskun, A.K., Atienza, D., Rosing, T.S., Brunschwiler, T., Michel, B.: Energy-efficient variable-flow liquid cooling in 3D stacked architectures. In: Design Automation and Test in Europe (DATE). (2009)
[6] Skadron, K., Stan, M., Huang, W., Velusamy, S., Sankaranarayanan, K., Tarjan, D.: Temperature-aware microarchitecture. In: International Symposium on Computer Architecture (ISCA). (2003)
[7] Li, P., Pileggi, L., Asheghi, M., Chandra, R.: Ic thermal simulation and modeling via efficient multigrid-based approaches. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **25**(9) (Sept. 2006) 1763–1776
[8] Wang, T.Y., Chen, C.: Thermal-adi - a linear-time chip-level dynamic thermal-simulation algorithm based on alternating-direction-implicit (adi) method. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on **11**(4) (Aug. 2003) 691–700
[9] Yang, Y., Gu, Z., Zhu, C., Dick, R.P., Shang, L.: Isac: Integrated space-and-time-adaptive chip-package thermal analysis. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **26**(1) (Jan. 2007) 86–99
[10] Atienza, D., Valle, P.D., Paci, G., Poletti, F., Benini, L., Micheli, G.D., Mendias, J.M.: A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip. In: Design Automation Conference (DAC). (2006)
[11] Lee, K.J., Skadron, K., Huang, W.: Analytical model for sensor placement on microprocessors. In: Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on. (Oct. 2005) 24–27
[12] Mukherjee, R., Memik, S.O.: Systematic temperature sensor allocation and placement for microprocessors. In: DAC '06: Proceedings of the 43rd annual Design Automation Conference, New York, NY, USA, ACM (2006) 542–547
[13] Hamann, H.F., Weger, A., Lacey, J.A., Hu, Z., Bose, P., Cohen, E., Wakil, J.: Hotspot-limited microprocessors: Direct temperature and power distribution measurements. Solid-State Circuits, IEEE Journal of **42**(1) (Jan. 2007) 56–65
[14] Mesa-Martinez, F.J., Nayfach-Battilana, J., Renau, J.: Power model validation through thermal measurements. SIGARCH Comput. Archit. News **35**(2) (2007) 302–311

[15] Brooks, D., Martonosi, M.: Dynamic thermal management for high-performance microprocessors. In: International Symposium on High-Performance Computer Architecture (HPCA). (2001) 171–182
[16] Heo, S., Barr, K., Asanovic, K.: Reducing power density through activity migration. In: International Symposium on Low Power Electronics and Design (ISLPED). (2003) 217–222
[17] Kumar, A., Shang, L., Peh, L.S., Jha, N.K.: HybDTM: a coordinated hardware-software approach for dynamic thermal management. In: DAC. (2006) 548–553
[18] Donald, J., Martonosi, M.: Techniques for multicore thermal management: Classification and new exploration. In: International Symposium on Computer Architecture (ISCA). (2006)
[19] Chaparro, P., Gonzalez, J., Magklis, G., Cai, Q., Gonzalez, A.: Understanding the thermal implications of multi-core architectures. IEEE Transactions on Parallel and Distributed Systems **18** (2007) 1055–1065
[20] Coskun, A.K., Rosing, T.S., Whisnant, K.A., Gross, K.C.: Static and dynamic temperature-aware scheduling for multiprocessor socs. IEEE Transactions on VLSI **16**(9) (Sept. 2008) 1127–1140
[21] Li, Y., Lee, B., Brooks, D., Hu, Z., Skadron, K.: Cmp design space exploration subject to physical constraints. In: High-Performance Computer Architecture, 2006. The Twelfth International Symposium on. (Feb. 2006) 17–28
[22] Monchiero, M., Canal, R., González, A.: Design space exploration for multicore architectures: a power/performance/thermal view. In: ICS '06: Proceedings of the 20th annual international conference on Supercomputing, New York, NY, USA, ACM (2006) 177–186
[23] Huang, W., Stan, M.R., Sankaranarayanan, K., Ribando, R.J., Skadron, K.: Many-core design from a thermal perspective. In: Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE. (June 2008) 746–749
[24] Topol, A.W., La Tulipe, Jr., D.C., Shi, L., Frank, D.J., Bernstein, K., Steen, S.E., Kumar, A., Singco, G.U., Young, A.M., Guarini, K.W., Ieong, M.: Three-dimensional integrated circuits. IBM J. Res. Dev. **50**(4/5) (2006) 491–506
[25] Tezzaron: 3D IC industry summary. http://www.tezzaron.com/technology/-3D_IC_Summary.html
[26] Samsung. http://www.samsung.com
[27] Reif, R., Fan, A., Chen, K.N., Das, S.: Fabrication technologies for three-dimensional integrated circuits. In: Quality Electronic Design, 2002. Proceedings. International Symposium on. (2002) 33–37
[28] Tsai, Y.F., Xie, Y., Vijaykrishnan, N., Irwin, M.J.: Three-dimensional cache design exploration using 3dcacti. In: ICCD '05: Proceedings of the 2005 International Conference on Computer Design, Washington, DC, USA, IEEE Computer Society (2005) 519–524
[29] Loh, G.H.: 3d-stacked memory architectures for multi-core processors. In: ISCA '08: Proceedings of the 35th International Symposium on Computer Architecture, Washington, DC, USA, IEEE Computer Society (2008) 453–464
[30] Puttaswamy, K., Loh, G.: Implementing caches in a 3d technology for high performance processors. In: Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on. (Oct. 2005) 525–532
[31] Puttaswamy, K., Loh, G.H.: Thermal analysis of a 3D die-stacked high-performance microprocessor. In: Proceedings of GLSVLSI. (2006)

[32] Xue, L., Liu, C., Tiwari, S.: Multi-layers with buried structures (mlbs): an approach to three-dimensional integration. In: SOI Conference, 2001 IEEE International. (2001) 117–118
[33] Healy, M., et al: Multiobjective microarchitectural floorplanning for 2-d and 3-d ICs. IEEE Transactions on CAD **26**(1) (Jan 2007)
[34] Li, Z., et al.: Integrating dynamic thermal via planning with 3D floorplanning algorithm. In: International Symposium on Physical Design (ISPD). (2006) 178–185
[35] Zhu, C., Gu, Z., Shang, L., Dick, R.P., Joseph, R.: Three-dimensional chip-multiprocessor run-time thermal management. IEEE Transactions on CAD **27**(8) (August 2008) 1479–1492
[36] Tuckerman, D.B., Pease, R.F.W.: High-performance heat sinking for VLSI. IEEE Electron Device Letters **5** (1981) 126–129
[37] Brunschwiler, T., et al.: Direct liquid-jet impingement cooling with micron-sized nozzle array and distributed return architecture. In: ITHERM. (2006)
[38] Bhunia, A., Boutros, K., Che, C.L.: High heat flux cooling solutions for thermal management of high power density gallium nitride HEMT. In: Inter Society Conference on Thermal Phenomena. (2004)
[39] Lee, H., et al.: Package embedded heat exchanger for stacked multi-chip module. In: Transducers, Solid-State Sensors, Actuators and Microsystems. (2003)
[40] Jang, H.B., Yoon, I., Kim, C.H., Shin, S., Chung, S.W.: The impact of liquid cooling on 3d multi-core processors. In: IEEE International Conference on Computer Design (ICCD). (2009)
[41] Laing: 12 volt DC pumps datasheets. http://www.lainginc.com/pdf/DDC3-_LTI_USletter_BR23.pdf
[42] Coskun, A.K., Rosing, T., Gross, K.: Proactive temperature balancing for low-cost thermal management in mpsocs. In: International Conference on Computer-Aided Design (ICCAD). (2008) 250–257
[43] Gross, K.C., Humenik, K.E.: Sequential probability ratio test for nuclear plant component surveillance. Nuclear Technology **93**(2) (Feb 1991) 131–137
[44] Leon, A., et al.: A power-efficient high-throughput 32-thread SPARC processor. International Solid-State Circuits Conference (ISSCC) (2006)
[45] McDougall, R., Mauro, J., Gregg, B.: Solaris Performance and Tools. Sun Microsystems Press (2006)
[46] SLAMD: Distributed Load Engine. www.slamd.com
[47] Tarjan, D., Thoziyoor, S., Jouppi, N.P.: CACTI 4.0. Technical Report HPL-2006-86, HP Laboratories Palo Alto (2006)
[48] Su, H., et al.: Full-chip leakage estimation considering power supply and temperature variations. In: International Symposium on Low Power Electronics and Design (ISLPED). (2003)