# Fuzzy Control for Enforcing Energy Efficiency in High-Performance 3D Systems

Mohamed M. Sabry[‡], Ayse K. Coskun[†], David Atienza[‡]
[‡]Embedded Systems Laboratory (ESL), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
[†]Electrical and Computer Engineering Department, Boston University, USA.

***Abstract*— 3D stacked circuits reduce communication delay in multicore system-on-chips (SoCs) and enable heterogeneous integration of cores, memories, sensors, and RF devices. However, vertical integration of layers exacerbates the reliability and thermal problems, and cooling is a limiting factor in multi-tier systems. Liquid cooling is a highly efficient solution to overcome the accelerated thermal problems in 3D architectures; however, liquid cooling brings new challenges in modeling and runtime management. This paper proposes a novel controller for improving energy efficiency and reliability in 3D systems through liquid cooling management and dynamic voltage frequency scaling (DVFS). The proposed fuzzy controller adjusts the liquid flow rate at runtime to match the cooling demand for preventing energy wastage of over-cooling and for maintaining a stable thermal profile. The DVFS decisions provide chip-level energy savings and help balancing the temperature across the system. Experimental results on 8- and 16-core multicore SoCs show that the controller prevents the system to exceed the given threshold temperature while reducing cooling energy by up to 50% and system-level energy by up to 21% in comparison to using a static worst-case flow rate setting.**

## I. Introduction

3D integration is a recently proposed design method for overcoming the limitations regarding the delay, bandwidth, and power consumption of the interconnects in multicore chips, while reducing the chip footprint and improving the fabrication yield. However, one of the main challenges for designing 3D circuits is their elevated temperatures resulting from higher thermal resistivity [12], [17]. Thus, it is more difficult to remove the heat from 3D ICs. 3D systems are also prone to large thermal variations; e.g., cores located at different tiers or at different coordinates across a tier have significantly different heating/cooling rates [7]. Such large thermal gradients have adverse effects on system reliability, performance, and cooling costs.

A number of thermal management techniques have been proposed for controlling temperature on 2D multicore systems. DVFS and thread migration/scheduling based on thermal feedback are examples of such techniques (e.g., [9]). Recent research has extended 2D management techniques for workload scheduling and DVFS-based thermal management in 3D multicore systems [28], [8], [27]. However, as power densities, number of cores, and number of tiers increase, extremely high temperature values appear in 3D stacks [27], resulting in severe restrictions in high-performance 3D system design.

Inter-tier 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 [5], [4]. This technology involves injecting fluid (e.g., water) through micro-channels (or pin-fin structures) between the tiers of a 3D stack using a pump to remove the heat. While liquid cooling has a large capability in terms of thermal reduction of 3D systems, it is necessary to use such technique in conjunction with task scheduling, load balancing, and DVFS to exploit trade-offs with other key parameters in 3D multicore systems, such as energy efficiency and performance. In addition, previous work shows that as workload dynamics change at runtime, choosing the flow rate setting dynamically to meet the cooling demand while preventing over-cooling saves significant energy [5]. Combining various control knobs in a single low-overhead optimum controller is a highly challenging task, as the control parameters differ in their time constants, performance and energy overheads, and benefits. For example, changing DVFS settings has typically an overhead on the order of several tens to hundreds of microseconds, while flow rate changes may take several hundreds of milliseconds. Also, the overhead for workload scheduling is typically low [8]; however when the system is highly utilized, job scheduling is not sufficient to control the temperature, requiring more aggressive techniques such as DVFS or applying a higher flow rate for the liquid.

In this paper, we propose a novel integrated DVFS and liquid flow rate fuzzy controller, which is able to run in conjunction with temperature-aware job scheduling methods. The controller combines the flow rate setting and DVFS decisions to simultaneously minimize the hot spots, the thermal imbalance across the system, and system-level energy. Our experimental results on 2- and 4-layered 3D systems show that our fuzzy controller prevents the system to go over the given threshold temperature while achieving up to 50% reduction in cooling energy and up to 21% reduction in system-level energy in comparison to setting the flow rate at the maximum value to handle the worst-case temperature.

The rest of the paper starts with an overview of the prior work in Section II. Section III describes the thermal model for liquid-cooled 3D systems. In Section IV, we provide details on the proposed fuzzy controller. The experimental results are presented in Section V, and finally Section VI summarizes the main conclusions of this work.

## II. Related Work

Accurate thermal modeling is critical in the design and evaluation of temperature-aware systems and policies. HotSpot [20] is an automated thermal model that calculates

transient temperature response given the physical and power consumption characteristics of the chip. To reduce simulation time for large multicore systems, a thermal emulation framework for FPGAs is proposed in [1]. Latest versions of HotSpot include 3D modeling capabilities. Recent work has extended HotSpot to model liquid-cooled systems as well [6]. Finally, 3D-ICE [22] is a new thermal modeling tool specifically designed for 3D stacks, and includes interlayer liquid cooling modeling capabilities.

Dynamic thermal management in microprocessors has been introduced by Brooks et al. [3], where the authors explore performance trade-offs among various dynamic thermal management mechanisms. Activity migration [11] and fetch toggling [20] are other examples of dynamic management techniques. Kumar et al. propose a hybrid method that combines clock gating and software thermal management [13]. The multicore thermal management method introduced by Donald et al. [9] combines distributed DVFS with process migration. For multicore systems, temperature-aware task scheduling [8] achieves desirable thermal profiles at low performance cost.

Most of the prior work in thermal management of 3D systems address design stage optimization, such as thermal-aware floorplanning (e.g. [10]) and integrating thermal via planning [15]. In [28], the authors evaluate several policies for task migration and DVFS. A recent paper proposes a temperature-aware scheduling method specifically for air-cooled 3D systems [7]. This method takes into account the thermal heterogeneity among the different layers of the system. In our recent work, we have proposed a methodology to model liquid-cooled systems, and we have shown that dynamic flow rate control is able to reduce cooling energy [5]. This work is the first to integrate DVFS and variable liquid flow rate control for 3D systems to achieve more aggressive energy savings while maintaining a reliable and balanced thermal profile.

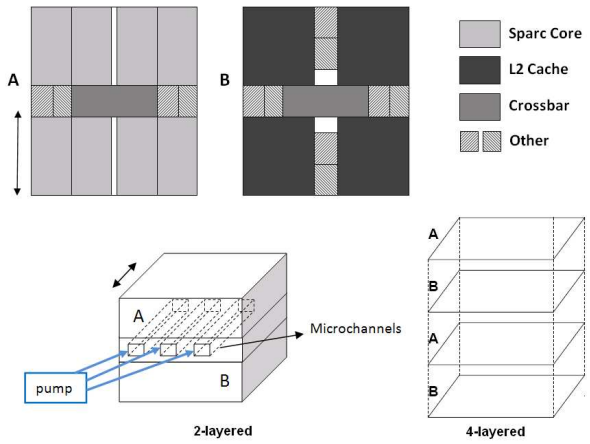


Fig. 1. Layouts of the 3D multicore systems.

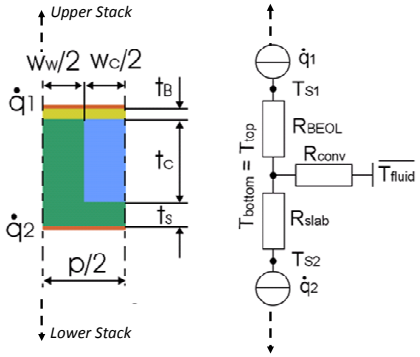


Fig. 2. Cross section of the 3D layers and the resistive network [5].

## III. 3D Stack Thermal Modeling Overview

Modeling the temperature dynamics of liquid-cooled 3D stacked architectures consist of forming the grid-level thermal R-C network, detailed modeling of the interlayer material between the tiers including the through-silicon-vias (TSVs) and the microchannels, and modeling the pump and the coolant flow rate. This section discusses the thermal modeling infrastructure for liquid-cooled 3D systems.

Fig. 1 shows the 3D systems targeted in this paper. The 3D 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. In this work, we deploy forced convective interlayer cooling with water [4]. The microchannels 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.

### A. Grid-Level Thermal Model for Liquid-Cooled 3D Systems

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 the 3D-ICE thermal model proposed in [22], which includes 3D stack modeling capabilities, with interlayer-liquid cooling.

To model the heterogeneous characteristics of the interlayer material including variable flow rate in microchannels, we introduce the ability to change the resistivity value of the cell can vary at runtime, to enable modeling the liquid coolant and dynamically changing flow rate. Thus, the interlayer material is divided into grid cells, 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. The thermal resistivity of the microchannel cells is computed based on the liquid flow rate through the cell at runtime. We set the width of grid cells to the width of a microchannel, which is $100\mu m$.

In a 3D system with liquid cooling, we compute the local junction temperature using a resistive network, as shown in Fig. 2. 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 is 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 toward the upper stack. Also, the network in Fig. 2 assumes isothermal channel walls; i.e., top and bottom of the microchannel have the same temperature.

The 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 ($\Delta T_{cond}$); (2) the coolant temperature, which increases linearly with position along the channel due to the absorption of sensible heat ($\Delta T_{heat}$); and (3) the convective ($\Delta T_{conv}$) portion, which increases until fully developed hydrodynamic and thermal boundary layers have been reached [4]. The total temperature rise on the junction, $\Delta T_j$, can therefore be 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, $\Delta T_{cond}$ is computed with Equations 2 and 3. Note that $\Delta T_{cond}$ is independent of the flow rate. Fig. 2 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, $\rho$ is the density of the coolant, and $\dot{V}$ is the volumetric flow rate in the microchannel (in l/min).

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

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

Such a temperature change is also defined by the temperature difference between the inlet and outlet temperatures of this cell. Hence, the change in temperature with respect to the fluid inlet temperature is computed using Equation 6, where $Q_n$ is the total heat flux at cell $i$.

$$\Delta T_{heat} = \sum_{i=0}^{n} \frac{\dot{Q_i}}{c_p \cdot \rho \cdot \dot{V}} \tag{6}$$

Finally, Equation 8 shows how to calculate $\Delta T_{conv}$. Note that $\Delta T_{conv}$ is independent of flow rate in case of developed boundary layers. $h$ is dependent on hydraulic diameter, Nusselt number, and conductivity of the fluid [4]. As $\Delta T_{conv}$ is not affected by the change in flow rate, we compute this parameter prior to simulation and use a constant value during experiments. Fig. 2 demonstrates $w_c$, $t_c$, and $p$ parameters on the cross-section of the 3D system.

$$\Delta T_{conv} = (\dot{q_1} + \dot{q_2}) \cdot h_{eff} \tag{7}$$

$$h_{eff} = h \frac{2 \cdot (w_c + t_c)}{p} \tag{8}$$

Table I lists all the parameters used in the computations, and provides the values we assume for the constants. The constants are taken from prior liquid cooling work [4]. 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), and this flow is further divided into the microchannels.

TABLE I. PARAMETERS FOR COMPUTING EQUATION 1

| Parameter | Definition | Value |
|---|---|---|
| $R_{th-BEOL}$ | Thermal resistance of wiring levels | Eqn.(3) |
| $t_B$ | See Fig. 2 | $300\mu m$ |
| $k_{BEOL}$ | Conductivity of wiring levels | $130W/(m \cdot K)$ |
| $R_{th-heat}$ | Effective thermal resistance | Eqn.(5) |
| $A_{heater}$ | Heater area | Area of grid cell |
| $c_p$ | Coolant heat capacity | $4183J/(kg \cdot K)$ |
| $\rho$ | Coolant density | $998kg/m^3$ |
| $\dot{V}$ | Volumetric flow rate per cavity | 0.01-0.0323 l/min |
| $h$ | Heat transfer coefficient | $371323W/(m^2 \cdot K)$ |
| $w_c$ | See Fig. 2 | $50\mu m$ |
| $t_c$ | See Fig. 2 | $100\mu m$ |
| $t_s$ | See Fig. 2 | $150\mu m$ |
| $p$ | See Fig. 2 | $150\mu m$ |

Considering the dimensions and pitch requirements of microchannels and TSVs, we assume there are 66 microchannels in between each two layers (in each cavity), and there are cooling layers on top of each tier. Thus, there are 66 and 198 microchannels in the 2- and 4-layered systems, respectively. In this work, we assign a uniform TSV density for the interlayer material. Based on the TSV density, we compute the joint resistivity of the interlayer material combining the resistivity values of the glue material and Cu. We assume 128 TSVs representing a 128-bit bus connecting the layers. We do not alter the capacitance of the interlayer material, as the TSV insertion to the heat capacity of the interface material is very small [6].

### *B. Modeling the Pump and Liquid Flow Rate*

All the microchannels are connected to a pump to receive the coolant. In a high-performance computing cluster, several stacks are typically included in multiple racks. In a such a set-up, using a different pump for each stack is impractical and costly. Therefore, we assume that a central pump, such as a centrifugal pump *EMB MHIE* [18], is responsible for the fluid injection to the whole system. This pump has the capability of producing large discharge rates at small pressure heads. Liquid is injected to the stacks from this pump via a pumping network. To enable using different flow rates for each stack, the cooling infrastructure includes valves in the network. We assume normally closed (NCV) valves provided by Festo group [19]. NCVs use external power to reduce the pressure drop and to increase the flow rate. Fig. 3 shows the pump and valve power consumption for three flow rate settings in a system with 60 stacks. All the pump and flow rate values provided are per each stack; e.g., the total pump power is 60X of the displayed power value.

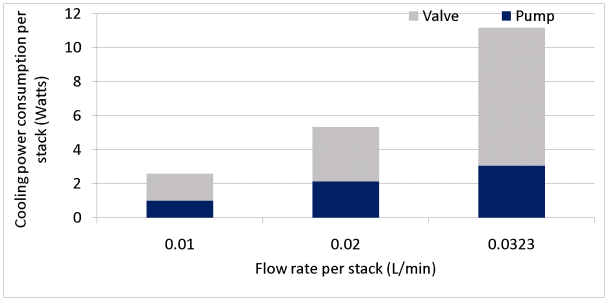


Fig. 3. Power consumption and flow rates of the cooling infrastructure.

## IV. Fuzzy Controller for Joint DVFS and Flow Rate Management

Fuzzy controllers take a control action through the use of linguistic variables. An input value to a fuzzy controller has a level of uncertainty within a specific range. This structure and flexibility make fuzzy controllers very adaptive to dynamic conditions, as they can effectively adjust a portion of or all the control knobs. This section provides the details of our *fuzzy controller*, which combines DVFS and dynamic flow rate management to achieve thermal balancing and energy efficiency.

### A. Controller Architecture

Our fuzzy controller is a Takagi-Sugeno fuzzy model [25], which defines the output(s) as a function of the inputs for a proper stability analysis. In this fuzzy model, the rules are given in the form:

$$IF\ x_1\ is\ A_{i1}\ AND\ x_2\ is\ A_{i2}\ ...\ AND\ x_k\ is\ A_{ik}$$
$$THEN\ Y = f(x_1, x_2, ...., x_k),\ i = 1, 2, ..., n;$$

where $x_k$ parameters are the inputs, $A_{ik}$ values refer to the pre-defined ranges, and Y is the output function. In our controller, the fuzzified inputs are: current temperature of the core, the recent utilization for the core, and the relative distance of the processing element from the nearest liquid inlet port. A schematic diagram of the building blocks of the fuzzy controller is shown in Fig. 4.

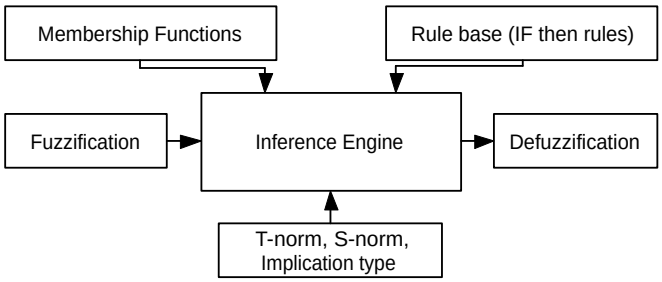


Fig. 4. Schematic diagram of the fuzzy-based thermal controller.

### B. Fuzzifier

The fuzzifier is the interface between the fuzzy controller and the outside world, as it transforms any numerical input to its corresponding fuzzy value. For any variable ($x = x_o,\ x_o \in R$) where $R$ is the range of $x$, a fuzzy function $\mu_o(x)$ is generated to be used with the rules-base in order to infer the proper output. In our controller we deploy a singleton fuzzifier [16], which is one of the most frequently used fuzzifiers in fuzzy control systems and very appropriate in our context due to its low implementation complexity. A fuzzy function has the form:

$$\mu_o(x) = 1\ when\ x = x_o,\ 0\ otherwise.$$

### C. Fuzzy Membership Functions

Membership functions are used to transfer a variable numerical range $R$ to a linguistic one, such that $\{\forall x \in R, \mu(x) \in [0, 1]\}$. This transformation is essential in fuzzy systems since the membership functions are used in the rule base. Moreover, one of the most important aspects in such a selection is the full coverage of the input variable range $R$ with $N$ fuzzy functions [16], such that $\{\forall\ x \in\ R,\ \bigcup_{i=1}^{N} \mu_i(x)\ > 0\}$. In our fuzzy-based thermal controller, each variable has full range coverage through three membership functions. We use triangular and trapezoidal-based memberships in the controller to minimize its execution complexity, as it is illustrated in Fig. 5. In particular, this figure shows the range coverage of thread utilization rate by the three functions used: low (L), medium (M), and high (H).

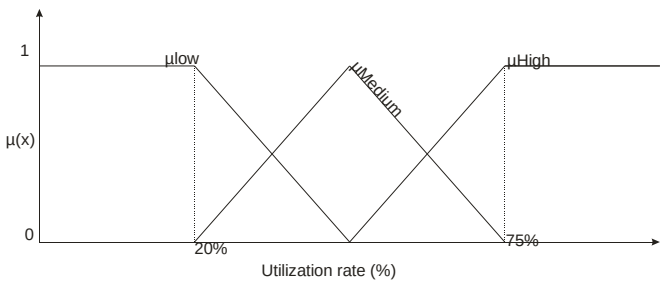


Fig. 5. An example of membership functions for thread utilization.

### D. Rule-Base Derivation

Knowledge acquisition is an important block in the design of any fuzzy controller, since it is used to derive the most appropriate rule base for the fuzzy inference engine. This acquisition can be achieved by utilizing expert knowledge or by other techniques such as genetic algorithms [24]. In our derivation, we rely on offline thermal response analysis to observe how each processing element is affected by each control variable.

Our controller uses three input variables to deduce the appropriate control action: the relative distance from the inlet port (D), the temperature (current or forecast) of the processing element (T), and the core utilization percentage (U). Then, the generated outputs are: the flow rate (FL) and the DVFS settings (VF). Each of these variables has three membership functions covering their full range, similar to Fig. 5 (the function symbols are: low (L), medium (M), and high (H)).

TABLE II. FUZZY DERIVED RULE BASE. X IS A "DON'T CARE".

| IF | | | THEN | |
|---|---|---|---|---|
| D is | AND T is | AND U is | VF is | AND FL is |
| L | X | X | H | L |
| M | L | X | H | L |
| M | M | L | L | L |
| M | M | M | M | M |
| M | M | H | M | M |
| M | H | L | L | L |
| M | H | M | M | M |
| M | H | H | M | H |
| H | L | X | H | L |
| H | M | L | L | L |
| H | M | M | M | L |
| H | M | H | H | M |
| H | H | L | L | M |
| H | H | M | L | H |
| H | H | H | M | H |

We analyzed 2- and 4-layered 3D systems using the 3D-ICE thermal simulator running various workloads, and we observed that the architectural blocks with closer to the inlet port (D is L) experience the lowest temperature, and they do not require DVFS changes (i.e., VF is always H) or the liquid flow rate (FL is always L). However, when the processing elements are located in a further location (D is M) to the input port, we need to monitor their current state and adapt accordingly. For instance, if the temperature of a core is low (T is L), no change is required in either VF or FL settings (VF is High, FL is Low). On the contrary, if the temperature reaches the medium range (T is M), the utilization rate plays a role in the controller decision. If the utilization is low (U is L), the VF and FL should be reduced to the minimum setting to minimize energy and thermal variations (VF is L AND FL is Low).

Increasing the flow rate could reduce the temperature of any core at any state, but using such method implies a large energy consumption which is not the optimal control action for an energy efficient controller. Increasing the flow rate should be applied if it is the only solution to reduce the hot spots, without jeopardizing the performance. In addition to that, the appropriate VF setting should be selected at every state to enable fine-grained thermal control along with minimal performance degradation.

In summary, by observing the similarities in a large set of different workload and temperature scenarios, we have derived the complete Takagi-Sugeno fuzzy rules base shown in Table II. In our case, the rules functions are constant values where such values are expressed as follows: $H$ corresponds to the maximum value applicable to a certain variable, $M$ is the mean value, and $L$ is the minimum value of the range of this variable.

## V. EXPERIMENTAL RESULTS

The 3D multicore systems we use in our experiments are based UltraSPARC T1 (i.e., Niagara-1) processor manufactured at 90nm node. The power consumption, area, and the floorplan of UltraSPARC T1 are available in [14]. 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 stacks, where the 4-layered system is the duplicated version of the 2-layered stack and therefore has 16 cores. We place cores and L2 caches of the UltraSPARC T1 on separate layers (see Fig. 1). Separating logic and memory layers is a preferred design scenario for shortening interconnections between the cores and their caches and achieving higher performance.

We use workload traces collected from real applications running on an UltraSPARC T1. We record the utilization percentage for each hardware thread at every second using `mpstat` for several minutes for each benchmark. We use various real-life benchmarks including web server, database management, and multimedia processing. The web server workload is generated by SLAMD [21] 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. 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 III. The utilization ratios are averaged over all cores throughout the execution. The workload statistics collected on the UltraSPARC T1 are replicated for the 4-layered 16-core system.

TABLE III. 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 | MPlayer | 6.5 | 9.6 | 136 | 1 |
| 6 | MPlayer&Web | 26.62 | 9.1 | 66.8 | 29.9 |

The peak power consumption of SPARC is close to its average value [14]. Thus, we assume that the instantaneous dynamic power consumption is equal to the average power at each state (active, idle). The active state power is taken as 3.3 Watts [14]. The cache power consumption is 1.92W per each L2, as computed by CACTI [26] and verified by the values in [14]. We model the crossbar power consumption by scaling the average power value according to the number of active cores and the memory accesses.

We compute the leakage power of processing cores as a function of their area and the temperature. We assume a base leakage power density of $0.25W/mm^2$ at 383K for 90nm as in [2]. To account for the temperature effects on leakage power, we use the model provided in [23], [11]. The leakage power at a temperature $T^oK$ is given by: $P(T) = P_o \cdot e^{\beta \cdot (T-383)}$, where $P_o$ is the leakage power at 383K, and $\beta$ is a technology-dependent coefficient. We set $\beta = 0.017$ [11].

To account for DVFS, three voltage and frequency settings are used in the simulations {V (in Volts), f (in MHz)}= $\{(1.2, 1200), (1.1, 1000), (1.0, 800)\}$. When a change in the assigned frequency occurs, the thread utilization $U$ on a core is modified to $U'$ using a worst-case performance hit assumption: $U' = U \frac{f_{old}}{f_{new}}$.

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,

TABLE IV. THERMAL MODELING PARAMETERS.

| Parameter | Value |
|---|---|
| Silicon conductivity | $130W/(m \cdot K)$ |
| Silicon capacitance | $1635660J/(m^3 \cdot K)$ |
| Wiring layer conductivity | $130W/(m \cdot K)$ |
| Wiring layer capacitance | $2174502J/(m^3 \cdot K)$ |
| Water conductivity | $0.6W/(m \cdot K)$ |
| Water capacitance | $4183J/(kg \cdot K)$ |
| Heat sink conductivity | $10W/K$ |
| Heat sink capacitance | $140J/K$ |
| Heat sink thickness | $0.7mm$ |
| Heat sink side length | $60mm$ |
| Die Thickness (one tier) | $0.15mm$ |
| Area per Core | $10mm^2$ |
| Area per L2 Cache | $19mm^2$ |
| Total Area of Each Layer | $115mm^2$ |
| Interlayer material thickness | $100\mu m$ |
| Interlayer material conductivity | $130.8W/(m \cdot K)$ |

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.

In 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 IV. This table contains the thermal conductance and capacitance values of the various materials used in modeling the stack. In our experiments, we compare air-cooled and liquid-cooled 2- and 4-layered 3D systems.

We implement various thermal management techniques to evaluate the thermal and energy efficiency of the proposed fuzzy-thermal management technique. 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. Temperature-triggered task migration (TTMig) moves tasks from a core if that core exceeds the threshold temperature ($85^oC$ in our case). TTMig has an impact on performance resulting from the time overhead required to move tasks between the cores (e.g., context switch overhead and cold start effects). In this work we assume a $1ms$ overhead when a thread is migrated to a new core. In temperature-triggered DVFS (TDVFS) the voltage and frequency settings of a core are reduced when the core's temperature exceed the $85^oC$ threshold value. In our implementation, as long as the temperature is above the threshold and there is a lower setting, we reduce the voltage frequency level at every DVFS interval. When the temperature falls below another threshold value ($82^oC$), we increase the voltage frequency setting by one step. TTMig and TDVFS can also be combined into a joint policy.

We experiment with both air-cooled (AC) and liquid-cooled (LC) systems for comparison purposes. In LC_LB, we apply the maximum flow rate (0.0323 l/min per cavity), while the jobs are scheduled with LB. Thermal impact of all the policies on the 2-layered system is shown in Fig. 6. This figure compares the % of time spent above the threshold temperature for the average case across all the workloads (marked as hot spots avg) and also for the benchmark with hottest temperatures, Web-high. TTMig and TDVFS help reduce the hot spots in air-cooled systems, while the integration of liquid-cooling

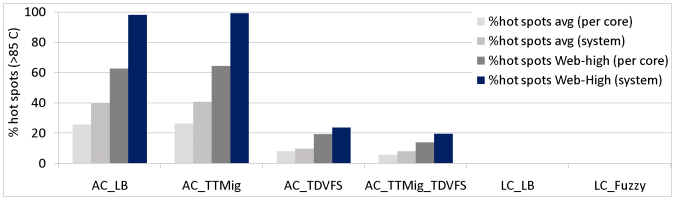


Fig. 6. Percentage of time we observe hot spots for all the policies, both for the average case across all workloads and for Web-high, the hottest benchmark. For both *avg* and *Web-high*, the figure shows the % values averaged per core and the % of time hot spots are observed across the SoC.

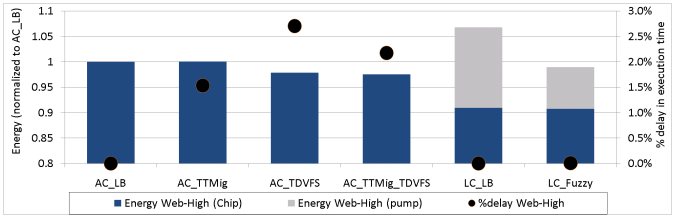


Fig. 7. Left-axis shows the energy consumption in the whole system (chip and cooling network) for Web-high, averaged per stack. The right-axis shows the % delay for each policy. Note that air cooling also includes fan power consumption, which is not included in the figure.

removes all the hot spots. The peak temperature with LB is $87^oC$, and the peak temperatures of TTMig and TDVFS are $85 - 86^oC$ in the air-cooled system. LC_LB reduces the peak temperature to $56^oC$. Fuzzy controller pushes the system into a higher peak of $68^oC$, but still avoids any hot spots. This way, it enables using a lower flow rate without creating any thermal problems.

Fig. 7 shows the total energy consumed and the performance overhead when running the various policies on the 2-tier stack for the Web-high workload. Energy consumption values are normalized with respect to the load balancing policy on a system with air cooling. The delay percentage refers to the delay in completion time of the job in comparison to running the default LB policy. The fuzzy controller achieves major reduction in both the coolant and the overall system energy consumption: LC_Fuzzy reduces system energy by 21% and cooling energy by 50% at high-utilization workloads in comparison to LC_LB. The reason LC_Fuzzy outperforms all other techniques in energy savings is due to the joint control of flow rate and VF settings at run time based on each core's thermal and utilization status. At lower utilization levels and lower temperatures, the integrated system leaks less energy than that of a higher thermal profile. On average, the proposed controller achieves 35% and 8% of coolant and overall system energy savings, respectively.

For our multicore 3D systems, we compute throughput as the performance metric. Throughput is the number of threads completed per given time. As we run the same workloads in all experiments, when a policy delays execution of threads, the resulting throughput drops. The performance degradation of Web-high under a set of policies is shown in Fig. 7. Liquid cooling-based systems do not suffer from any performance degradation since the temperature of such systems does not rise to a value where another thermal management technique

should be applied. Although our proposed fuzzy controller uses DVFS, as we apply DVFS based on the core utilization, the performance degradation results do not exceed 0.01%. This overhead negligible in comparison to the degradation observed in other scenarios (air-cooling).

In the 4-layered experiments, we observe that similar to the 2-tier stack, inter-layer liquid cooling fully eliminates the hot spots in the 4-tier stack. Simulation results show that the minimum temperature in the air-cooling stack exceed $85^oC$ at all times, leaving little opportunity for any thermal management technique to successfully control the hot spots without severely degrading the performance. On the contrary, liquid-cooling stack is able to maintain the temperature below the thermal boundaries (no hot spots exist above $85^oC$). Our proposed fuzzy controller maintains the benefit of liquid-cooling in addition to the reduction of energy consumption. In the 4-tier scenario, the proposed fuzzy controller achieves 48% and 15% average coolant and overall system energy savings, respectively, with 60% and 31% peak savings in comparison to setting the highest flow rate. The substantial increase in energy savings with respect to the 2-tiered system is due to the increased amount of cavities in the stack. With 3 existing cavities (as in Fig. 1), a flow rate of $3 \times 0.0323$ l/min is required, which in turn increases the pumping and valving power required.

## VI. Conclusion

Microchannel-based liquid cooling is a promising solution to overcome the pressing thermal challenges of high-performance 3D multicore architectures. As workload significantly changes over time and results in highly variant thermal profiles. Thus, we need intelligent control of the coolant flow rate to avoid the wasted energy consumption for over-cooling the system when the system is under-utilized. In fact, achieving high reliability, performance, and energy efficiency simultaneously in a liquid-cooled 3D stack requires joint control of a number of knobs in addition to adjusting the flow rate setting.

In this paper we have presented a novel fuzzy controller that adjusts the liquid flow rate and the DVFS settings to balance temperature across the 3D stack and to minimize system energy consumption while preventing thermal hot spots. Our experimental results with 2- and 4-tier 3D multicore case studies illustrate that our fuzzy controller maintains the temperature below the desired levels, while reducing cooling energy by up to 50% and achieving overall energy savings up to 21% with respect to setting the highest coolant flow rate to match the worst-case temperature.

## Acknowledgement

This research has been partially funded by the Nano-Tera RTD project CMOSAIC (ref.123618), financed by the Swiss Confederation and scientifically evaluated by SNSF, and the PRO3D EU FP7-ICT-248776 project.

## References

[1] D. Atienza, P. Del Valle, G. Paci, F. Poletti, L. Benini, G. De Micheli, and J. M. Mendias. A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip. In *DAC*, 2006.

[2] P. Bose. Power-efficient microarchitectural choices at the early design stage. In *Keynote Address on PACS*, 2003.

[3] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, pages 171–182, 2001.

[4] T. Brunschwiler, B. Michel, H. Rothuizen, U. Kloter, B. Wunderle, H. Oppermann, and H. Reichl. Interlayer cooling potential in vertically integrated packages. *Microsyst. Technol.*, 15(1):57 – 74, 2009.

[5] A. K. Coskun, D. Atienza, T. Simunic Rosing, T. Brunschwiler, and B. Michel. Energy-efficient variable-flow liquid cooling in 3D stacked architectures. In *Design Automation and Test in Europe (DATE)*, 2010.

[6] A. K. Coskun, J. Ayala, D. Atienza, and T. Simunic Rosing. Modeling and dynamic management of 3D multicore systems with liquid cooling. In *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC'09)*, 2009.

[7] A. K. Coskun, T. Simunic Rosing, J. Ayala, D. Atienza, and Y. Leblebici. Dynamic thermal management in 3D multicore architectures. In *Design Automation and Test in Europe (DATE)*, 2009.

[8] A. K. Coskun, Tajana Simunic Rosing, and Kenny Gross. Utilizing predictors for efficient thermal management in multiprocessor socs. *IEEE Transactions on CAD*, 28(10):1503–1516, 2009.

[9] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *ISCA*, 2006.

[10] M. Healy, M. Vittes, M. Ekpanyapong, Ch. Ballapuram, S. K. Lim, H. S. Lee, and G. H. Loh. Multiobjective microarchitectural floorplanning for 2-d and 3-d ICs. *IEEE Transactions on CAD*, 26(1), Jan 2007.

[11] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *ISLPED*, pages 217–222, 2003.

[12] W.-L. Hung, G.M. Link, Y. Xie, N. Vijaykrishnan, and M.J. Irwin. Interconnect and thermal-aware floorplanning for 3d microprocessors. In *ISQED*, pages 98–104, 2006.

[13] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha. HybDTM: a coordinated hardware-software approach for dynamic thermal management. In *DAC*, pages 548–553, 2006.

[14] A. Leon, K. W. Tam, J. L. Shin, D. Weisner, and F. Schumacher. A power-efficient high-throughput 32-thread SPARC processor. *ISSCC*, 42(1):7 – 16, 2007.

[15] Z. Li, X. Hong, Q. Zhou, Sh. Zeng, J. Bian, H. Yang, V. Pitchumani, and Ch. Cheng. Integrating dynamic thermal via planning with 3D floorplanning algorithm. In *ISPD*, pages 178–185, 2006.

[16] H. T. Nguyen and N. R. Prasad. *Fuzzy modeling and control, selected works of M. Sugeno*. CRC press, 1999.

[17] K. Puttaswamy and G. H. Loh. Thermal analysis of a 3D die-stacked high-performance microprocessor. In *ACM Great Lakes Symposium on VLSI (GLSVLSI 2006)*, pages 19–24, 2006.

[18] WILO MHIE centrifugal pump. http://www.wilo.com/cps/rde/xchg/en/layout.xsl/3707.htm.

[19] Festo electric automation technology. http://www.festo.com.

[20] K. Skadron, M.R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *ISCA*, 2003.

[21] SLAMD Distributed Load Engine. www.slamd.com.

[22] A. Sridhar, A. Vincenzi, M. Ruggiero, D. Atienza, and T. Brunschwiler. 3d-ice: Fast compact transient thermal modeling for 3D-ICs with inter-tier liquid cooling. In *International Conference on Computer-Aided Design (ICCAD'10)*, 2010.

[23] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In *ISCA*, page 276, 2004.

[24] M. Su and H. Chang. Application of neural networks incorporated with real-valued genetic algorithms in knowledge acquisition. *Fuzzy Sets and Systems*, 112(1):85 – 97, 2000.

[25] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116 – 132, 1985.

[26] D. Tarjan, Sh. Thoziyoor, and N. P. Jouppi. CACTI 4.0. Technical Report HPL-2006-86, HP Laboratories Palo Alto, 2006.

[27] X. Zhou, J. Yang, Y. Xu, Y. Du, and Y. Zhang. Thermal management for 3D processors via task scheduling. In *37th International Conference on Parallel Processing (ICPP)*, 2008.

[28] C. Zhu, Z. Gu, L. Shang, R. P. Dick, and R. Joseph. Three-dimensional chip-multiprocessor run-time thermal management. *IEEE Transactions on CAD*, 27(8):1479–1492, August 2008.