Exploring the Role of Large Centralised Caches in Thermal Efficient Chip Design

SHOUNAK CHAKRABORTY, NTNU, Norway
HEMANGEE K. KAPOOR, IIT Guwahati, India

In the era of short channel length, Dynamic Thermal Management (DTM) has become a challenging task for the architects and designers while engineering modern Chip Multi-Processors (CMPs). Ever increasing demand of processing power along with the developed integration technology produces CMPs with high power density, which in turn increases effective chip temperature. This increased temperature leads to increase in the reliability issues for the chip-circuitry with significant increment in leakage power consumption. Recent DTM techniques apply DVFS or Task Migration to reduce temperature at the cores, the hottest on-chip components, but often ignore the on-chip hot caches. To commensurate the high data demand of these cores, most of the modern CMPs are equipped with large multi-level on-chip caches, out of which on-chip Last Level Caches (LLCs) occupy the largest on-chip area. These LLCs are accounted for their significantly high leakage power consumption which can also potentially generate on-chip hotspots at the LLCs similar to the cores. As power consumption constructs the backbone of heat dissipation, hence, this work dynamically shrinks cache size while maintaining performance constraint to reduce LLC leakage, primarily. These turned off cache portions further work as on-chip thermal buffers for reducing average and peak temperature of the CMP without affecting the computation. Simulation results claim that, at a minimal penalty on the performance, proposed cache based thermal management having 8MB centralised multi-banked shared LLC gives around 5°C reduction in peak and average chip temperature, which are comparable with a Greedy DVFS policy.

CCS Concepts: • **Computer systems organization** → **Multicore architectures**; • **Hardware** → **Temperature optimization**; **Power estimation and optimization**.

Additional Key Words and Phrases: Cache Memory, Last Level Cache(LLC), Temperature, Thermal Buffer, Chip Multi-Processors(CMPs), Hotspot, Leakage Power, Dynamic Power, IPC, Reconfiguration Time

ACM Reference Format:
Shounak Chakraborty and Hemangee K. Kapoor. 2019. Exploring the Role of Large Centralised Caches in Thermal Efficient Chip Design. *ACM Trans. Des. Autom. Electron. Syst.* 00, 0, Article 00 (May 2019), 29 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Gradual decrements in Channel Length of modern transistors over the last decade with ever increasing demand of processing power motivate the chip designers to integrate more

Authors' addresses: Shounak Chakraborty, Department of Computer Science, NTNU, Trondheim, NO-7491, Norway, shounak.chakraborty@ntnu.no; Hemangee K. Kapoor, Department of Computer Science and Engineering, IIT Guwahati, Guwahati, Assam, 781039, India, hemangee@iitg.ac.in.

1084-4309/2019/5-ART00 $15.00
https://doi.org/10.1145/1122445.1122456

CPU cores in a single chip, called as Chip Multi-Processors (CMPs). To commensurate the high data demand of this set of cores, larger multi-level on-chip caches are integrated in the same chip, out of which, Last Level Caches (LLCs) occupy the maximum area on the wafer real estate. In addition, with due increment in circuit complexity, this heavy integration increases on-chip power density with enough spatial variations. This elevated power density with spatial variations introduces severe local hotspot problems raising the reliability issues of the on-chip circuitry, higher cooling costs and significant degradation in performance. Finding out the most optimal thermal management solution while maintaining performance is still a challenging task for the researchers and architects [19].

Most of the classical Dynamic Thermal Management (DTM) methods evaluate cores as the hottest on-chip elements and handle the thermal issues in recent CMPs by considering runtime system performance [19]. Usually, the recent DTM techniques find out the hottest CPU cores on-the-fly and reduce their temperature either by (I) DVFS (Dynamic Voltage and Frequency Scaling), or by (II) migration of task to colder cores [9, 19]. DVFS dynamically scales down a core's voltage and frequency settings (V/F settings) to reduce the heat dissipation of the on-chip circuitry and achieves significant savings in the energy consumption. However, slowing down a processor core through DVFS degrades computational performance. In recent past, a plethora of DVFS based works attempted to balance the performance and energy consumption optimally [5, 15, 18, 27]. Task migration, on the other hand, gathers thermal sensors' outputs before migrating tasks from a hotter zone to a relatively colder region. Depending upon the task's vitae, migration overheads are taken care while reducing ample amount of chip temperature [12, 36].

Micro-processors	Cache Power
ARM 920T	44%
Strong ARM SA-110	27%
21164 DEC Alpha	$25 - 30\%$
Niagra	12%
Niagra2	21%
Alpha 21364	13%
Xeon (Tulsa)	13%

Table 1. Power consumed by on-chip caches with respect to total power consumed by the CMP [44].

On-chip LLCs are often neglected in most of the DTM techniques as they are comparatively colder on-chip elements [19]; but, larger LLCs can be used to enhance thermal efficiency of the chip without much aggravation in performance penalty. A recent survey [16] mentioned 30 Kelvin of spatial variations in on-chip cache temperature, which strongly claims the prominent existence of cache hotspots. Moreover, a significant portion of total power consumption of modern chips comes from the high leakage power consumed by the LLCs (as given in Table 1), which in turn increases chip temperature. Later in Figure 2 and 3, we report the static and dynamic power values obtained by running benchmarks on our experimental setup. As observed, leakage component is very significant compared to dynamic and hence the temperature of area occupied by cache varies at places. Moreover, temperature increment is significantly observed in the vicinity of the cores (ref. Figure 1). However, the LLC leakage consumption can be reduced either by putting some cache portions in turned off mode (called as *Decay Cache*), or by putting the cache portion in a low power drowsy state [29, 44]. As turned off cache portions create on-chip thermal buffers, hence, in this regard, *Decay Cache*

will be the more promising option to reduce chip temperature. In particular, we attempt to reduce temperature by targeting turned-off cache banks closer to the cores.

This paper analyses the role of a centralised multi-banked SNUCA LLC (as shown in Figure 1) in thermal management while maintaining performance. We dynamically resize LLC to optimally balance performance and chip temperature by offering two levels of thermal management-(i) controlling cache temperature, and (ii) reducing temperature of the global hotspots by governing on-chip conductive heat transfer. The turned off cache banks are eventually cooled down, and create on-chip thermal buffers that reduce temperature of the adjacent on-chip components (other banks and/or cores). The major contributions of this paper can be summarised as follows:

(1) Considering performance as a system-wide constraint, we have developed an analytical model for our architecture to determine the optimal cache size.
(2) The analytically determined optimal cache size is used for resizing LLC by following the three thermal efficient patterns-
   - **AltRow.** Shuts down alternate rows of cache banks.
   - **Chess.** Generates a chessboard or checkerboard like pattern in LLC, with two colours. One of the colour will represent prospective shutdown candidates.
   - **OptTar.** Cache banks closer to cores are assigned highest shutdown priority. Additionally, future requests of the turned off cache banks are optimally handled, as more cache portions are turned off than earlier methods.

In case of caches, heavily used blocks can generate cache hotspots, whereas least used cache portions unnecessarily increase the leakage consumption contributing to the chip temperature. Furthermore, in case of some modern applications, cache access patterns do not conform to the classical cache access property, (the *Locality of Reference*), in the long run. These existing diversities in cache access behaviour across the applications shows necessity for dynamic cache resizing.

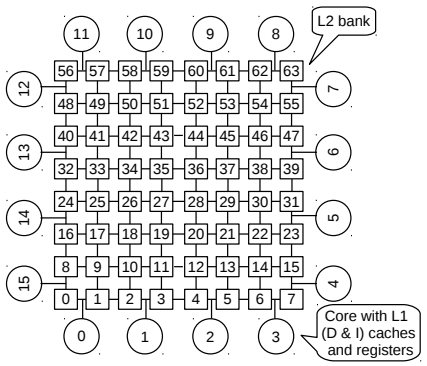


Fig. 1. Baseline CMP architecture with multi-banked centralised NUCA L2 as LLC.

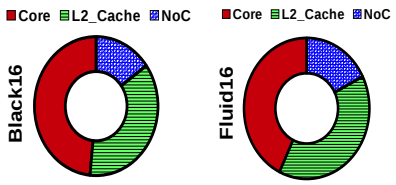


Fig. 2. Percentage contributions of the major on-chip components to the total on-chip power consumption. All values include both Dynamic & Static power (for Black16 & Fluid16).

The rest of the paper is organised as follows. Motivation for our work is given in Section 2. We discuss the preliminaries along with analytical formulation of our proposed work in Section 3. Section 4 states the proposed algorithm with brief elaboration on the issues towards implementation of dynamic cache resizing. Simulation setup and Analysis of Results are discussed in Section 5. Before concluding the article in Section 7, available literature are described in Section 6.

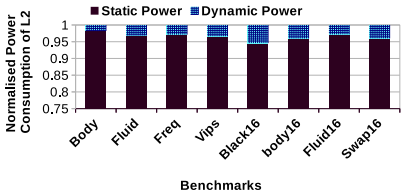


Fig. 3. L2 Power Values: Dynamic and Static.

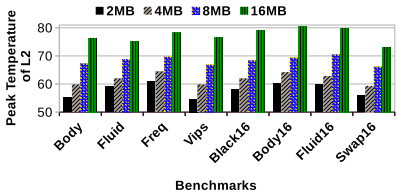


Fig. 4. Size vs LLC's Peak Temperature (in °C).

2 MOTIVATION

In this section, we discuss the role of LLCs towards the chip temperature. The observations shown here motivate us to target the LLCs and its reconfiguration to control the chip temperature.

2.1 Baseline Architecture

The baseline architecture, used in this work (shown in Figure 1), contains 16 homogeneous CPU cores, which are placed along the periphery of the chip. Each circle in the figure, numbered from 0 to 15, represents a single core along with its private Data and Instruction L1 caches. The centrally located shared L2 cache [1] (on-chip LLC in our case) is sliced set-wise into identical 64 banks, numbered from 0 to 63, called as centralised shared cache. This set-wise division implies that, all ways of a set are present in the same bank. Note that, we are using Static NUCA (SNUCA) architecture, where a block is always placed at a fixed cache set on its every allocation. A 2D mesh Network on Chip (NoC) connects all the L2 banks and the Cores.

2.2 Leakage Hungry LLCs

For our baseline architecture (Figure 1), the total power consumption of the chip can be divided into three major components: (I) power consumed by the individual cores that includes processing power along with power consumed by L1 (data & instruction) caches; (II) power consumed by NoC; and (III) power consumed by the LLC (L2 in our case). Figure 2 depicts power consumption of these major three on-chip components. The power values include both dynamic as well as static components. However, L2 contributes a significant portion to the total power consumed by the CMP, for both the applications- Black16 (compute intensive) and Fluid16 (memory intensive). In case of Fluid16, L2 consumes more (dynamic) power than Black16, but a small increment is noticed, because majority of L2 power is coming from its static (leakage) component [1] (as shown in Figure 3). It can now be concluded that, leakage power of L2 (as LLC in our case) has become a significant contributor to the total on-chip power consumption. Note that, these power values are derived by running a set of PARSEC benchmark applications [4]. Furthermore, independent to the cache accesses, leakage power that has a circular dependency on temperature can be reduced by gating the cache banks. So, reducing LLC leakage power by dynamically resizing it, can be a promising option to reduce chip temperature without affecting the computational units.

[1]Breaking up of *Core Power* by exposing *L1* and *Computational* power consumption individually can show significant changes across various workloads [37].

2.3 Thermal Potential of the LLCs

Figure 4 shows the changes in peak temperature of LLCs for 4 different sizes. The values are derived (for a set of PARSEC applications) from our simulation setup discussed later. Peak temperature of LLC gradually increases with its size and even reaches around 80°C for a number of cases. Also observe that, for each of these benchmarks temperature increases with the cache size indicating the effect of leakage. As larger caches accumulate more transistors, these in turn increase both leakage as well temperature. These values strongly indicate the existence of cache hotspots in larger LLCs having size 8MB or more. This phenomenon motivates us to explore the on-chip thermal efficiency while dynamically resizing larger LLCs. Also note that, in case of memory intensive applications, cache temperature is higher than the others, as seen from Figure 4.

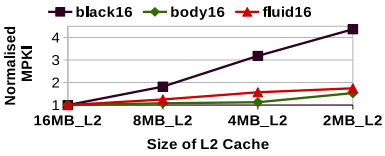


Fig. 5. Change in (Normalised) MPKI with respect to Size of LLC.

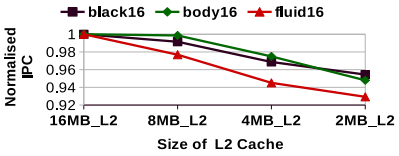
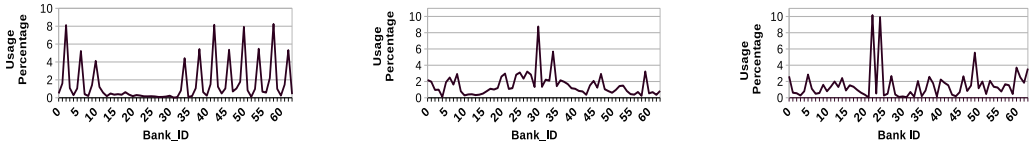


Fig. 6. Change in (Normalised) IPC with respect to Size of LLC.

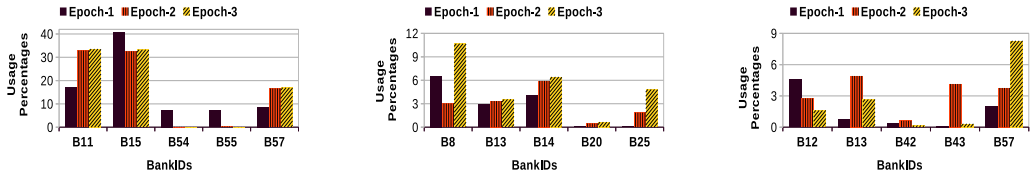
2.4 LLC size vs. Performance

Dynamic cache resizing, on the other hand, can severely retard system performance if the application's Working Set Size (WSS) does not fit in the resized cache. We performed a set of experiments with the same three applications mentioned above for different cache sizes. The capacity and conflict misses increase with the reduction in cache size, which reduces system IPC (Instructions Per Cycle) by incurring more memory stall cycles. Figure 5 shows the change in MPKI (Misses Per Kilo Instructions) while changing cache size from 16MB to 2MB and the corresponding system-wide IPCs are shown in Figure 6. IPC degradation is 7% (for fluid16) while reducing the cache size to 2MB, and this degradation is 5% for 4MB. However, if the cache is dynamically resized we can have the possibility of using the size as per application's requirement. In particular, required cache sizes can be used at different time slices by keeping an appropriate performance degradation constraint.



(a) black16 (b) body16 (c) fluid16

Fig. 7. Non uniform distribution of cache bank accesses.



(a) black16 (b) body16 (c) fluid16

Fig. 8. Change in cache bank access behaviour over time.

Runtime Cache Behaviour. The run-time cache accesses across the banks are unevenly distributed in our multi-banked baseline architecture (Figure 1). Figure 7 shows the distribution of cache accesses for all 64 banks. The results are shown for three long run applications-black16, body16 and fluid16, after running them upto consecutive 100 millions of instructions in our simulation setup. Although cache accesses exploit *Locality of Reference*, but, in the case of long running applications, this property violates with respect to bank mappings. The change in access patterns over long time-span for Black16, Body16 and Fluid16 are shown in Figure 8. The access patterns are shown along y-axis of the cache banks over the epochs for Black16, Body16 and Fluid16, respectively. We have taken 5 banks (along the x-axis) from each of the benchmarks, to show heavily used, moderately used and least used banks in each epoch. Epochs are implying a time-span of last 10 million cycles, just after the warmup (*Epoch-1*), at the middle of execution (*Epoch-2*), and at the end (*Epoch-3*). The figures show that, for all the applications, accesses for a bank are changing for the three different epochs. A heavily used bank can become a lightly used one later or vice versa.

Finally, from above discussions, following observations can be listed:

(1) Diversity exists in cache access behaviour across the applications with significant changes during execution.
(2) *Locality of Reference* with respect to *bank id* may not be exploited over a long-run.
(3) Access behaviour for the banks are diverse in nature and also changes during execution.

From a performance perspective, these observations indicate cache bank turn-off during less utility and turn-on when in greater demand.

2.5 Core based vs Cache based Thermal Management

According to the survey [19], both DVFS and Task Migration techniques are promising options to reduce peak and average temperature but may suffer from performance degradation. Migrating tasks from a hot core to a colder one incurs idle clock cycles, whereas, slowing down by DVFS increases the execution time which can violate the overall EDP (Energy Delay Product) budget. Cache based policies, on the other hand, may increase memory stalls. But modern CMPs are equipped with the larger caches, hence, reduction in cache size may not be always performance degradation prone. Thus, cache based methods can assist in thermal management. They can be used either in isolation or in conjunction with core based methods. Note that, cache based methods are more effective in case of large LLCs.

3 PRELIMINARIES AND ANALYTICAL PROBLEM FORMULATION

3.1 SRAM: from a Power/Thermal Perspective

Energy consumption of the SRAM cells can be divided as follows:

$$E_{total} = E_{Dynamic} + E_{Static} \tag{1}$$

$E_{Dynamic}$, the Dynamic energy, is consumed during read or write accesses of the cache blocks. As writing energy is computed by the similar set of equations like read energy, hence, only the equations for read accesses are provided here to avoid redundancy. $E_{Dynamic}$ [24, 31] for a read access is computed as follows:

$$E_{dyn-read} = E_{dyn-read-req-net} + E_{dyn-read-data} + E_{dyn-rep-net} \tag{2}$$

Here, $E_{dyn-read-req-net}$ denotes the energy consumption per read request and $E_{dyn-rep-net}$ represents the energy consumption for replying a read request. Energy consumption during accessing data array, i.e. $E_{dyn-read-data}$, can be written as follows:

$$E_{dyn-read-data} = E_{dyn-predec-blks} + E_{dyn-dec-drivers} + E_{dyn-read-bitlines} + E_{dyn-senseamps} \tag{3}$$

The components $E_{dyn-predec-blks}$ and $E_{dyn-dec-drivers}$ imply dynamic energy consumption of predecoder and decoder drivers, respectively. Sense amplifiers' dynamic energy is $E_{senseamps}$ whereas dynamic energy for reading bitlines is $E_{dyn-read-bitlines}$.

E_{Static}, on the other hand, represents static/leakage energy of the SRAM cell, having direct dependency both upon the running temperature and supply voltage. Modern CMPs with 32nm or lesser technology are equipped with larger LLCs, which increases on-chip transistor counts with shorter channel length, inherently increasing the power density and in turn higher leakage energy consumption. The total leakage energy consumption E_{Static} can be written as follows:

$$E_{leak-read} = E_{leak-req-net} + E_{leak-data-array} + E_{leak-rep-net} \tag{4}$$

$E_{leak-req-net}$ and $E_{leak-rep-net}$ represent leakage energy consumptions for the request and the reply networks, respectively. Leakage for data array is denoted by $E_{leak-data-array}$ which is further divided into predecoder's leakage, decoder driver's leakage, sense-amplifier's leakage and leakage of memory cells:

$$E_{data-array} = E_{leak-predec-blks} + E_{leak-dec-drivers} + E_{leak-mem-cells} + E_{leak-senseamps} \tag{5}$$

Increment in cache temperature increases its leakage consumption which can even dominate the other energy components of the chip [24] (Figure 2). Following equation shows the direct dependency of leakage upon the running temperature and supply voltage [14]:

$$P_s(t) = K_1 V_{DD}(t) T^2(t) e^{(\alpha V_{DD}(t)+\beta)/T(t))} + K_2 e^{(\gamma V_{DD}(t)+\delta)} \tag{6}$$

$P_s(t)$ denotes the static power consumption at time t for a CMOS circuit. V_{DD} is the supply voltage and $T(t)$ implies the current temperature. $K_1, K_2, \alpha, \beta, \gamma$ and δ are empirical constants which represent different circuit parameters.

The conductive heat transfer from the hotter core part of our baseline architecture will increase temperature of the nearby LLC banks which in turn increases the cache leakage consumption. This increased leakage further increases chip temperature and forms a circular dependency between leakage power and temperature.

3.2 Core Temperature Modeling

Cores, the hottest on-chip elements, also consume dynamic and static (leakage) powers, but, unlike caches, core's dynamic power is higher than its leakage. The dynamic power consumption of a core C_i, denoted by $P_d^{Ci}(t)$, at time t is directly proportional to its frequency and supply voltage-

$$P_d^{Ci}(t) = \alpha.C_{Ci}.V_{Ci}^2(t).f_{Ci}(t) \tag{7}$$

where α, C_{Ci}, $V_{Ci}(t)$ and $f_{Ci}(t)$ denote activity factor, capacitance, supply voltage and running frequency of core Ci at time t, respectively. The static/leakage power at time t has the similar formula like Equation 6, which shows that, leakage power has direct dependency on temperature. Again, the temperature of a circuit also increases with the heavy leakage power consumption. This phenomenon of leakage is called as circular dependency of leakage on temperature. Authors in [14], have decoupled this circular dependency with the help of Piece-Wise Linear Approximation of a curve. We adopt this in our thermal model as follows:

$$P_s^{Ci}(t) = P_{s-min}^{Ci} + k_T^{Ci}.T_{Ci}(t) + k_v^{Ci}.V_{Ci}(t) \tag{8}$$

And the core-temperature is modeled as,

$$T_{Ci}(t) = (P'^{Ci}_d.V_{Ci}^2(t).f_{Ci}(t) + \zeta_v^{Ci} + P'^{Ci}_s).R \tag{9}$$

where, $P'^{Ci}_d \triangleq \zeta_T^{Ci}.P_d^{Ci}$, $P'^{Ci}_s \triangleq \zeta_T^{Ci}.P_s^{Ci}$ and R is a system's constant. ζ_T^{Ci} and ζ_v^{Ci} are temperature-leakage coefficient and voltage-leakage coefficient, respectively [14].

3.3 Problem Formulation

This paper basically focuses on reducing chip temperature by decaying/reconfiguring its on-chip LLC dynamically. Figure 9 shows the internal architectures of our target CMP [1]. The inner grey blocks numbered from 0 to 63 are indicating L2 banks. Power supply is separated to each bank. Initially, the system starts running and chip temperature increases as the execution proceeds, until it reaches at its steady state, where dissipation of generated heat is same with the heat absorption by attached cooling mechanism.

Analytically our problem can be defined as follows. Given the number of (homogeneous) cores, N, and the number of cache banks (of same size), B. Our problem is to find out the optimal number of turned on banks (b) for a given performance constraint (C) which can minimise the running average chip temperature (T_{mean}).

Find optimal value for $b(\leq B)$, the number of powered on banks, such that,

$$T_{mean} = Mean(T_1^c, T_2^c, ..., T_N^c, T_1^b, T_2^b, ..., T_B^b) \tag{10}$$

is minimised, where T_i^c is the temperature of core i and T_j^b is the temperature of L2 bank j. Now, our problem is to find out the optimal value of b for which our performance constraint $IPC_i \geq C$ is not violated. Keeping b as system wide shared parameter, we formulate another performance maximisation problem as given below. Maximise

$$IPC = \frac{1}{N}\sum_{i=1}^{N} IPC_i, \tag{11}$$

subject to,

$$IPC_i = \frac{IC_i}{CC_i + MC_i(b)} \geq C. \tag{12}$$

IPC is the system wide average IPC across the cores with b number of L2 banks to satisfy the above defined constraints. IPC at core i is represented by IPC_i, which depends

upon the total instruction counts (IC_i) of itself, the number of cycles required to perform computation (CC_i) at core i and number of memory cycles (MC_i) (a function of b), required to access L2 (includes both hit and miss cycles). Dynamic cache resizing modifies b and in turn MC_i. The modification details are given in Appendix A, where it is shown how IPC depends on b (ref. Equation 24). We will use this analysis to obtain the optimal number of cache banks needed to get the desired performance.

3.4 Performance Modeling with Cache Size

Each core's IPC (IPC_i) is directly proportional with its own instruction count (IC_i) and inversely proportional with the total clock cycles required to execute IC_i at core i. The total number of clock cycles is summation of its CPU cycles (CC_i) and memory latency ($MC_i(b)$). Keeping IC_i and CC_i as constants, we can say that, IPC is a function of $IPC_i(b)$. Equation 11 can be a concave function now, if we vary b in the $MC_i(b)$. So, performance improvement at core i can then be written as-

$$PI_i = \frac{\partial IPC_i(b)}{\partial b} \tag{13}$$

Values of PI_i varies with the applications. The estimation of PI_i at t is basically derived from its value at $t-1$. Performance degradation sets value of PI_i as negative. As L2 is a shared resource to all the cores, we can assume a uniform change in PI_i for all i.

$$PI_1 = PI_2 = PI_3 = ... = PI_N \tag{14}$$

By using Lagrange Multiplier [8], above equation can be proved as:

$$LF = \frac{1}{N}\sum_{i=1}^{N} IPC_i(b) + \sum_{i=1}^{N} \lambda_i (IPC_i - C) \tag{15}$$

The first term in the RHS of above equation implies the objective whereas second term implies the constraints to be satisfied. Equation 15 is maximised when $(b, \lambda_1, \lambda_2, ...\lambda_N)$ is a stationary point, and first order derivative is zero [8]. Note that, λ_i is the i-th Lagrange Multiplier for i-th core and LF denotes the Lagrange Function.

$$\frac{\partial LF(b, \lambda_1, \lambda_2, ..., \lambda_N)}{\partial (b, \lambda_1, \lambda_2, ..., \lambda_N)} = 0 \tag{16}$$

This has now formed $N+1$ equations with $N+1$ unknowns, and IPC maximises when Equation 14 satisfies and optimal value of b can be derived.

3.5 Thermal Model

The temperature of on-chip components are driven by the following factors: (a) the component's own power consumption, (b) heat abduction by the ambient and (c) heat exchange among the peer components. The temperature of a component $T_{com}(t)$ at time t can be modeled as [38]:

$$T_{com}(t) = T_{com}(t-1) + f_{gen}(P_{dyn}(t) + P_{st}(t)) - f_{rem}(T_b(t-1) - T_a) + \sum_{m=1}^{p_{com}} f_{tr}(T_{com}(t-1) - T_m(t-1)) \tag{17}$$

where, $T_{com}(t-1)$ is the temperature of the component *com* at time $t-1$. $f_{gen}(P_{dyn}(t)+P_{st}(t))$ denotes the generated temperature due to its power consumption, whereas $f_{rem}(T_b(t-1)-T_a)$ is the change in temperature due to heat abduction or removal by the ambient, the effective way of cooling. The last component $f_{tr}(T_{com}(t-1) - T_m(t-1))$ implies the temperature change due to heat transfer among the peers (p_{com}), which obeys the principle of *superposition and reciprocity* [38].

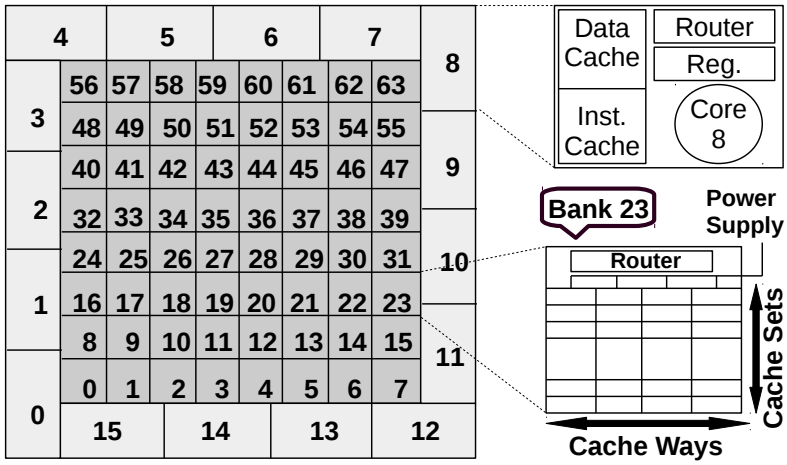


Fig. 9. Internals of our baseline CMP.

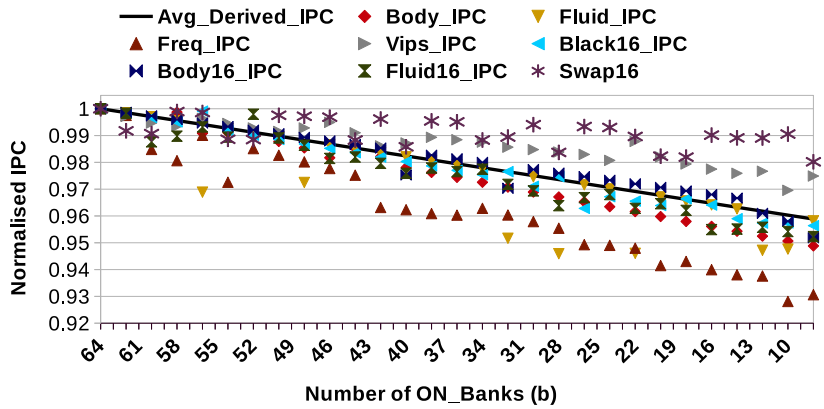


Fig. 10. Relationship between IPC and Cache Size.

Before modeling temperature for our CMP (ref. Figure 9), we divided the whole CMP into three zones - (i) the core area, for which the thermal status depends on other adjacent core blocks and the neighbouring cache banks; (ii) the cache banks adjacent to the cores, where heat exchanges between the core blocks and the peer cache banks; and (iii) other cache banks, where heat flows only among the cache banks. Therefore, the temperature of a core block (C), the bank (M) adjacent to core and a inner bank (I) at time t can be modeled respectively, as-

$$T_C(t) = T_C(t-1) + f_{gen}(P_{dyn}(t) + P_{st}(t)) - f_{rem}(T_C(t-1) - T_a) + \sum_{m=1}^{p_C+p_B} f_{tr}(T_C(t-1) - T_m(t-1)) \quad (18)$$

$$T_M(t) = T_M(t-1) + f_{gen}(P_{dyn}(t) + P_{st}(t)) - f_{rem}(T_M(t-1) - T_a) + \sum_{m=1}^{p_C+p_B} f_{tr}(T_M(t-1) - T_m(t-1)) \quad (19)$$

$$T_I(t) = T_I(t-1) + f_{gen}(P_{dyn}(t) + P_{st}(t)) - f_{rem}(T_I(t-1) - T_a) + \sum_{m=1}^{p_B} f_{tr}(T_I(t-1) - T_m(t-1)) \quad (20)$$

The notations have their usual meaning as given in equation 17. p_C denotes the number of peer core blocks and p_B is the number of adjacent cache banks for the corresponding element whose temperature is being modeled. We further modify equation 18 by replacing f_p with the power consumption parameters for a core of Equation 9 as follows:

$$T_C(t) = T_C(t-1) + ({P'_d}^C.{V_C}^2(t).f_C(t) + \zeta_v^C + {P'_s}^C).R - f_a(T_C(t-1) - T_a) + \sum_{m=1}^{p_C+p_B} f_c(T_C(t-1) - T_m(t-1)) \quad (21)$$

We use the insight gained from these zone based temperature variations to decide locations of cache banks as candidates for shutting down. In particular, we propose and analyse three different patterns (given in Section 4.2).

3.6 Combined Analytics

Shutting down of a cache bank will make $f_{gen}(P_{dyn}(t) + P_{st}(t)) = 0$ in Equations 20 and 19. When power consumption for a block is zero, its temperature will only depend upon the ambient temperature (T_a) and the temperature of its peers. Zero power consumption over a long time-span will retard the temperature increment rate and gradually the block will cool down. More temperature difference with colder (power gated) peers will increase conductive heat transfer from the hotter powered-on block to the colder peers, and eventually temperature of the powered-on block will also reduce. Therefore, output of f_{tr} in Equations 18, 19 and 20 will produce smaller values and will reduce temperatures of cores along with the cache area. Hence, it can be concluded that, the mean temperature of the chip will be reduced more by gating more number of cache banks at appropriate locations.

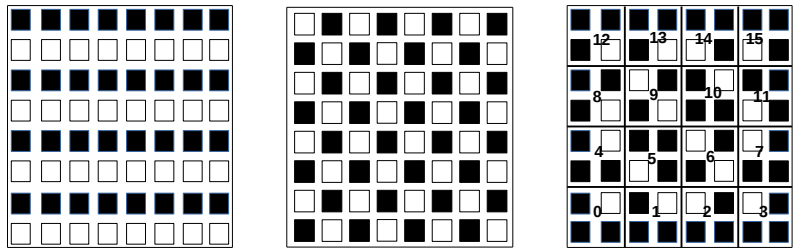
On the other hand, drastic reduction in the cache size curtails the performance by incurring more number of cache misses. According to section 3.4, performance has been maximised based on the available number of cache banks for a given performance constraint. To show the effectiveness of our theoretical model we assume a CMP having a uniform execution pattern, i.e., same number of instructions are executed by each of the cores and all cache banks are assigned with uniform workload. Now, while dynamically reducing cache size our model must satisfy the Equation 14 for maximising performance with the available cache size. Moreover, cache size reduction through power gating increases both the miss rate and the NoC latency while accessing the target banks.

3.7 Finding out Optimal b

According to the equations 12 and 13, IPC of a core will be affected if the cache capacity changes. This is reflected in the component $MC_i(b)$ in the equations that represents the memory access latency. As derived in Appendix A, this latency depends on the number of cache misses, the available number of cache banks, off chip access delay and delay incurred for redirected requests to target banks.

Benchmarks	Body	Fluid	Freq	Vips	Black16	Body16	Fluid16	Swap16
%-age fluctuation	2.14	1.12	1.92	2.46	1.58	1.09	2.36	1.92

Table 2. Maximum %-age of IPC fluctuation over the intervals for baseline architecture.



(a) AltRow (b) Chess (c) OptTar

Fig. 11. L2 bank Shutdown patterns.

Our aim is to shutdown cache banks to control temperature while keeping IPC under a given constraint. In other words, we have to put a limit on the banks that can be shutdown in order to remain under the given IPC constraint. The number of banks required by an application to maintain its performance is dependent on its WSS. As WSS varies across the applications, hence, it is difficult to arrive at an optimal number of banks to be kept powered-on. We therefore obtain the value of IPC for different number of cache banks for all benchmark programs (ref. Sec. 5). Then by applying linear regression on the obtained values we plot the trend of IPC w.r.t. the number of cache banks, as shown in Figure 10. In this Figure, for memory intensive processes like *freq*, *fluid* and *fluid16*, the slope is higher, and for compute intensive processes like *swap16*, *vips*, values are above the derived line. For mixed load like *body*, *body16* it is almost with the average line derived from the regression. But, for *black16*, a compute intensive large workload, it is also close to the line, due to higher NoC overhead after shutting down some banks. Practically, in case of *black16*, turning off more banks has increased the distances between some heavily used cores and their frequently accessed LLC blocks. Note that, for this experiment, we decided 8 fixed target banks (with static remapping) at the central location of the chip. However, using this plot, we further fixed up the value of b (#banks to be kept ON) for a given value of IPC degradation. Additionally, IPC fluctuates for a set of multi-threaded applications as the execution proceeds. But, it is evident from our experiments that, for PARSEC applications, this fluctuation is lesser than 2.5% for our set of applications in baseline architecture. Hence, reflected IPC degradation that is more than 2.5% clearly indicates that it is caused due to cache resizing. Table 2 shows the temporal fluctuations of IPC for Black16, Body16 and Fluid16. Therefore, for our experiments, we have put a limit of 4% on IPC degradation and hence using results of Figure 10 we derive the value of b as 16 for our setup. Thus we can shutdown maximum of $(B - b)$ number of L2 banks.

4 DYNAMIC CACHE RESIZING FOR THERMAL EFFICIENCY

4.1 Implementing Dynamic Cache Resizing

Dynamic cache resizing follows the classical state destroying cache decay approach, where cache blocks need to be moved to some other location before turning off [29] through power gating [34]. The cache bank which is going to be turned off is termed as *victim bank* in our paper. Blocks at victim are evacuated by either of the two ways- (a) write them back to the lower level memory (may be off chip), or (b) transfer them to some other location/bank, called as target bank. We have chosen the latter one as it retains the blocks on-chip and

avoids immediate costly off-chip accesses. These blocks are migrated to its target bank before turning off the bank and all of its future requests are redirected to the target bank. The victim stalls its incoming requests during migration and establishes remapping at victim's controller on completion of the migration. At this point, the victim is gated and all of its stalled and subsequent requests are redirected to the target bank.

Selection of Target. Target banks are selected based upon-(a) its usage (number of accesses to the bank), (b) distance from the victim, and (c) on-chip location. If the usage is very high, then selecting this bank as a target may increase its conflict and capacity misses. The distance implies number of NoC hops that has to be traversed by the future requests to reach the target. Hence, more distance increases both access delay (ref. Appendix A) and NoC power. Lastly, the targets have to be assigned a bit far from the local hotspots or power dense on-chip locations, as it will handle extra load which can make it a hotspot leading to increased heat dissipation.

Unlike some prior works [29, 44], we implement request remapping in L2 controller (by keeping L1 transparent). While placing victim's data at target bank, we use *Reuse Counter* like [7] for LRU selection in replacement. Our policy does not allow shutting down of target banks to avoid multiple or transitive redirection. However, a bank can become a target of many banks, but, each victim/turned-off bank has a unique target.

4.2 Patterns for Cache Resizing

The thermal model in Section 3.5 discussed that temperature of any on-chip component depends on 3 prime factors: (a) power consumption of the component, (b) heat abduction by the ambient and (c) heat exchange among the peers. The insight gained from here is used to decide which cache banks can be turned-off to control the temperature. (For example, if cache banks near the hot processing cores are turned off, they can assist in temperature reduction by effectively applying (a) & (c) mentioned above.) Banks along the periphery are the ones closest to the cores. As the cores are one of the hottest on-chip elements, if the nearby cache banks are turned off it can create thermal buffers helping in reducing overall temperature. In particular, we design three patterns for cache resizing that indicate which banks are to be shutdown and their corresponding target locations. Figure 11 shows three patterns, named as AltRow, Chess and OptTar, respectively.

- *AltRow* (Figure 11(a)), the first pattern, turns off cache banks located on alternate rows. The target banks for a shutdown bank is the powered-on bank in the neighbouring row, having NoC hop-distance 1. During execution, the powered-off and target banks switch roles.
- *Chess* follows a pattern like a chess board (ref. Figure 11(b)) where black coloured banks are turned off and others are kept powered-on becoming targets of their gated peers. This pattern is also altered like the earlier one during execution.
- *OptTar* shuts down banks adjacent to the cores (shown in black in Figure 11(c)) for creating more thermal buffers near the chip's hotspots. The banks are clustered into a size of 4. Each cluster along the periphery has one powered-on and three gated banks. The inner clusters can have 2-ON and 2-OFF banks as per requirements. The ON/OFF banks change roles during execution. However, banks along periphery, i.e. near the cores are kept turned off. The bank which is ON in a cluster becomes the target for the OFF banks in that cluster.

ALGORITHM 1: Performance Constrained and Thermal Efficient Dynamic Cache Resizing.

Input: i, r, t, b_{opt}, δ, M_c, M_p
Initialize $m_c = 0$, $m_b = 0$ and add $Bank_ID$s to M_c and M_b according to the selected pattern;
Run with the baseline system for initial duration i ;
repeat
  Call Reconf();
  Call Trans();
until *end of execution*;

Function Reconf()
repeat
  Call Turn-off();
  Call Turn-on();
until *end of* r;
Return;

Function Trans()
Turn on all the turned off banks;
$m_c = m_p = m = 0$;
repeat
  Run the application;
until *end of* t;
Modify the selected pattern by updating M_c & M_p, such that the roles of ON/OFF banks swap;
Return;

Function Turn-off()
if $IPC_{deg} < \delta$ & $m < b_{opt}$ **then**
  if $m_p < |M_p|$ **then**
    Select hottest bank m_{hot} from M_p, make Target m_{tar} ;
    $m_p = m_p + 1$;
  end
  else if $m_c < |M_c|$ **then**
    Select hottest bank m_{hot} from M_c, make Target m_{tar} ;
    $m_c = m_c + 1$;
  end
  Migrate blocks from m_{hot} to m_{tar};
  Turn Off m_{hot} and enable remapping to target at m_{tar} ;
  $m = m_c + m_p$;
end

Function Turn-on()
if $IPC_{deg} \geq \delta$ & $m \geq 1$ **then**
  Turn on coldest bank m_{cold} from the list of shutdown banks;
  if $m_{cold} \in M_c$ **then**
    $m_c = m_c - 1$;
  end
  if $m_{cold} \in M_p$ **then**
    $m_p = m_p - 1$;
  end
  $m = m_c + m_p$;
end

4.3 Algorithms and Discussions

The implementation of the proposed policy needs to track core-wise IPCs dynamically, with the current temperature (monitored by the on-chip thermal sensors [19] located across the chip wafer).

The practical implementation of our algorithm divides the whole execution time into several big intervals. The intervals are used either for the reconfigurations or for the transitions, as shown in Figure 12. The reconfiguration intervals allow cache resizing while maintaining

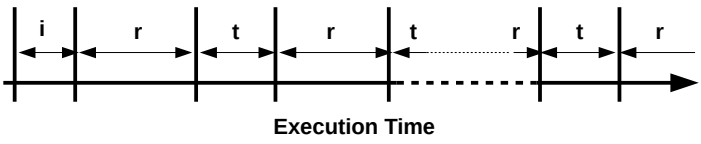


Fig. 12. The division of Execution time while implementing Algorithm 1.

Param.	Descriptions	Param.	Descriptions
i	Length of initial duration	t	Length of transition interval
r	Length of reconfig. interval	δ	Maximum percentage of IPC degradation
M_c	Set of central Bank_IDs to be gated	M_p	Set of peripheral Bank_IDs to be gated
m_c	Counts no. of turned-off banks from M_c	m_p	Counts no. of turned-off banks from M_p
b_{opt}	Maximum no. of banks can be turned-off	m	Counts total no. of turned-off banks

Table 3. Description of Parameters used in Algorithm 1. (*Param.* implies *Parameters*)

performance constraint, whereas transition intervals do not allow any cache resizing and runs the whole system by turning on all shutdown components. After running the application for some initial duration we collect bank usage-statistics for all the banks along with the thermal profile of the chip. The list of banks along the periphery and in the central part of the chip that can be shutdown are maintained by two sets, M_p and M_c, respectively. Using the temperature values, if the number of turned-off banks has not reached its limit, we select a bank from either of the lists as a candidate for shutdown. The target bank for this candidate bank is chosen as per the selected pattern (discussed in Section 4.2). The data blocks of the candidate are transferred to the target banks and on completion of this process the candidate is powered off. All subsequent future requests are forwarded to the target. The successive bank shutdown process continues until we reach at the maximum limit of turned off banks or the performance degrades beyond the predefined limit. In case the performance constraint is violated, the coldest among the turned off banks is selected for turning on. The remapped data belonging to this bank is relocated from its target before resuming its normal operations. This whole process continues for a long enough interval.

The target banks during the reconfiguration interval are overloaded with the workloads of the turned off banks, resulting in increment in their power density hence, their temperature increases. Even for some memory intensive applications, hotspots can get generated at the target banks if they are being accessed heavily for a certain time quantum. Hence, in order to reduce the power density (or hotspot) we need to switch their roles with the colder turned off banks. This procedure goes through a transition interval where all the powered off banks are turned on, and cache size remains unchanged until the end of the interval. With the onset of the next reconfiguration interval, the selected pattern is updated by switching the roles of banks and starts the cache resizing process. This process continues until the end of the application execution.

Detailed steps of the whole process are given in Algorithm 1 which divides the whole process into three parts. The parameters used in this algorithm are described in Table 3.

- *The master part* (line no. 1 to 5)- With the beginning of the execution, it initialises all the required parameters (listed in Table 3) and run the system for an initial duration of i clock cycles. While the process is running, system alternatively calls Reconf() and Trans() functions which represent two intervals r and t respectively as shown in Figure 12.
- *Function Reconf()* (line no. 8 to 13)- This function checks whether to power on (Turn-on() sub-routine) or off (Turn-off() sub-routine) the cache banks while maintaining performance constraint δ. The number of shutdown cache banks are also kept within optimal limit b_{opt}. The algorithm tries to turn off banks from the peripheral parts (line no. 26 to 28). Once this list is exhausted it then attempts to turn off the inner central banks (line no. 30 to 32). On violation of performance degradation constraint, the coldest among the turned off banks is turned on (line no. 40 to 48). The whole

process runs in each reconfiguration interval r. The cache resizing follows a particular selected pattern from the three discussed earlier.

- *Function Trans()* (line no. 15 to 22)- The gated banks are turned on at the beginning of the function. During the transitive interval t, no cache resizing is allowed. On completion of t, roles are exchanged among the candidates for ON and OFF banks of the last r interval while following the selected pattern.

Components	Parameters
No. of Banks	64
Processor	UltraSPARCIII+
Flit Size	16 bytes
Buffer Size	4
#Virtual Networks	5
L1 I/D Cache	64KB, 4-way
L2 Cache bank	128KB, 8-way
Memory bank	1GB, 4KB/page
Pipeline Stage	5-stage
VCs per Virtual Network	4

Table 4. System and Network Parameters

Name	Details	Input, Intensity
blackscholes (*Black*)	multithreaded appl (16 threads)	Medium, Comp.
Swaptions (*Swap*)	multithreaded appl (16 threads)	Medium, Comp
Vips (*Vips*)	multithreaded appl (16 threads)	Medium, Comp.
Black16	16 copies of blackscholes	Large, Comp.
Swap16	16 copies of swaptions	Medium, Comp.
Body16	16 copies of bodytrack	Large, Mix.
Fluidanimate (*Fluid*)	multithreaded appl (16 threads)	Large, Mem.
Freqmine (*Freq*)	multithreaded appl (16 threads)	Large, Mem
Fluid16	16 copies of freqmine	Large, Mem.

Table 5. PARSEC [4] details. Comp. is CPU intensive, Mem. is memory intensive, and Mix. implies Mixed loads.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

For our hardware platform, we use the floorplan as shown in Figure 9. The whole chip is divided into 80 tiles, which are of two types-(a) the 16 core tiles, located along the periphery, and (b) 64 central cache tiles. Each of the core tiles consists of an UltraSPARCIII in-order core, in 32nm technology. The core tiles are homogeneous in nature and composed by several

Cache Parameters	Values	Core Parameters	Values
Cache Level	L2	Clock rate	3000MHz
Technology used	32nm	MUL per core	1
Size of a L2 Bank	128KB	ALU per core	2
Associativity	8	FPU per core	1
Block Size	64 Bytes	Ambient temperature	47°C
Cache Model	SNUCA		

Table 6. McPAT and HotSpot Configurations

units- an Instruction Fetch Unit (IFU), a Load Store Unit (LSU), a private L1 Data and Instruction cache. The shared L2 as on-chip LLC, is divided into 64 homogeneous banks and are distributed uniformly across the 64 cache tiles (ref. Figure 9). These on-chip components are connected through a 2D mesh, for which a router is equipped to each of the cache tiles as well as with the cores. Table 4 contains configuration details of processor cores, memory and NoC which are used in our simulation. Using analysis of Section 3.7, we use the value obtained there for optimal b (i.e. number of banks to be kept ON) in our experimental evaluation. Also note that, IPC degradation constraint is kept to 4%.

The whole system is simulated in a multi-core timed-based simulator GEMS [26] runs on top of a full system simulator, SIMICS [25]. Ruby, the memory module of GEMS, handles coherency issues of CMP cache, with its own SLICC based cache controller. The cache resizing overheads of proposed policies are also implemented in this memory module. GEMS uses Garnet [2] to simulate the on-chip interconnects. Garnet further includes Orion [42] power model to simulate the NoC power consumption. We use McPAT [24] and HotSpot 6.0 [46] to simulate power and thermal profile of our proposed architecture, respectively. The trace based thermal simulator HotSpot 6.0 generates temperature traces from the power values derived by McPAT. Performance traces are collected from GEMS at each 0.1 million Ruby cycles and are fed into McPAT for power evaluation from which HotSpot derives the thermal profile of the system. Table 6 contains the configuration details used by the McPAT and HotSpot 6.0. This closed looped evaluation involving GEMS [26], McPAT [24] and Hotspot 6.0 [46] uses the multithreaded PARSEC benchmark suite [4] to validate the proposed architecture (Table 5). PARSEC is composed of a set of multi-threaded emerging real-life applications (both memory and compute intensive) that include several multi-media processing and mathematical tasks. Multiple copies of a single application are further clubbed and run together by using Solaris commands to get the multiprogrammed workloads.

5.2 Leakage Energy and EDP Savings

As power reduction constructs the backbone of any thermal management policy, hence, we first show the leakage energy reduction while implementing the three above mentioned patterns in our closed loop simulation environment. The mix of 8 applications from PARSEC [4] are executed, and their corresponding leakage energy savings are shown in Figure 13. The x-axis represents the benchmarks whereas y-axis shows the leakage energy consumptions normalised to the baseline's leakage consumption. The maximum leakage savings of 40.3% are achieved for OptTar. On an average, a static energy savings of 26% and 26.5% have been achieved for AltRow and Chess, respectively.

The effect in static energy savings has sound reflection on total energy consumption of the chip, with an effect on IPC (ref. equation 12). Figure 14 shows the savings in EDP, which is derived from total energy consumption of chip (including both cores' and cache-energy) and IPC. The average EDP savings for three patterns are 11%, 11.5% and 18.7%, respectively. The more static energy saving in the OptTar gives more EDP gains than the others.

Policy	Body	Fluid	Freq	Vips	Black16	Body16	Fluid16	Swap16	Average
AltRow	5.35	4.29	3.90	4.19	5.80	4.50	3.78	3.43	4.40
Chess	4.82	4.69	5.67	5.26	4.99	4.17	4.62	3.52	4.72
OptTar	5.96	3.54	6.04	9.62	8.23	7.26	3.19	5.60	5.82

Table 7. Increment in NoC Energy over baseline (in %-age)

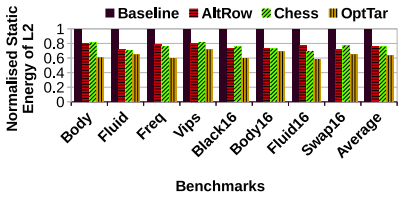


Fig. 13. Static Energy savings at L2 Cache.

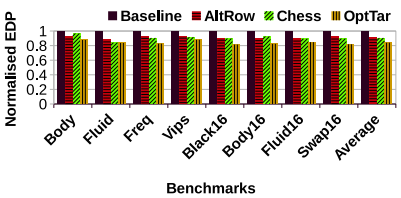


Fig. 14. EDP Savings of the chip.

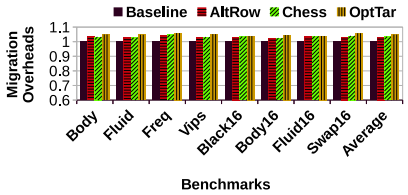


Fig. 15. Migration Overhead.

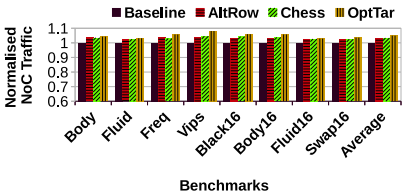
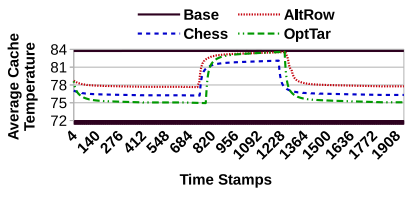


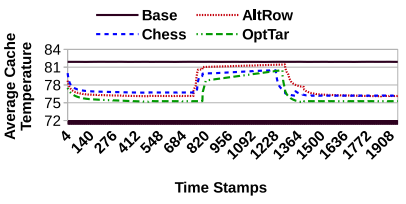
Fig. 16. Increment in NoC Traffic.

5.3 Dynamic Cache Resizing Overhead in NoC

Dynamic cache resizing increases NoC overhead due to two major operations-(a) block migration during turn-off process, and (b) remapping of requests after bank shutdown. Figure 15 shows the extra clock cycles required for migration. For most of the applications, this average increment is around 3% for AltRow and Chess, whereas for OptTar, it is close to 5%. However, this extra time-span keeps idle only the victim banks, those are going to be turned-off just after completion of migration process. Whereas, the operations at the non-victims are executed normally during this period. On the other hand, the remapped blocks to the target locations increase NoC traffic. The maximum average increment in NoC traffic (for OptTar) is 4.86%, which is shown in Figure 16. Migration and remapping altogether further increase NoC energy consumption (ref. Table 7) by 5.82%, (on an average across the applications), for OptTar than our baseline architecture. The distance cognizant target selection for all of our patterns incorporate a maximum hop distance of 2 in OptTar, and 1 for the rest. Hence, OptTar experiences maximum energy overhead than AltRow and Chess, however, this overhead has been compensated by the gained leakage and EDP savings.



(a) Fluid16



(b) Body16

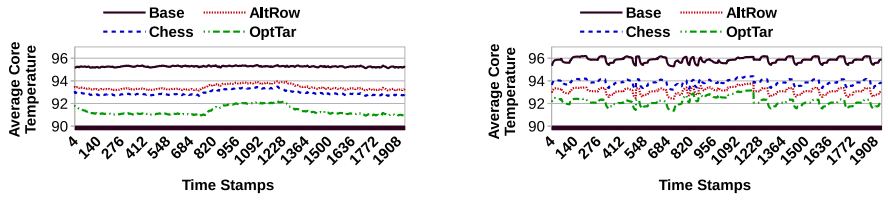
Fig. 17. Snapshot of temporal changes in **average** temperature of L2 during execution.

5.4 On-Chip Thermal Profile

The turned off banks consume zero power and eventually cooled down due to the heat abduction by the ambient (equation 17). Additionally, the on-chip components at vicinity of these shutdown banks will transfer heat towards this cold zone and temperature of the adjacent components will also decrease gradually. To show the effect of our proposal, we compare all three patterns for the best case (for Fluid16) and for the worst case (for Body16) scenarios. We show the reduction in average temperature of cache area and core blocks, separately. Figure 17 and 18 show the reduction in average temperatures for cache and core area, respectively. The x-axes in these figures represent a portion of the execution interval which includes two consecutive reconfiguration ($\boldsymbol{r}$) intervals with a transition ($\boldsymbol{t}$) in between. The temporal changes in both caches' and cores' temperatures are noticed due to bank shutdown in both **r** intervals as well as in **t**. The y-axis denotes the temperature values in °C.

For all the cases, OptTar reduces more temperature than the others due to gating of more and optimally located cache area. The reduction in average temperature for OptTar is around 8.3°C in case of Fluid16. For Body16, OptTar reduces the cache temperature by 5.7°C, during the same interval. Due to their adjacency to the turned off cache banks, average temperature for the core area is reduced by 4.3°C for Fluid16, whereas for Body16 the value is around 3.8°C. The core temperature for Body shows fluctuation due to temporal variation in processing loads across the cores. Furthermore, at the beginning of **t**, the banks are turned-on, hence, the average cache temperature escalates. With the beginning of the next **r** after **t**, again the temperature decreases for both L2 cache as well as for cores as the cache banks are gradually shutdown.

Furthermore, we also show the thermal status for individual L2 banks along with their adjacent cores at an instance during **r**. Instead of showing 64 banks with 16 cores, we have taken the bottom left part of the chip (Core id 0, 1, 14 and 15, with 4 consecutive banks from each row starting with bank id 0, 8, 16 and 24 in Figure 9). Figure 19 shows the temperatures for these 16 banks with 4 adjacent cores while running Fluid16. The grey blocks in the figure represents shutdown banks. Trivially, remarkable temperature reductions are noticed for the turned off cache area. At the cores, for all of our policies, temperature reduction are in a range of $3-4$°C, with maximum reduction of 4.1°C for OptTar. All these results conform to the analytical model set in Section 3. In particular, shutting down the banks helps in removal of heat by creating thermal buffers. At the same time, using the optimal value of b, we are able to maintain the performance constraint. The suggested three patterns for thermal balancing are also seen to be effective.



(a) Fluid16 (b) Body16

Fig. 18. Snapshot of temporal changes in **average** temperature of the cores during execution.

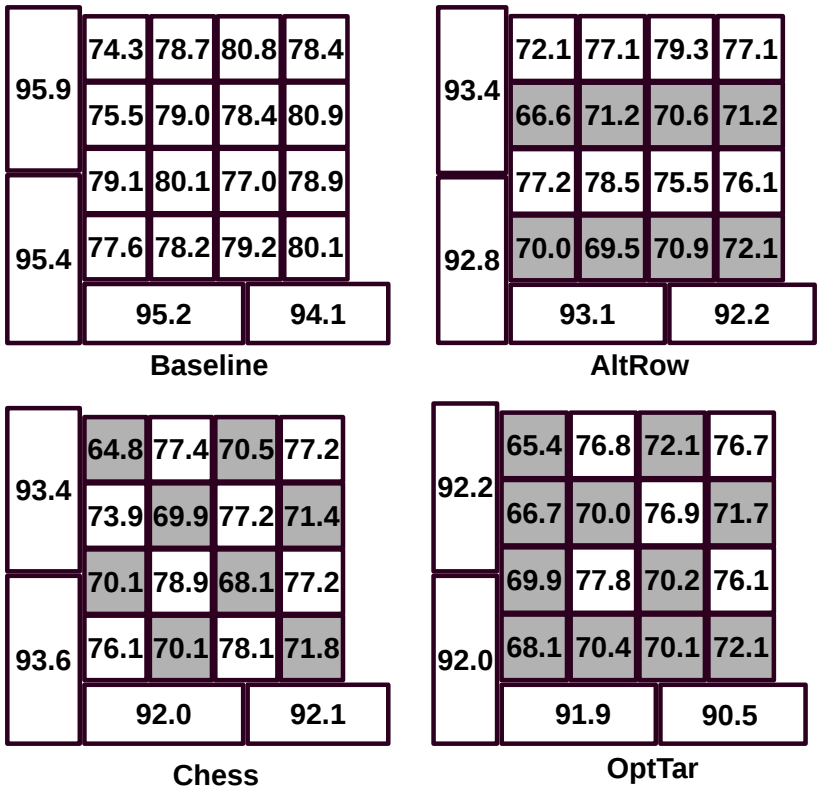
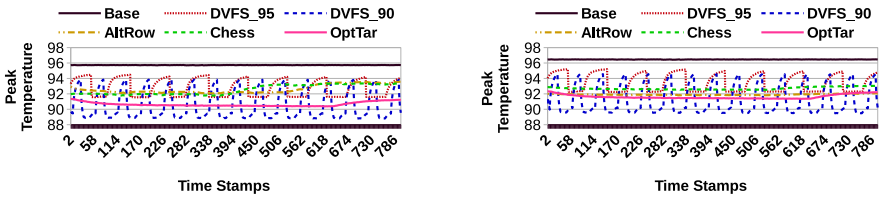


Fig. 19. Temperatures (in °C) of the individual banks during **r**, for Fluid16.

5.5 Comparison with Greedy DVFS [28]

We have implemented a per-core DVFS based thermal optimisation technique (in our simulation framework), called as Greedy DVFS [28] for comparing our cache based thermal efficient policies. Greedy DVFS uses a predefined threshold temperature value, on violation of which V/F setting is scaled down by one step dynamically. Conversely, when temperature of a core is below the threshold, V/F setting will be stepped up for better performance. The change in V/F setting is done periodically per-core at the end of a fixed interval during execution. The detailed V/F settings used in our implementation are given in Table 8. We



(a) Fluid16 (b) Body16

Fig. 20. Snapshot of temporal changes in **peak** chip temperature during a **r**.

Voltage, Frequency	1.8v, 3.0GHz	1.65v, 2.4GHz	1.5v, 1.8GHz	1.35v, 1.2GHz

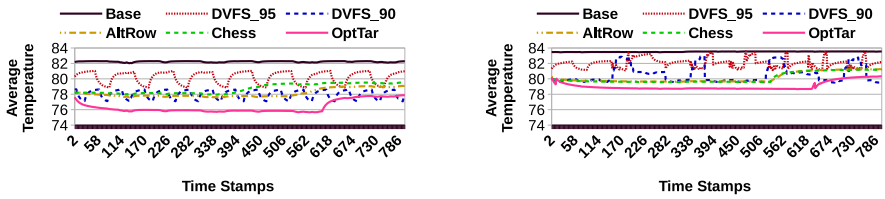
Table 8. V/F Settings for UltraSPARCIII (used in our simulation)

use two threshold values of 95°C and 90°C which are termed as DVFS_95 and DVFS_90, respectively.

Figure 20 shows the temporal changes in peak temperature of entire chip for both Greedy DVFS and our policies during the reconfiguration interval. For both Fluid16 and Body16, our policy shows lesser peak temperature reduction than DVFS_90. As DVFS directly reduces the core's energy consumption, hence, core temperature reduces more than the cache based method. However, in order to compensate the system performance, DVFS shortly scales up the V/F setting resulting into further increment in temperature within a short time-span. This frequent temperature change can affect the on-chip circuitry due to unstable thermal profile. On the other hand, cache based method ensures stability in reduced temperature range. We achieve around 5.5°C and 4.6°C reduction in peak temperature for Fluid16 and Body16, respectively. Thus, we can further claim that, the proposed cache based policy can be a supplementary approach to the core-based policies that use DVFS. Additionally, applying both core and cache based policies together will also be more effective in improving on-chip thermal efficiency.

Scaling V/F settings in DVFS generates huge temporal fluctuation in core's temperature incorporating an adverse effect in circuit reliability [6]; whereas cache based techniques offer stable thermal status of the chip. Figure 21 shows the temporal change in average chip temperature while using DVFS and our cache based methods. The frequent change in cores' temperature due to Greedy DVFS incurs fluctuation in average chip temperature for both Fluid16 and Body16. The detailed maximum reduction in peak and average chip temperature are given in Table 9 and 10, respectively. On an average, DVFS_90 and OptTar reduces peak temperature by 6.6°C and 5.5°C, respectively, but cache based policy offers more thermal stability which may help for better durability for the on-chip circuitry. Due to shutting down of larger on-chip area, cache based policy shows more decrements in average chip temperature which is around 4.9°C, whereas DVFS_90 reduces the same by 4.2°C on an average.

A stable thermal profile ensures better reliability for on-chip circuitry. Tables 11 and 12 show the standard deviations for the temporal changes in peak and average temperature of the chip for different applications. The maximum standard deviation for peak and average temperature are 0.58 and 0.64, respectively, almost for all of our cache based techniques.



(a) Fluid16 (b) Body16

Fig. 21. Snapshot of temporal changes in **average** chip temperature during a **r**.

Benchmarks	DVFS_95	DVFS_90	AltRow	Chess	OptTar
Body	3.1	6.1	3.5	3.7	4.9
Fluid	4.4	5.9	3.9	3.6	5.1
Freq	4.2	5.4	4.3	4.5	5.4
Vips	5.7	5.8	4.0	3.9	4.8
Black16	5.5	6.1	3.9	4.5	5.2
Body16	3.8	6.3	4.6	3.9	4.6
Fluid16	4.2	6.6	3.9	4.1	5.5
Swap16	3.2	5.9	3.6	3.8	4.9
Gmean	**4.2**	**6.0**	**3.9**	**4.0**	**5.1**

Table 9. Maximum Reduction in Peak Temperature (°C) of the Chip w.r.t. baseline.

Benchmarks	DVFS_95	DVFS_90	AltRow	Chess	OptTar
Body	2.2	4.0	4.2	3.9	4.8
Fluid	2.1	4.4	4.3	4.2	4.6
Freq	2.8	4.2	4.1	4.0	4.6
Vips	2.4	4.3	4.3	4.4	5.0
Black16	3.2	4.8	4.2	4.1	4.7
Body16	2.7	3.9	3.9	3.8	4.4
Fluid16	3.4	4.8	4.3	4.0	5.8
Swap16	2.0	4.6	4.0	4.2	5.4
Gmean	**2.6**	**4.2**	**4.2**	**4.1**	**4.9**

Table 10. Maximum Reduction in Average Temperature (°C) of the Chip w.r.t. baseline.

Whereas, in case of DVFS based policies, least values for standard deviation is 1.01 and 0.67 for peak and average chip temperature, respectively. These values represent better thermal stability of our cache based policies over greedy DVFS.

Benchmarks	DVFS_95	DVFS_90	AltRow	Chess	OptTar
Body	2.68	2.19	0.46	0.42	0.58
Fluid	1.32	1.01	0.05	0.09	0.21
Freq	1.78	2.13	0.08	0.19	0.34
Vips	1.31	2.67	0.14	0.16	0.42
Black16	1.10	1.95	0.17	0.24	0.28
Body16	2.71	2.51	0.11	0.21	0.32
Fluid16	2.56	2.20	0.16	0.14	0.26
Swap16	2.83	1.36	0.13	0.21	0.23
Gmean	**1.91**	**1.93**	**0.13**	**0.19**	**0.31**

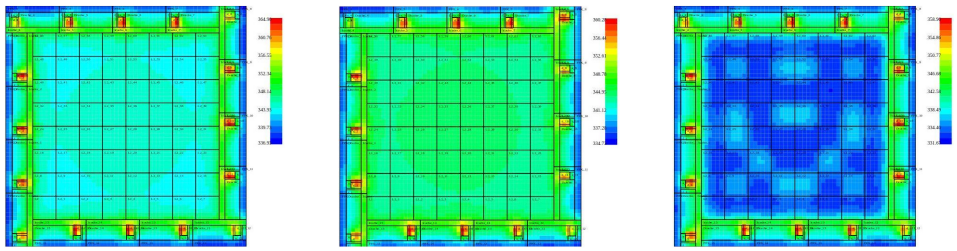
Table 11. Standard Deviation: **Peak** Temperature of the Chip.

5.5.1 Spatial Thermal Status. Diversities in power consumption across the on-chip area define the spatial thermal status of the chip. Figure 22 shows the thermal status of the chip for baseline, DVFS_90 and OptTar during some $\boldsymbol{r}$, generated from Hotspot 6.0 [46]

Benchmarks	DVFS_95	DVFS_90	AltRow	Chess	OptTar
Fluid	0.73	0.81	0.36	0.12	0.61
Body	1.21	0.79	0.51	0.61	0.90
Freq	1.41	0.72	0.23	0.41	0.63
Vips	0.67	0.71	0.55	0.44	0.64
Black16	1.20	1.02	0.51	0.43	0.61
Body16	1.21	1.32	0.71	0.52	0.60
Fluid16	1.11	0.80	0.41	0.17	0.61
Swap16	1.14	0.98	0.34	0.35	0.60
Gmean	**1.05**	**0.85**	**0.43**	**0.34**	**0.64**

Table 12. Standard Deviation: **Average** Temperature of the Chip.

tool while running Fluid16 application. The corresponding temperature ranges are given in the figures. The peak temperatures have been reduced by 4K for DVFS_90 and 6K for OptTar, respectively. In case of OptTar, cache temperature is reduced more as expected and for which average chip temperature shows better reduction than DVFS_90. Moreover, created thermal buffers reduce the peak temperature values of the core area by generating more conductive heat flow towards the cache area from the core. DVFS_90 reduces the core temperature which hardly reduces the cache temperature as reduced cores' temperatures are still a bit higher than the cache temperature. Note that, all the temperature values in Figure 22 are in Kelvin (K).



(a) Baseline (b) DVFS_90 (c) OptTar

Fig. 22. On-chip spatial thermal behaviour generated by Hotspot 6.0 [46] for Fluid16 at some certain instance during **r**. Temperature Ranges for three configurations-(a) Baseline: $336K$ to $364K$, (b) DVFS_90: $334K$ to $360K$, (c) OptTar: $332K$ to $358K$. Red colour represents peak value where Blue represents the lowest temperature.

Benchmarks	**DVFS_95**	**DVFS_90**	**AltRow**	**Chess**	**OptTar**
IPS values (normalised w.r.t. baseline)					
Body	0.97	0.86	0.98	0.95	0.95
Fluid	0.93	0.90	0.97	0.98	0.97
Freq	0.95	0.85	0.98	0.97	0.97
Vips	0.96	0.87	0.98	0.98	0.97
Black16	0.97	0.88	0.97	0.96	0.96
Body16	0.95	0.86	0.96	0.93	0.96
Fluid16	0.96	0.90	0.96	0.97	0.95
Swap16	0.96	0.88	0.97	0.98	0.98
Average degradation in IPS (w.r.t. baseline)					
	4.38%	**12.51%**	**2.87%**	**3.51%**	**3.62%**

Table 13. Change in IPS w.r.t. baseline.

5.5.2 Effect on Performance. The block migration during shutting down of the cache banks increases NoC workloads. Once this process is initiated at victim bank, the incoming requests at the victim are only stalled, while other cache banks work normally. After commencing of migration process, remapping is established and eventually the stalled requests are forwarded to the target. Moreover while handling additional load, target banks also experience slight increment in its capacity and conflict misses. The integrated effect of these factors stated above aggravates IPC by a maximum of 4.7% in case of OptTar. However, our policy puts a limit on this degradation as a performance constraint.

Table 13 presents the performance (IPS) degradation for all the benchmarks with both DVFS policies. DVFS_95, for all the applications, uses maximum frequency and steps it

Parameters	DVFS_95	DVFS_90	AltRow	Chess	OptTar
Max Peak Temp Reduction	5.7°C	6.6°C	4.6°C	4.5°C	5.5°C
Max Avg Temp Reduction	3.4°C	4.8°C	4.3°C	4.4°C	5.8°C
Std Dev of Peak Temp Change	1.91	1.93	0.13	0.19	0.31
Std Dev of Avg Temp change	1.05	0.85	0.43	0.34	0.64
IPS Degradation	4.38%	12.51%	2.87%	3.51%	3.62%

Table 14. Summary

down by one step whereas DVFS_90 uses three levels of frequencies for more reduction in temperature. Hence, overall execution time increases (i.e. the IPS reduces) for the applications in DVFS_90 than DVFS_95. The cache based policies does not modify the running frequencies of the cores, hence performance (IPS) degradation is much lesser than the DVFS policies, although dynamic cache resizing incurs an extra memory latency, which has been taken care of in our simulation. However, on an average, DVFS_95 and DVFS_90 degrade performance by 4.38% and 12.51%, respectively. The average performance degradation for all the cache based policies is lesser than 4%.

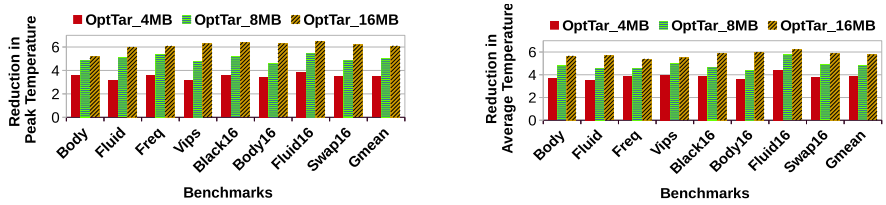
5.5.3 Summary. Table 14 summarises results for all the policies. DVFS_90 reduces both peak and average temperature at most by 6.6°C and 4.8°C, respectively, whereas OptTar (the best of our proposed policies) reduces peak temperature by 5.5°C and average temperature by 5.8°C. Although, OptTar reduces lesser peak but more average chip temperature than DVFS_90, but, DVFS_90 degrades performance by 12.51% which is 3.62% for OptTar.

5.6 Evaluating the Scalability

To show the scalability, we used OptTar and ran the same set of applications with 4MB and 16MB L2 cache having 64 banks. Figure 23 shows the reduction in peak and average chip temperature over various cache sizes across the applications. We could shutdown 48 banks in the case of 16MB cache and this reduced the peak and average temperatures by 6.1°C and 5.8°C on an average, respectively. Larger turned off banks create larger thermal buffers, so, more reduction in temperatures. Gating 48 banks in 16MB cache, also maintains the cache size at 4MB which provides lesser cache space to the applications, hence, the IPC degrades around 2%, on an average. Conversely, for 4MB L2 cache, powered off banks create smaller thermal buffers than larger ones, hence, lesser temperature reduction is obtained. The average reduction in peak temperature for 4MB is around 3.5°C, which is around 5.1°C in case of 8MB cache. Moreover, 4MB cache does not allow much resizing while maintaining performance, hence, the thermal benefit is lesser. Our performance cognizant dynamic resizing maintains the cache size according to the WSS of the applications. Thus, implementing our policy in larger caches gives more opportunity for reconfiguration and temperature control.

5.7 Effects on diverse application-set

To show the effectiveness of our proposal on a diverse application-set, we run *Body16* & *Swap16* together. Here, *Swap16* is computational load, whereas Body16 is the memory intensive. Hence, it has been seen that, *Body16* takes the upper-hand in case of cache access whereas, *Swap16* plays major role in cores' thermal behaviour. In this scenario, when OptTar is applied the cache resizing is sensitive mostly towards *Body16*, but, loads of *Swap16* also



(a) Peak Temperature of the Chip

(b) Average Chip Temperature

Fig. 23. Comparing temperature reduction (in °C) with different cache sizes.

however has minimal effects on cache temperature. But, core temperature is increased due to combined effects of both. Therefore, temperature reductions (both peak & average) are lesser than their individual execution. The average and peak temperature reductions for this workload are 4.3°C and 3.9°C, respectively. For combined workloads of *Black16* (compute intensive) & *Fluid16* (memory intensive), the average and peak temperature reductions are 5.1°C and 4.9°C, respectively.

6 RELATED WORK

In recent years, architects and designers are much more concentrating on DTM techniques while designing modern CMPs. A plethora of such recent techniques are discussed in [19]. These mitigations can either (a) minimise chip temperature for a given performance constraint, or (b) maximise performance for a predefined power budget and/or thermal constraint.

6.1 DVFS and Task Migration

DVFS and task migration are two most promising techniques to mitigate on-chip hotspots, by controlling dynamic power consumption of the cores. Leakage power, on the other hand, having a circular dependency with temperature, is efficiently modeled in [18]. Additionally, an exponential algorithm has been proposed to estimate upper bound on energy savings [43]. Even for a given thermal constraint, researchers have attempted to maximise throughput of the processors [35] by employing DVFS as backbone. Apart from DVFS, a latter exploration [15] proposes an efficient thread migration method for thermal efficiency. In another work [14], DVFS and thread migration are merged to achieve active scalable cooling of cores. Through integration of optimal control mechanism [45] or Model Predictive Controller [41], DVFS can also be achieved at circuit level.

Global thermal management suffers from scalability for modern CMPs having hundreds of cores. Hence, authors in [18] tried to reduce energy consumption in diverse runtime characteristics of the threads. Even DVFS and PCPG (Per Core Power Gating) together [21] can improve processors' throughput and reduce temperature in large CMPs. Moreover, insertion of idle time slice to a core with a thermal aware task scheduling can further reduce peak temperature of the chip [5]. The proactive dynamic task migration from hotter cores to colder area can reduce the chance for thermal imbalance [12, 13]. But frequent task shifting towards thermal efficiency leads to a costly on-chip ping-pong effect, which is addressed by Mizunuma in [30].

6.2 Cache in Thermal Management

The large LLCs of modern CMPs are accounted for their significant leakage power consumption [11, 19, 29, 44]. Reduction in LLC leakage can be done either by (a) cache resizing or by (b) reducing cache hotspots. The classical techniques for cache leakage reduction exploit both state preserving and state destroying policies [17, 23].

A *Power density-Minimized Architecture (PMA)* with a *Block Permutation Scheme (BPS)* [20] decreases cache temperature by exploiting Gated-Vdd [34]. Leakage also plays the pivotal role of thermal control for a performance aware thermal efficient technique, shadow tag [39, 40]. Noori et al. further analysed the increment in cache energy and temperature along with their inter-dependency for a 100nm technology [32]. In [33], Temperature Aware Cache Configuration (TACC) has been proposed to optimally reconfigure caches at different execution phases through both offline and online analyses of cache-usage. Sentry tags, in another work [10], eliminate unnecessary cache accesses to minimize power and temperature. Moreover, prediction based cache block migration also reduces cache temperature where migration is triggered due to overheating [3].

Modern 3D CMPs suffer from high power density, hence, the increased effective temperature [47]. A runtime thermal management is proposed for 3-D chip at way level granularity in [22] which combines DVFS with a novel thermal aware technique for hybrid (MRAM/SRAM) cache.

7 CONCLUSION

The paper explores the possibility of using dynamic cache resizing towards temperature control in CMPs. We provide a cache based thermal management technique for modern large centralised LLC based CMPs. The cores are placed along the periphery of the chip. Processor based techniques like DVFS are effective in temperature control, however have considerable performance impact.

Considering performance as a system wide constraint, we dynamically resize the LLC to create on-chip thermal buffer which reduces chip temperature. Three different patterns are proposed to get maximum reduction in average as well as peak temperature while maintaining performance within a limit. The simulation results, prepared by running PARSEC workloads, are further compared with the Greedy DVFS, a core based thermal management technique. Both Greedy DVFS and OptTar show maximum reduction in peak temperature by 6.6°C and 5.5°C, respectively, for an $8MB$ LLC. The temperature reduction is more with larger LLCs, due to more chances to create larger thermal buffers. Dynamic cache resizing overheads are taken into account in our simulation, which can be further overlapped with context switching for enhancing performance. Cache based policy shows better thermal stability than DVFS based ones, hence offers more circuit reliability.

In recent CMPs, the cache capacity and area is considerable, and hence, one can use cache reconfiguration to lower overall chip temperature with minimal impact on the performance. The results from the paper also show that LLCs can contribute to safeguard against thermal breakdown.

A CHANGE IN MEMORY CYCLES WHILE RESIZING CACHE

In Equation 12, as CC_i includes L1 access latency, hence, MC_i only represents L2 accesses as follows:

$$MC_i(b) = \sum_{j=1}^{b} (h_j^i.d_j^i + a_o^{ij}.d_o), b \leq B. \tag{22}$$

b represents number of banks currently ON, and B is the total number of available L2 banks. h_j^i is the number of hits at bank j, whose requests have been generated at core i. The d_j^i is the delay at bank j to send block to core i. a_o^{ij} implies the number of off-chip accesses due to misses at bank j which have been requested by core i, and (uniform) off-chip access latency is represented by d_o.

Initially, in Equation 22, if all banks are turned on, then $b = B$. Shutting down of cache banks introduces target banks, which incurs a few extra cycles for the remapped requests to reach at target. So, Equation 22 can be rewritten as follows.

$$MC_i(b) = \sum_{j=1}^{b}(h_j^i.d_j^i + a_o^{ij}.d_o) + \sum_{k=1}^{B-b} r_k^i.n_t^{ik}, b \leq B. \tag{23}$$

r_k^i is the number of remapped requests at $(B-b)$ number of turned off banks, generated by core i. Number of NoC cycles required to reach at target t (from k) is represented by n_t^{ik}.

During reconfiguration, system also needs a few clock cycles to move data from victim to target or vice versa, which is negligible if it is done limited number of times. Experimental results of the paper claim the proof of our statement. Finally, Equation 11 can be rewritten as:

$$IPC(b) = \frac{1}{N}\sum_{i=1}^{N} IPC_i(b) = \frac{1}{N}\sum_{i=1}^{N}(\frac{IC_i}{CC_i + MC_i(b)}) = \frac{1}{N}\sum_{i=1}^{N}(\frac{IC_i}{CC_i + \sum_{j=1}^{b}(h_j^i.d_j^i + a_o^{ij}.d_o) + \sum_{k=1}^{B-b} r_k^i.n_t^{ik}}) \geq C \tag{24}$$

Reduction in b reduces cache capacity, hence, the number of misses i.e. a_o^{ij} increases, so, decrement in h_j^i (Equation 24). As, $d_o > d_j$, so, increment in a_o^{ij} curtails performance by increasing MC_i. Moreover, r_k^i also increases for more reduction in b. So, Equation 24 clearly shows how performance is related to the number of cache banks (b).

REFERENCES

[1] 2011. Oracle's SPARC T3-1, SPARC T3-2, SPARC T3-4 and SPARC T3-1B Server Architecture. http://www.oracle.com/. [Online; accessed 19-Jan-2017].

[2] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha. 2009. GARNET: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. 33–42.

[3] R. Ayoub and A. Orailoglu. 2010. Performance and Energy Efficient Cache Migration approach for Thermal Management in Embedded Systems. In *Proceedings of the 20th Symposium on Great Lakes Symposium on VLSI*. 365–368.

[4] C. Bienia, S. Kumar, J. P. Singh, and K. Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT '08)*. 72–81.

[5] T. Chantem, R. P. Dick, and X. S. Hu. 2008. Temperature-aware Scheduling and Assignment for Hard Real-time Applications on MPSoCs. In *Proceedings of the Conference on Design, Automation and Test in Europe*. 288–293.

[6] A. K. Coskun, T. Simunic Rosing, and K. Whisnant. 2007. Temperature Aware Task Scheduling in MPSoCs. In *Proceedings of the Conference on Design, Automation and Test in Europe*. 1659–1664.

[7] S. Das and H. K. Kapoor. 2016. A Framework for Block Placement, Migration, and Fast Searching in Tiled-DNUCA Architecture. *ACM Trans. Des. Autom. Electron. Syst.* 22, 1 (May 2016), 4:1–4:26.

[8] H. Everett. 1963. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Oper. Res.* 11, 3 (June 1963), 399–417.

[9] S. Eyerman and L. Eeckhout. 2011. Fine-grained DVFS Using On-chip Regulators. *ACM Trans. Archit. Code Optim.* 8, 1 (Feb. 2011), 1:1–1:24.

[10] M. Farahani and A. Baniasadi. 2009. Temperature Reduction Analysis in Sentry Tag Cache Systems. In *Proceedings of the 10th Workshop on MEmory Performance: DEaling with Applications, Systems and Architecture*. 22–27.

[11] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. 2002. Drowsy caches: simple techniques for reducing leakage power. In *Proceedings 29th Annual International Symposium on Computer Architecture*. 148–157.

[12] Y. Ge, P. Malani, and Q. Qiu. 2010. Distributed task migration for thermal management in many-core systems. In *Design Automation Conference*. 579–584.

[13] Y. Ge, Q. Qiu, and Q. Wu. 2012. A Multi-Agent Framework for Thermal Aware Task Migration in Many-Core Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20, 10 (2012), 1758–1771.

[14] V. Hanumaiah, S. Vrudhula, and K. S. Chatha. 2009. Maximizing performance of thermally constrained multi-core processors by dynamic voltage and frequency control. In *IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*. 310–313.

[15] V. Hanumaiah, S. Vrudhula, and K. S. Chatha. 2011. Performance Optimal Online DVFS and Task Migration Techniques for Thermally Constrained Multi-Core Processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 11 (Nov 2011), 1677–1690.

[16] H. Homayoun, M. Rahmatian, V. Kontorinis, S. Golshan, and D. M. Tullsen. 2012. Hot peripheral thermal management to mitigate cache temperature variation. In *Thirteenth International Symposium on Quality Electronic Design*. 755–763.

[17] S. Kaxiras, Z. Hu, and M. Martonosi. 2001. Cache decay: exploiting generational behavior to reduce cache leakage power. In *Proceedings 28th Annual International Symposium on Computer Architecture*. 240–251.

[18] W. Kim, M. S. Gupta, G. Y. Wei, and D. Brooks. 2008. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *IEEE 14th International Symposium on High Performance Computer Architecture*. 123–134.

[19] J. Kong, S. W. Chung, and K. Skadron. 2012. Recent Thermal Management Techniques for Microprocessors. *ACM Comput. Surv.* 44, 3 (June 2012), 13:1–13:42.

[20] J. C. Ku, S. Ozdemir, G. Memik, and Y. Ismail. 2005. Thermal Management of On-Chip Caches Through Power Density Minimization. In *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture*.

[21] J. Lee and N. S. Kim. 2012. Analyzing Potential Throughput Improvement of Power- and Thermal-Constrained Multicore Processors by Exploiting DVFS and PCPG. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20, 2 (Feb 2012), 225–235.

[22] S. Lee, K. Kang, and C. M. Kyung. 2015. Runtime Thermal Management for 3-D Chip-Multiprocessors With Hybrid SRAM/MRAM L2 Cache. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 3 (March 2015), 520–533.

[23] L. Li, I. Kadayif, Y. F. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and A. Sivasubramaniam. 2002. Leakage energy management in cache hierarchies. In *Proceedings.International Conference on Parallel Architectures and Compilation Techniques*. 131–140.

[24] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 469–480.

[25] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. 2002. Simics: A full system simulation platform. *Computer* 35, 2 (Feb 2002), 50–58.

[26] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. 2005. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. *SIGARCH Comput. Archit. News* 33, 4 (2005), 92–99.

[27] M. Martonosi, S. Malik, and F. Xie. 2005. Efficient behavior-driven runtime dynamic voltage scaling policies. In *Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. 105–110.

[28] A. Mirtar, S. Dey, and A. Raghunathan. 2015. Joint Work and Voltage/Frequency Scaling for Quality-Optimized Dynamic Thermal Management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 6 (June 2015), 1017–1030.

[29] S. Mittal. 2014. A survey of architectural techniques for improving cache power efficiency. *Sustainable Computing: Informatics and Systems* 4, 1 (2014), 33 – 43.

[30] H. Mizunuma, Y. C. Lu, and C. L. Yang. 2013. Thermal coupling aware task migration using neighboring core search for many-core systems. In *International Symposium on VLSI Design, Automation, and Test (VLSI-DAT)*. 1–4.

[31] R. Balasubramonian N. Muralimanohar and N. P. Jouppi. 2007. CACTI 6.0: A Tool to Model Large Caches. In *Technical Report HPL-2009-85, HP Laboratories*.

[32] H. Noori, M. Goudarzi, K. Inoue, and K. Murakami. 2007. The Effect of Temperature on Cache Size Tuning for Low Energy Embedded Systems. In *Proceedings of the 17th ACM Great Lakes Symposium on VLSI*.

[33] H. Noori, M. Goudarzi, K. Inoue, and K. Murakami. 2008. Improving Energy Efficiency of Configurable Caches via Temperature-Aware Configuration Selection. In *IEEE Computer Society Annual Symposium on VLSI*. 363–368.

[34] M. Powell, S. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. 2000. Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-submicron Cache Memories. In *Proceedings of International Symposium on Low Power Electronics and Design*.

[35] R. Rao, S. Vrudhula, C. Chakrabarti, and N. Chang. 2006. An Optimal Analytical Solution for Processor Speed Control with Thermal Constraints. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 292–297.

[36] B. Salami, M. Baharani, and H. Noori. 2014. Proactive Task Migration with a Self-adjusting Migration Threshold for Dynamic Thermal Management of Multi-core Processors. *J. Supercomput.* 68, 3 (June 2014), 1068–1087.

[37] A. Sembrant, E. Hagersten, and D. Black-Shaffer. 2013. TLC: A tag-less cache for reducing dynamic first level cache energy. In *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 49–61.

[38] K Stavrou and P Trancoso. 2005. TSIC: Thermal Scheduling Simulator for Chip Multiprocessors. In *Proceedings of the 10th Panhellenic Conference on Advances in Informatics*. Springer-Verlag, 589–599.

[39] G. Sun, X. Wu, and Y. Xie. 2009. Exploration of 3D Stacked L2 Cache Design for High Performance and Efficient Thermal Control. In *Proceedings of International Symposium on Low Power Electronics and Design*.

[40] G. Sun, H. Yang, and Y. Xie. 2012. Performance/Thermal-Aware Design of 3D-Stacked L2 Caches for CMPs. *ACM Trans. Des. Autom. Electron. Syst.* 17, 2 (April 2012).

[41] H. Wang, J. Ma, S. X.-D. Tan, C. Zhang, H. Tang, K. Huang, and Z. Zhang. 2016. Hierarchical Dynamic Thermal Management Method for High-Performance Many-Core Microprocessors. *ACM Trans. Des. Autom. Electron. Syst.* 22, 1 (Aug. 2016), 1:1–1:21.

[42] H. Wang, X. Zhu, L. Peh, and S. Malik. 2002. Orion: a power-performance simulator for interconnection networks. In *35th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 294–305.

[43] Fen Xie, M. Martonosi, and S. Malik. 2005. Bounds on power savings using runtime dynamic voltage scaling: an exact algorithm and a linear-time heuristic approximation. In *Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.* 287–292.

[44] W. Zang and A. Gordon-Ross. 2013. A Survey on Cache Tuning from a Power/Energy Perspective. *ACM Comput. Surv.* 45, 3 (July 2013), 32:1–32:49.

[45] F. Zanini, D. Atienza, C. N. Jones, L. Benini, and G. De Micheli. 2013. Online Thermal Control Methods for Multiprocessor Systems. *ACM Trans. Des. Autom. Electron. Syst.* 18, 1, Article 6 (Jan. 2013), 6:1–6:26 pages.

[46] R. Zhang, M. R. Stan, and K. Skadron. 2015. HotSpot 6.0: Validation, Acceleration and Extension.. In *University of Virginia, Tech. Report CS-2015-04*.

[47] Y. Zhang, L. Li, Z. Lu, A. Jantsch, M. Gao, H. Pan, and F. Han. 2014. A Survey of Memory Architecture for 3D Chip Multi-processors. *Microprocess. Microsyst.* 38, 5 (July 2014), 415–430.