WaFFLe: Gated Cache-Ways with Per-core Fine-grained DVFS for Reduced On-chip Temperature and Leakage Consumption

SHOUNAK CHAKRABORTY and MAGNUS SJÄLANDER, Department of Computer Science, Norwegian University of Science and Technology (NTNU), Norway

Managing thermal imbalance in contemporary chip multi-processors (CMPs) is crucial in assuring functional correctness of modern mobile as well as server systems. Localized regions with high activity, e.g., register files, ALUs, FPUs, etc., experience higher temperatures than the average across the chip and are commonly referred to as hotspots. Hotspots affect functional correctness of the underlying circuitry and a noticeable increase in leakage power, which in turn generates heat in a self-reinforced cycle. Techniques that reduce the severity of or completely eliminate hotspots can maintain functional correctness along with improving performance of CMPs. Conventional dynamic thermal management targets the cores to reduce hotspots but often ignores caches, which are known for their high leakage power consumption.

This paper presents *WaFFLe*, an approach that targets the leakage power of the last level cache (LLC) and hotspots occurring at the cores. *WaFFLe* turns off LLC-ways to reduce leakage power and to generate on-chip thermal buffers. In addition, fine-grained DVFS is applied during long LLC miss induced stalls to reduce core temperature. Our results show that *WaFFLe* reduces peak and average temperature of a 16-core based homogeneous tiled CMP with up to 8.4 °C and 6.2 °C, respectively, with an average performance degradation of only 2.5 %. We also show that *WaFFLe* outperforms a state-of-the-art cache-based technique and a greedy DVFS policy.

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

Additional Key Words and Phrases: Cache-way shutdown, Fine grained DVFS, Thermal efficiency, Hotspots, Leakage power, Cache coherence, Chip-mutiprocessors

1 INTRODUCTION

The end of Dennard scaling has caused power densities to escalate along with associated thermal issues, which have become a primary obstacle in recent microprocessor chip design [28]. Regulating the source voltage or power gating a portion of a chip are classical techniques to manage power and temperature, but applying restrictions in the power supply requires efficient management schemes to prevent performance degradation [6, 41, 45, 46]. To handle the ever-increasing size of run-time workloads of recent applications [31, 44, 53], modern chip multi-processors (CMPs) are equipped with out-of-order (OoO) cores that offer higher performance as compared to their in-order (InO) counterparts. But, the improved performance offered by OoO cores is achieved at the cost of increased energy usage that noticeably increases on-chip temperature, which can cause thermal breakdown of the underlying circuitry if not kept in check.

Authors' address: Shounak Chakraborty, shounak.chakraborty@ntnu.no; Magnus Själander, magnus.sjalander@ntnu.no, Department of Computer Science,
Norwegian University of Science and Technology (NTNU), Sem Sælandsvei 9, Gløshaugen, Trondheim, Norway, 7491.

XXXX-XXXX/2021/1-ART1
https://doi.org/10.1145/3471908

Specifically, highly active microarchitecture units (e.g., register files, ALUs, FPUs, etc.) of the OoO cores are susceptible to suffer from local hotspots. Additionally, the presence of large on-chip last-level caches (LLCs) with proximity to these hotspots increases the LLC temperature [12, 13]. The increased LLC temperature results in increased leakage power consumption, which in turn generates heat in a self-reinforced cycle and can affect the functional correctness of the circuit [50]. However, a significant portion of the LLC often remains underutilized [34, 35]. Thus, dynamic power gating of underutilized cache portions, which also generate on-chip thermal buffers, can be an apt choice for both energy and thermal efficiency enhancement [13, 47]. The majority of prior thermal-management techniques govern the individual core's power supply for peak temperature reduction by employing DVFS and/or task-migration [14, 18, 19, 28, 43]. On the other hand, prior cache based policies reduce leakage without comprehensively analyzing the on-chip thermal status [11, 25, 51]. Basically, effective reduction in peak temperature by the core-based policies reduces spatial thermal variance on-chip, which can further be extended temporally by keeping an underutilized cache portion power gated for a certain time-span on-the-fly.

In this work, we perform a comprehensive study on how dynamic way-shutdown based LLC resizing together with a fine-grained DVFS (FG-DVFS) of the cores impacts the spatio-temporal thermal behavior of a CMP. *WaFFLe* incorporates a performance-cognizant way-shutdown mechanism for the LLC that power-gates the cache-ways by invalidating clean non-MRU cache blocks (based on their availability) from each cache set. Moreover, unlike prior dynamic cache-resizing techniques [12, 13], *WaFFLe* enables normal cache accesses during the resizing process. Shutting down cache-ways reduces on-chip power consumption and assists in reducing average chip temperature by acting as on-chip thermal buffers, but its effect on peak temperature is limited compared to core-based techniques [51]. Hence, *WaFFLe* also incorporates an efficient FG-DVFS strategy to scale down voltage and frequency (V/F) settings of the cores during stalls induced by off-chip memory accesses, i.e., during LLC misses. The stalls are determined by accessing information at the LLC directory (in the coherence controller). In fact, the opportunity to apply FG-DVFS increases with the reduction in LLC size, since the number of misses are likely to increase. *WaFFLe*'s dynamic LLC resizing prudentially balances thermal efficiency against performance. Our simulation results show the presence of such long uninterrupted stalls that can be exploited to apply FG-DVFS and reduce the peak temperature without a significant impact on performance.

In brief, *WaFFLe* provides the following contributions in reducing on-chip temperature as well as power consumption:

(1) **Dynamic cache-way shutdown.** Identifying the potential of performing a dynamic cache-way shutdown in mitigating hotspots by generating thermal buffers on-chip. The cache resizing overheads are limited as normal cache operations are not stranded during the resizing process and writing back of dirty blocks has been minimized.
(2) **Fine-grained DVFS during memory stalls.** Efficient detection and exploitation of stall cycles at the cores (induced by on-chip LLC misses) for applying FG-DVFS to reduce peak on-chip temperature. To the best of our knowledge, *WaFFLe* is the first approach that exploits coherence information in enabling FG-DVFS at the cores during LLC-miss induced stalls.
(3) **Thermal efficiency.** By combining cache-way shutdown and FG-DVFS, *WaFFLe* reduces both the peak and average temperatures of a 16-core homogeneous CMP, by 8.4 °C and 6.2 °C, respectively, with only a slight degradation in performance. In fact, *WaFFLe* offers better thermal gains than its individual techniques (cache-way shutdown and FG-DVFS) in case of both peak and average chip temperature. Moreover, the core based policy is benefited by a (slightly) higher number of LLC misses caused by the dynamic cache-way shutdown. *WaFFLe*

also outperforms several state-of-the-art thermal management policies. Our results show that *WaFFLe* reduces the spatial thermal variance of a 16-core CMP by 4.5 °C, on average (Sec. 5.6).

(4) **Energy efficiency.** With the significant reduction in chip temperature, *WaFFLe* also reduces the overall leakage power consumption, which improves the energy delay product (EDP) of the CMP by 25.2 %, on average (Sec. 5.3).

The proposed performance-aware dynamic cache-resizing technique (Sec. 3.1) strives to stabilize the on-chip temporal thermal variation by dynamically generating and maintaining on-chip thermal buffers, while the proposed V/F governor (Sec. 3.2) takes advantage of memory stalls, which are further induced by the dynamic cache-resizing technique. The combined effect of the two techniques (i.e, cache resizing and fine-grained DVFS) results in a solution with low spatial and temporal thermal variation with a negligible performance degradation of 2.5%. The proposed solution outperforms multiple state-of-the-art thermal management techniques [12, 36].

Paper Organization. The rest of the paper is organized as follows. After discussing some relevant backgrounds in Sec. 2, we illustrate the detailed mechanism of *WaFFLe* in Sec. 3. Next, we present details of our simulation methodology in Sec. 4 and in Sec. 5, we present our results and analysis. The relevant prior research work are discussed in Sec. 6 before concluding the paper in Sec. 7.

2 BACKGROUND

Effects of Dynamic Cache Resizing. Temperature of any on-chip component obeys the basic *superposition and reciprocity* principle of heat transfer, that is driven by three prime factors: *(1)* the component's own power consumption, *(2)* heat abduction by ambient, and *(3)* conductive heat transfer with its peers [47]. Hence, intelligent selection of LLC-ways for powering off on-the-fly can potentially reduce the chip temperature [2, 4, 12, 13] by *(a)* curtailing its own power consumption, and *(b)* incorporating heat transfer at the generated on-chip thermal buffers, while maintaining performance. Note that, shutting down LLC-ways does not incorporate significant overhead of book-keeping and remapping of future requests as incurred by bank-shutdown based policies [12].

DVFS and On-chip Voltage Regulators. A plethora of prior thermal management policies employ DVFS at the cores [28], but the majority of these techniques apply DVFS at a coarse-grained time-scale by employing off-chip voltage regulators. Such techniques can reduce peak temperatures, but potentially aggravate core performances due to slow voltage-switching speeds. The promise of significantly smaller timing overhead makes on-chip voltage regulators a prospective candidate to replace its slow off-chip counterparts [3, 9, 16, 50]. On-chip voltage regulators enable fine-grained tuning of a core's V/F settings, which enables DVFS during relatively short core-stall cycles, e.g., caused by LLC misses. On-chip voltage regulators might pose their own challenges regarding power consumption and hotspots, but these can be addressed by techniques like ThermoGater [26].

3 *WAFFLE*

WaFFLe employs *(i)* power gating of cache-ways in a physically-distributed yet logically shared multi-banked LLC to create thermal buffers and *(ii)* fine-grained DVFS at the cores to reduce hotspots by taking advantage of stalls caused by long-latency memory operations.

3.1 Power-Gated Cache-ways

Power gating of a cache-way is triggered by changes in the miss ratio for each individual cache bank. If the (fractional) change in the miss ratio is smaller (or larger) than a provided *POWER_DOWN* (or *POWER_UP*) threshold, then a cache-way is powered off (or powered on). The miss ratio is computed from the number of cache accesses (*#accesses*) and misses (*#misses*) that are collected for each cache bank during a set interval (*Interval*).

Once a way shutdown decision is triggered, the data of the victim (the way to be gated) need to be evicted or invalidated. Writing back dirty data to main memory may degrade performance and increases power consumption due to increased off-chip memory accesses. Furthermore, if live data are evicted then the data will be fetched once more, which will further degrade performance. To mitigate these issues, *WaFFLe* attempts to evict *clean non-MRU (CN)* blocks from each set by employing a proactive search and intra cache-set move operation. For each set where the victim way contains dirty or MRU data another way containing a CN block is identified, if available, to which the victim data are written. This eliminates the write to off-chip memory at the cost of an additional write to the cache bank.

Specifically, a single write-back to off-chip memory from the LLC includes delays for the following operations that are performed in sequence: *(a)* read the evicted block (d_{rd}), *(b)* transfer the block from the cache set to the write buffer of the memory controller (d_{tran}), *(c)* writing and reading the write buffer ($2 \times d_{buff\_MMC}$), *(d)* waiting at the write buffer (d_{wait}), and *(e)* writing the data to main memory (d_{wr_MM}). On the other hand, a single intra-set move operation is done by embedding a temporary buffer, as the read and write at the cache set is done through the peripheral circuitry. These transfers include the time required for reading (d_{rd}) and writing (d_{wr}) the block at the LLC and the temporary buffer ($2 \times d_{buff}$). The total time taken for a write-back (T_{wb}) and a intra-set migration (T_{mig}) can then be expressed as:

$$T_{wb} = d_{rd} + d_{tran} + 2 \times d_{buff\_MMC} + d_{wait} + d_{wr\_MM} \tag{1}$$

and

$$T_{mig} = d_{rd} + d_{wr} + 2 \times d_{buff} \tag{2}$$

For an intra-set move to be beneficial, the following condition needs to be fulfilled $T_{mig} < T_{wb}$, which expands to $d_{rd} + d_{wr} + 2 \times d_{buff} < d_{rd} + d_{tran} + 2 \times d_{buff\_MMC} + d_{wait} + d_{wr\_MM}$ and reduces to $d_{wr} < d_{tran} + d_{wait} + d_{wr\_MM}$, assuming that the write buffer in the memory controller and the temporary buffer has equal delays for reading and writing (i.e., $d_{buff} = d_{buff\_MMC}$). Thus, an intra-set move is beneficial since an off-chip write operation to main memory is normally longer than an on-chip access to an LLC cache block. Moreover, writing back to the off-chip main memory consumes more power than an on-chip intra-set block migration. Hence, we use an intra-set block migration when possible during cache-way shutdown.

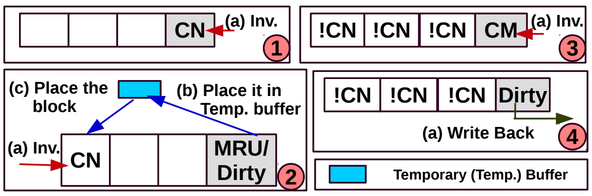


Fig. 1. Managing data at victim way during way-shutdown of a 4-way cache. Gray cells are the victim way.

Data eviction for a victim way consists of four different cases that have to be handled (shown in Figure 1): ① The data are not dirty and it is not the MRU block (CN), which is the simplest case where the block is simply invalidated. ② The data are either dirty or it is the MRU block, and there exists at least one other clean block in the set. The victim block is then read and replaces the identified CN block, avoiding eviction of potential live data or writing back dirty data. ③ The data are clean and it is the MRU (CM) block, but all other blocks in the set are dirty, causing the block to be invalidated. (4) All blocks in the set are dirty and the dirty data of the victim are written back to the main-memory.

3.1.1 Way-Shutdown Management Algorithm. The complete algorithm for performing way-shutdown is presented in Algorithm 1. The maximum number of ways that are allowed to be shutdown is configurable ($\#Way_{Limit}$) and is taken as an input to the algorithm together with the above-mentioned variables and the number of sets ($\#sets$) in the banks. The $ratio$ is calculated by $\#misses(B)/\#accesses(B)$ for individual banks on the completion of an interval ($Interval$) (line 4 to 5). If the $ratio$ is smaller than $POWER\_DOWN$, and the number of turned off ways ($\#Off\_ways$) is less than the maximum allowed ($\#Way_{Limit}$) (line 6) then a way is shutdown. The victim way is selected from the edge of the bank as it increases the probability of generating a thermal buffer adjacent to the cores (line 7). Before turning off the way, each set (line 9) is checked for the four cases described in Figure 1 (line 10 to 15). While the victim way is being evicted the bank can still serve external memory accesses (unlike prior cache resizing policies[1] [10, 12, 13]). The main difference is that on an eviction caused by a cache miss, the selected way to be evicted cannot be the victim way. Once all sets have been invalidated, the way is turned off (line 17). On the other hand, if the $ratio$ is larger than $POWER\_UP$ and there exists at least one power-gated way (line 20) then a way is turned on (line 21). Note that, by using two separate limits where $POWER\_UP$ is larger than $POWER\_DOWN$ reduces the probability for oscillating resizing, where one way is repeatedly powered on and off during stable execution phases.

A suitable $Interval$ length, $POWER\_UP$ and $POWER\_DOWN$ thresholds depend on system parameters, and the average expected workload of the system (see Sec. 4). Hence, these may be set at design time or made configurable. MRU status and dirty bits are commonly stored together with the data in SRAMs resulting in the number of sets that can be evicted per cycle (so, the size of the temporary buffer in Figure 1) to be limited by the number of memory ports.

3.1.2 Preliminary analysis. We analyzed nine PARSEC applications [7] to determine the likelihood of the four described cases for eviction. We simulated each application in gem5 [8] for 80 million cycles (see Sec. 4) and collected

- the maximum and average number of clean non-MRU blocks per set (*CN*)
- the number of dirty blocks per way,
- the number of MRU blocks per way, and
- the number of clean and dirty MRU blocks per way (*CM* and *DM*).

We also observed three differently accessed LLC banks (heavily (*Heavy*), moderately (*Moderate*), and lightly (*Light*)) to determine the average number of clean blocks per set and the maximum percentage of dirty blocks in a cache-way.

Figure 2 and 3 show a summary of the collected parameters. For dirty blocks per way (Figure 3), the way with the highest number of dirty blocks is shown for a heavily accessed LLC-bank. Figure 2 shows that there exist, on average, six clean non-MRU (CN) blocks in a set for a heavily used LLC-bank. We report the percentages of MRU, CM, and DM blocks of the same cache-way of a heavily accessed bank, which is depicted in Figure 3. However, to realize the complete scenario of the highest number of dirty blocks per way in differently accessed banks, we further plot Figure 4. The average number of CN blocks per set is depicted in Figure 5, for three differently accessed LLC-banks, which shows on an average all of these three types of banks maintain five to six CN blocks per set. These observations lead us to conclude the following:

- Shutting down of a cache-way by simply invalidating or writing back cache blocks might degrade performance severely, since the way may contain many dirty blocks.

[1]where performance degrades as normal cache operations are stranded during the resizing process

Algorithm 1: Way-shutdown management

```
   Input: $Interval$, $POWER\_DOWN$, $POWER\_UP$, $Way_{Limit}$, $\#sets$
1  $\#Off\_ways = 0$;
2  while System is running do
3      if Interval == completed then
4          For each bank (B) do in parallel (line 5 to 24);
5          $ratio = \#misses(B)/\#accesses(B)$;
6          if (ratio < POWER_DOWN) and (#Off_ways < $Way_{Limit}$) then
7              Select a turned on way as victim (V) from bank-edge;
8              $i = 0$;
9              while $i$ < #sets do
10                 if CN exists and V(i) == (MRU or dirty) then
11                     Read the victim block ($V(i)$) and write it to a clean non-MRU (CN) block in set $i$;
12                 if No CN exists and V(i) == dirty then
13                     Write $V(i)$ back to memory;
14
15                 Invalidate $V(i)$;
16                 i++;
17             Power gate the victim way (V);
18             $\#Off\_ways$++;
19         else
20             if (ratio > POWER_UP) and (#Off_ways > 0) then
21                 Power on a way;
22                 $\#Off\_ways$--;
23             else
24                 # Steady state, no change;
25     else
26         # Execute as normal;
```

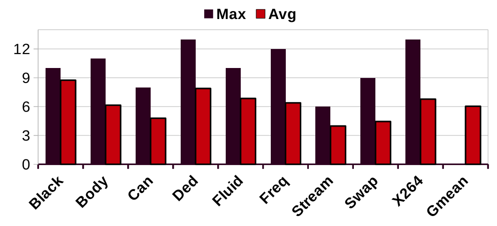


Fig. 2. Maximum and average number of clean non-MRU blocks per cache-set for a heavily used LLC-bank of size 1MB (16-way).

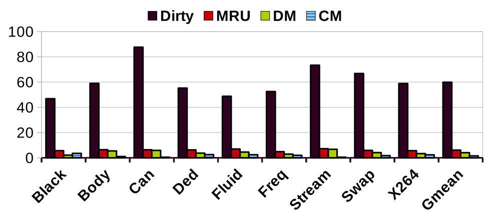


Fig. 3. Maximum percentage of dirty blocks per cache-way for a heavily used LLC-bank of size 1MB (16-way). Percentages of MRU, DM and CM blocks for the same way are also shown.

- Cases $\textcircled{3}$ and $\textcircled{4}$ (in. Figure 1) rarely happen since less than 2% of the cache-sets contain *only dirty data*, on average.
- Figure 3 shows that on average 60% (ranging between $47 - 88\%$ for a heavily used bank) of the victim way's data are dirty resulting in the data being copied, i.e., Case $\textcircled{2}$ (in. Figure 1), while for the remaining 40% the blocks are simply invalidated, i.e., Case $\textcircled{1}$. Note that, the observed behavior is similar for the cases of moderately and lightly used banks.

We further analyzed the efficacy of two different way shutdown mechanisms: *(a)* WS_WB—the cache-way is gated by simply writing back dirty data to off-chip memory while clean blocks is

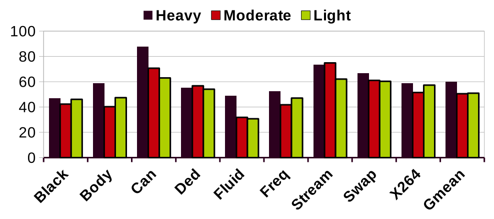


Fig. 4. Maximum percentage of dirty blocks per cache-way for three differently used LLC-banks (each one is 1MB (16-way)).

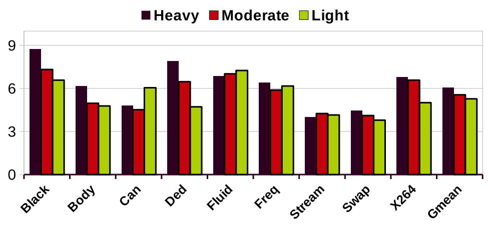


Fig. 5. Average number of CN blocks per cache set for three differently used LLC-banks (each one is 1MB (16-way)).

simply invalidated, and *(b)* WS_*WaFFLe*—the way shutdown is performed as per Algorithm 1. Figure 6 shows the increases in miss rates for nine PARSEC applications with both way-shutdown mechanisms and in most of the cases higher miss rates are experienced in WS_WB. As WS_*WaFFLe* attempts to keep MRU data on-chip by incorporating swapping technique, the increase in miss rate is comparatively smaller than WS_WB. Moreover, writing data to off-chip is temporally costlier than the on-chip block swapping technique, further aggravating the overall performance, which is shown in Figure 7. Trivially, WS_*WaFFLe* outperforms WS_WB for most of the nine PARSEC applications. On an average, WS_*WaFFLe* incurs nearly 3% degradation in performance, as compared to 5.5% by WS_WB.

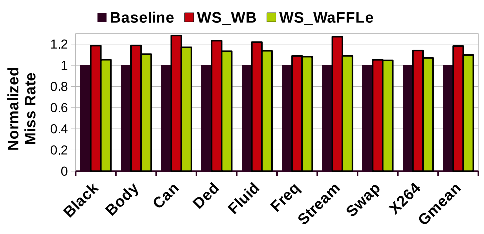


Fig. 6. Miss-rates for two way-shutdown policies.

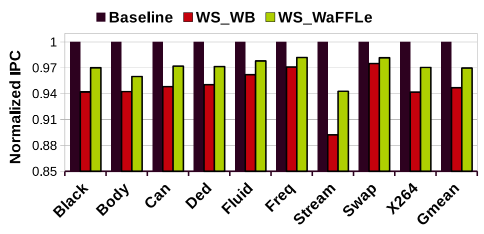


Fig. 7. IPCs for two way-shutdown policies.

3.1.3 Resizing Overheads. LLC resizing is implemented on a multi-level inclusive cache hierarchy, where multiple copies of a single cache block might reside at different cache levels. Hence, while resizing LLC, our technique always tries to invalidate clean blocks, which are to be simply invalidated at the LLC along with its local copies at the sharers. For the cases ①, ② and ③ in Figure 1, a clean block is finally invalidated at the LLC. Hence, before invalidating the LLC copy, the sharers' copies at the local caches are invalidated. For case ④ in Figure 1, before writing back the dirty data, its local copies will be written back to LLC (if needed), and then data will be evicted from the LLC, subsequently. The first three cases will not incorporate any noticeable (temporal) overheads, as the data have to be simply invalidated. However, case ④ in Figure 1 might incur additional temporal overheads, but, as per our analysis (see Figure 2 to 5), this is a rare event to handle during resizing, hence, has negligible impact on the performance. We have evaluated all such issues in our simulation framework at the memory module of the gem5 simulator [8] (see Sec. 4).

3.2 Fine-Grained DVFS

WaFFLe uses DVFS to reduce hotspots appearing at the cores by lowering the voltage and frequency when an LLC miss is detected, which otherwise might result in the OoO core to be stalled. The detection of an LLC miss might not mean that the full time of an off-chip access will be observed due to memory level parallelism (MLP). If a core makes multiple accesses to the same cache block, only the first access (referred to as an isolated LLC miss) will see the full time of the memory access. Thus, only LLC misses that do not have a miss handling status register (MSHR) allocated trigger DVFS to be applied. If a core accesses different cache blocks, it is the most recent block request that will determine when the data is returned (neglecting potential memory controller scheduling interference). Thus, for each LLC miss that does not have an MSHR the period for which DVFS can be applied is renewed. In a multi-core system, an isolated LLC miss at one core might not mean that the full time of an off-chip access will be observed since another core might already have requested the same block, thus, reducing the effective time before the block is returned. Only the core that is first to issue the request for the block will experience the full time for retrieving it (referred to as a global isolated LLC miss).

Detecting if an LLC miss is a local-isolated miss is trivial by checking if a local MSHR is already allocated. However, to know if the full access time will be experienced by the core it is necessary to know if it is also a global isolated LLC miss, which is more challenging to determine as MSHRs are local to the cores, and hence, are transparent to the global information. Therefore, to efficiently determine isolation status of a particular requested block, *WaFFLe* looks into the LLC directory that not only keeps track of the present cache blocks but also contains entries for the missed cache blocks for which requests have already been forwarded to the off-chip memory. If the current miss is detected as a global isolated one by looking at its directory entry, the information will be immediately sent to the respective requester core's controller to enable FG-DVFS.

3.2.1 Mechanism. Figure 8 shows a graphical representation of the process, that triggers FG-DVFS at the cores. In this figure, a 16-core based CMP is considered ($P0$ to $P15$), where each core is equipped with its local data and instruction L1 caches. The physically distributed (multi-banked) yet logically shared L2 cache is considered as our LLC. Once an L1 (D/I) miss is detected, the control is transferred to check the presence of the block in the L2 like conventional cache design. On an L2 miss, our logic for FG-DVFS inspects the status of the missed block in the LLC-directory for detecting its global isolation status. Each of the directory entries contains the address of the requested block, the coherence state, the request type (read/write) and the bit vector. The bit vector keeps track of the requester/sharers of a particular cache block. In this figure, the MSB (most significant bit) of the bit vector represents core $P0$ and core $P15$ is represented by the LSB (least significant bit). We show a possible case to detect a global isolated miss, where the same address (`x005`) is requested multiple times by different cores with overlapped timelines.

Once an LLC miss takes place, the controller first attempts to determine and locks the cache space in the data array[2] where the block will be placed upon its arrival from off-chip memory. In addition, a directory entry will be created (at the coherence controller) to keep track of the metadata related to the cache block (as we discussed above). The coherence state is usually set to a particular transient state (*CS1* in our example), when the request for a block has been sent to main memory and the block is yet to arrive. In Figure 8, `x005` is only shared by *P0* at the beginning in the second entry of the directory, for which the MSB of the bit vector is set. As there is no other outstanding misses requested by $P0$ at the moment, the miss is locally isolated and as all other bits are 0 in the bit vector, this miss is declared as a global isolated miss, and subsequently, FG-DVFS will be

[2]Note that, this operation follows the cache eviction process, while the respective cache set has no empty space.

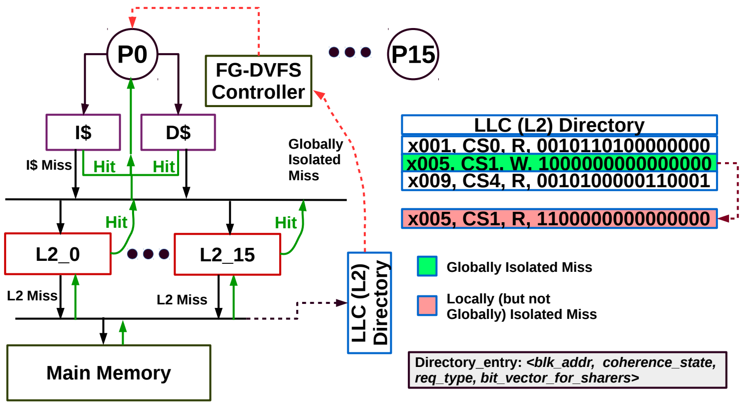

P0
P15
FG-DVFS Controller
I$
D$
I$ Miss
Hit
Hit
Globally Isolated Miss
Hit
Hit
L2_0
L2_15
L2 Miss
L2 Miss
Main Memory
LLC (L2) Directory
LLC (L2) Directory
x001, CS0, R, 0010110100000000
x005, CS1, W, 1000000000000000
x009, CS4, R, 0010100000110001
x005, CS1, R, 1100000000000000
Globally Isolated Miss
Locally (but not Globally) Isolated Miss
Directory_entry: <blk_addr, coherence_state, req_type, bit_vector_for_sharers>

Fig. 8. Local and global isolated LLC miss detection and triggering of FG-DVFS.

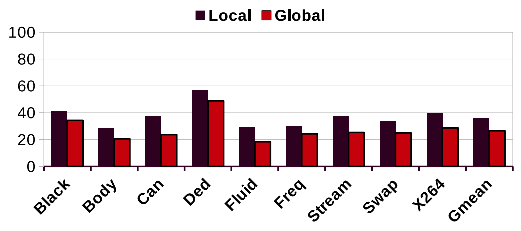


Fig. 9. Percentage of isolated misses for the PARSEC applications.

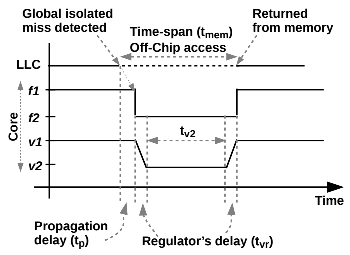

Global isolated miss detected
Time-span (t_{mem}) Off-Chip access
Returned from memory
LLC
f1
f2
v1
v2
Core
t_{v2}
Time
Propagation delay (t_p)
Regulator's delay (t_{vr})

Fig. 10. Timing-diagram of FG-DVFS (not to scale).

enabled at $P0$. Later, x005 is further requested by another core $P1$ (for which the change in entry is shown in red), that has an overlapped time-line with the request from $P0$, hence, the full stall length will not be observed at $P1$, and thus, it will not be eligible for FG-DVFS. Note that, in case of the last entry, x005 is a local isolated miss at $P1$, but, is not qualified as a global isolated one.

3.2.2 Analytical Background. To determine the efficacy of our proposed FG-DVFS policy, we further traced LLC misses for nine multi-threaded PARSEC applications [7] by executing each of them for $80M$ instructions in gem5 [8] on our baseline architecture (see Sec. 4). The LLC misses are further segregated as *local* and *global* isolated misses and the respective statistics are depicted in Figure 9. This empirical result shows a significant portion (19%-50%) of the total misses are *global* for all nine applications, which is exploited by our FG-DVFS technique.

Figure 10 shows an example of a timeline of the FG-DVFS process. Once a global isolated miss is detected at the LLC directory, the information has to be propagated to the requester core where FG-DVFS can be applied. In this paper, we termed this time-span as propagation delay (t_p), the value of which depends on the distance between the LLC bank (where the request is being handled) and the requester core. We assumed the directory is distributed among the tiles, and traversing between two neighboring tiles has a propagation delay of 1 cycle. Thus, for a 4×4 grid like tiled floorplan (see Figure 13), the worst case propagation delay will be 6 cycles (2ns in 3GHz frequency), where the best case value is 0 cycle. Once the FG-DVFS is activated, the core will reduce its frequency ($f1$ in Figure 10) to the lower level ($f2$) first and voltage regulator (VR) of the respective core will step down the voltage (from $v1$ to $v2$). We have assumed $v1/f1$ and $v2/f2$ as $1.12v/3.0GHz$ and

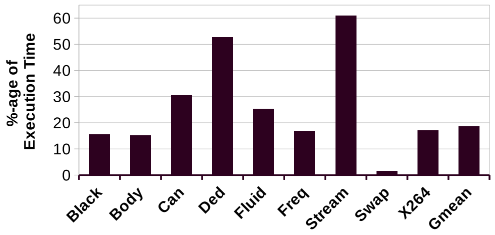


Fig. 11. Stall-share of the execution time (in %-age).

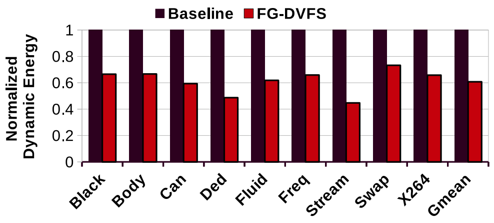


Fig. 12. Core-dynamic energy reduction by FG-DVFS.

$0.76v/1.2GHz$, respectively. In this paper, we also assumed uniform delay of 70ns for all off-chip accesses[3]. This assumption helps *WaFFLe* to initiate the voltage stepping up process early enough, so that the core can start processing instructions at the higher frequency ($3.0GHz$ in our case) once the missed data block will be available. The considered per-core VR has a switching speed of 20mV/ns that takes around 13ns (t_{vr}) to switch between 1.12v and 0.76v. Rest of the time during the stall period, the core will be at lower V/F setting, which is in between 42 to 44ns.

While maintaining higher V/F setting, the core consumes 337nJ during a stall period, which is considered as our baseline. This energy usage can be reduced drastically by applying FG-DVFS during the stall interval. During t_p, core can consume up to 9.3nJ energy depending upon the block's residency on-chip. Additionally, the voltage switching power will be consumed by the VR [50] during t_{vr} and the core will also consume power during the switching process. To compute the total energy usage during t_{vr}, we discretized core power consumption at the precision level of nanosecond for better accuracy. The total switching energy for $2 \times t_{vr}$ will be 44nJ and overall, during t_{mem} the energy usage will be between 86 to 94nJ which is 337.1nJ in case of our baseline. Such significant reduction (around 72% to 74%) in energy usage motivated us to employ FG-DVFS at the cores during the memory stalls. Towards showing the effectiveness of FG-DVFS in overall core-energy reduction, we simulated nine PARSEC applications for $80M$ cycles in gem5, and observe the stall length for all of them. Next, we derived the dynamic energy consumption with and without employing FG-DVFS at the cores during the stall-spans (induced by global isolated misses). Figure 11 and Figure 12 illustrate the percentage of total execution time applications stall due to off-chip accesses and reduction in core-dynamic energy consumption if FG-DVFS is applied, respectively. Overall, FG-DVFS is able to reduce core-dynamic energy by 39%, on average, and up to 55% (for *Stream*). In case of core, dynamic energy shares a significant amount of total energy usage [28], hence, such noticeable reduction in core-dynamic energy can motivate one to employ FG-DVFS towards improving on-chip energy as well as thermal efficiency. Note that, we derived all of these values by simulating the architecture in our simulation setup. The detailed simulation infrastructure and the configuration details are discussed in Sec. 4.

3.2.3 Fine Grained DVFS Algorithm. We present the whole process of applying FG-DVFS at the cores in Algorithm 2. *DVFS_PERIOD* is taken as input to the algorithm, that is the expected average time that DVFS can be applied during an off-chip memory access. On detection of an LLC miss, the algorithm probes the LLC directory to determine if the miss is already handled by some other core(s). If the directory has no entry for the current missed block, a global isolated miss will be detected, and the requester core will be eligible for applying FG-DVFS (line 6 to 8). However, on presence of an entry in the directory, where the block is already in transit between main memory

[3]Our assumption for uniform memory access time is a simplification. But, studying *off-chip memory access latency prediction* is another topic of research, which is out of scope of this paper.

Algorithm 2: Fine-grained DVFS

```
   Input: DVFS_PERIOD
1  dvfs_enabled = 0;
2  apply_dvfs = 0;
3  # Unsigned saturating counter;
4  counter = 0 ;
5  while System is running do
6      if LLC miss is detected for Address then
7          if No entry exists for Address in LLC Directory then
8              apply_dvfs = 1;
9          else
10             # The block is already being handled by an earlier request;
11             # Execute as normal;
12             apply_dvfs = 0;
13     if apply_dvfs == 1 then
14         if dvfs_enabled == 0 then
15             # Apply DVFS to scale down V and F;
16         else
17             # DVFS already applied by a previous request;
18         apply_dvfs = 0;
19         dvfs_enabled = 1;
20         counter = DVFS_PERIOD;
21     if (counter == 0) and (dvfs_enabled == 1) then
22         # Apply DVFS to scale up V and F;
23         dvfs_enabled = 0;
24     counter--;
```

and LLC, requested by other core(s), then the current core will not be allowed to apply FG-DVFS and will proceed execution as normal (line 10 to 12). If the *apply_dvfs* flag is set and the algorithm finds that DVFS has not already been enabled at the requester core, then the core's V/F settings are scaled down (line 13 to 15). A counter (line 20) is used to check when the time-limit for applying DVFS (*DVFS_PERIOD*) is over. Once the counter reaches zero for a DVFS enabled core (line 21), it scales up the V/F setting of the core (line 22) to serve the outstanding instructions at the highest speed on completion of the off-chip memory access.

3.3 WaFFLe: Combined Approach

WaFFLe enhances both temporal and spatial thermal efficiency of contemporary CMPs by combining both core-based and cache-based techniques. Practically, an increased reduction in LLC size will provide greater thermal benefits:

- by generating larger on-chip thermal buffers that can reduce the temperature of adjacent on-chip components, and
- by offering more opportunities for applying FG-DVFS at the cores since the number of LLC misses might increase slightly as an effect of the smaller LLC.

It is hence possible to trade temperature benefits against performance by adjusting the thresholds (i.e., *POWER_UP* and/or *POWER_DOWN* in Algorithm 1). We will illustrate this effect in Sec. 5.5.

Turning off LLC-ways can generate thermal buffers which can occupy a large area on-chip, and maintaining such large thermal buffers for a longer time-span can improve both energy efficiency and thermal benefits. Basically, such a mechanism can maintain a stable thermal status over time. Additionally, employing way-shutdown slightly increases the number of LLC misses, that offers

WaFFLe a larger scope for applying FG-DVFS. Most of the contemporary applications spend a significant portion of its total execution in accessing off-chip memory [7, 37, 51] and a large portion of these off-chip accesses can stall the cores. Fine-grained DVFS efficiently detects and exploits such memory stalls induced by the LLC-misses in order to reduce core energy, which results in noticeable thermal benefits without impacting the performance. In a nutshell, way-shutdown enhances temporal thermal stability by maintaining large thermal buffers, whereas fine-grained DVFS directly targets hotspots generated at the cores, resulting in reduced on-chip spatial-thermal variance.

3.4 Hardware Mechanism

Both of our proposed algorithms (Algorithm 1 and 2) can be implemented separately at the respective controllers. The way-shutdown logic at the LLC controller, that adopts power gating [42] at the way-level granularity of the individual LLC banks, monitors the miss ratio periodically and decides about dynamic cache resizing. Contemporary CMPs are equipped with a set of such performance monitoring counters at the cores as well as at the caches [20], which can be exploited to implement way shutdown logic without any further hardware implementation cost. Moreover, our Algorithm 1 uses conventional control bits of the cache (e.g., valid bit, dirty bits, etc.) that also draws no extra implementation overhead.

On the other hand, FG-DVFS of the cores exploits long memory-stalls. Through selective detection of global isolated LLC misses by inspecting the directory entries, the FG-DVFS mechanism scales down the V/F settings of the individual cores at the beginning of the stall period. Upon a global isolated miss detection, DVFS controller attached to the requester core will be activated. This activation delay is directly proportional to the distance between the requester core and block's (prospective) location on-chip, and thus, in case of home tile, the DVFS activation will experience no delay. The monitoring logic for activating DVFS can be implemented at the LLC controller, whereas each core will be equipped with a voltage regulator, which is a common design choice in modern CMPs. Note that, a counter might be attached to keep track of the viable time-span for which the lower V/F setting is maintained. However, all such logic implementations are trivial and the associated hardware cost for which are limited.

4 METHODOLOGY

We simulated a homogeneous tiled CMP with 16 tiles (see Figure 13) in the gem5 full system simulator [8]. Each tile contains an Alpha 21364 OoO core along with its private L1 caches. A part of the shared L2 cache, called L2-bank, is included in each tile. The whole L2 is physically distributed among the tiles (evenly) but shares a single address space. A 2D-mesh-NoC connects the tiles, hence, each tile is equipped with a router (depicted by the blue colored circles in Figure 13).

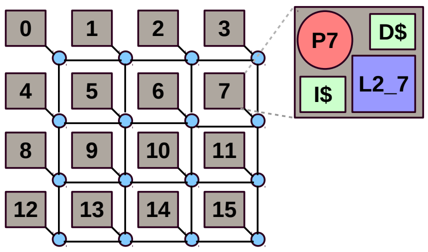


Fig. 13. Homogeneous tiled CMP.

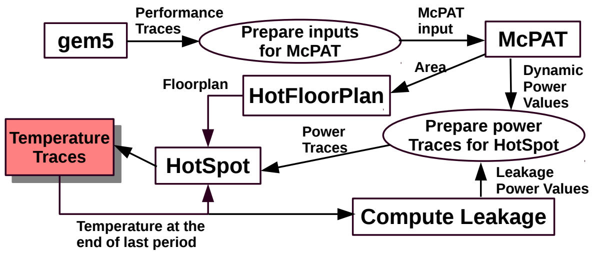


Fig. 14. Simulation framework.

We implemented the *WaFFLe* way-shutdown policy in the Ruby module of gem5 and also implemented FG-DVFS at the cores. For power and temperature estimations, we integrated McPAT [32] and HotSpot 6.0 [52] with gem5, and built a combined performance-power-thermal simulation framework as shown in Figure 14. The periodic *performance traces* are collected from gem5 and are sent to McPAT (*McPAT input*). Based on prior thermal analysis [12, 13], we set the span of this periodic interval at 0.33 μs, during which the temperature across the CMP is assumed stable. The dynamic power consumption is calculated for individual on-chip components by executing McPAT. The leakage power estimation in McPAT assumes uniform on-chip temperature. To overcome this limitation we compute component-wise leakage power by considering temperatures of individual on-chip components at the end of the last period [21–23]. Finally, the total power consumption is derived from dynamic and leakage power estimations, and are sent to HotSpot to generate the temperature traces. The HotFloorPlan module generates floorplan of the CMP once at the beginning by considering the component-wise area estimation from McPAT. We consider 32nm technology nodes for estimating power and area details.

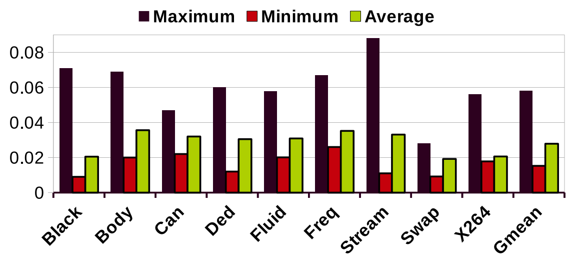


Fig. 15. Range of cache miss ratio (*ratio* in Algorithm 1).

Table 1. System parameters

Parameters	Values	Parameters	Values
ISA	Alpha 21364	L1-I	64 KB, 4-way
Exec. units	2 int/br., 1 mul, 1 fp, 1 ld/st	L1-D	64 KB, 4-way, 3 cycles, 8 MSHRs
Max. V/F	1.12V, 3.0GHz	L2	1MB, 16-way
Min. V/F	0.76V, 1.2GHz	L2 Latency	12 cycles
ROB Size	200	Cache	LRU, 64 byte blocks
Issue width	8	Cache model	SNUCA
Phys. Reg	80/72	DRAM latency	70 ns
#Threads per core	1	Technology	32 nm
Tiles	16	Ambient Temp.	47°C

The interval length used for Algorithm 1 is set to 2M clock cycles and is based on a prior empirical cache locality analysis [33]. We observe the range of the *ratio* for nine PARSEC applications over 80M clock cycles (from the RoI part of the execution) to set $POWER\_UP$ and $POWER\_DOWN$ in Algorithm 1. Figure 15 shows, for individual applications the range of cache miss ratio varies from 1% to 9% with an average of 2.75%. The observed small difference between the smallest and the average values implies that for most intervals the miss ratio is small. For our evaluation, we set the values for $POWER\_UP$ and $POWER\_DOWN$ as 0.04 and 0.025, respectively, i.e., a bank miss ratio of more than 0.04 will power up a cache-way while a miss rate of less than 0.025 will power down a cache-way in the bank.

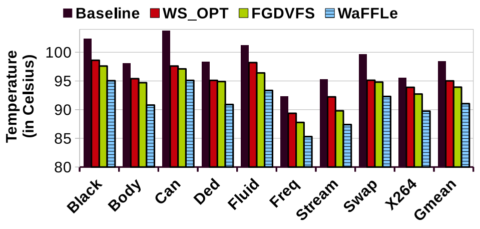


Fig. 16. Effects on peak temperature.

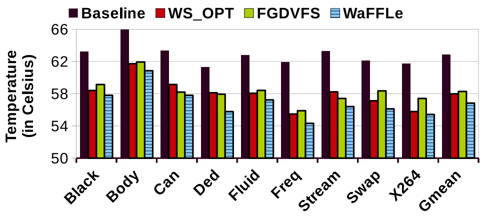


Fig. 17. Effects on average temperature.

We use the multi-threaded PARSEC benchmark suite [7] (with large input sets) to evaluate the proposed architecture. We use 16 threads for each application, where each core executes one thread at a time. For individual PARSEC applications, we collected the results/traces from the RoI portion of the simulation. Note that, for each of our simulations, the baseline configuration (see Table 1) details are specified once at the beginning of the execution. While evaluating *WaFFLe*, LLC is resized over the baseline configuration on-the-fly as per Algorithm 1, which is implemented in the Ruby module of gem5 (as mentioned earlier). We have assumed the CMP is executing the highest possible workload, and hence, all cores are always active. Table 1 contains the configuration parameters for the processor cores and memories used in the evaluations. The employed on-chip VR has a switching speed of 20 mV/ns [16]. We have based the power consumption overheads for each of the on-chip VR on the regulator-power model proposed by Kim W. et al. [50] with a switching activity factor of 0.5. Each of the regulators attached with our OoO cores consumes 0.106W and 0.168W for 0.76 V and 1.12 V outputs, respectively. These switching power overheads are marginal with respect to the run-time power consumption of cores [32]. Note that, during down scaling, the voltage will be scaled down after reducing the frequency, whereas for upscaling the voltage has to be ramped up before increasing the frequency. For each voltage level, the corresponding frequency represents the highest possible attainable value (see Table 1). In our evaluation, we use only the highest and the lowest V/F settings, but *WaFFLe* is also capable of supporting multiple V/F settings.

5 EVALUATION

5.1 Changes in On-chip Thermal Status

Towards analyzing the benefits gained by our cache-based and core-based policies, we applied all of them individually in the following manner: *WS_OPT* (see Algorithm 1) will be applied at 8 MB LLC (L2 cache), *FGDVFS* (see Algorithm 2), and finally, our combined approach, *WaFFLe*. Figure 16 and Figure 17 show the maximum changes in peak and average temperatures of the CMP, respectively, for all nine PARSEC applications. *WaFFLe* shows, for all applications, the best thermal efficiency for both peak and average chip temperature. For *Black*, *Fluid* and *Stream* the reductions are significant for all policies. Due to higher computational loads, the peak temperature is high enough for *Black* and *Fluid*. Hence, despite shutting down numerous LLC-ways in *WS_OPT* the reduction in peak temperature is comparatively lesser than the DVFS based one for both of these benchmarks. But, the large number of gated ways significantly reduces average temperature for *Black* and *Fluid* over the core-based ones. The memory-intensive application like *Stream*, core temperature reductions by both *WS_OPT* and DVFS-based policies are lesser than *Black*. The presence of a massive amount of temporal data increases cache accesses for *Stream* but offers enough scope to shut down many cache-ways even from the heavily used LLC-banks, and thus it outperforms prior bank-based policies [12, 13]. Out of all applications, *X264* has the least reduction for all cases due to fluctuations

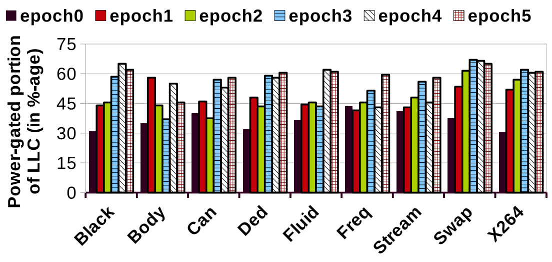
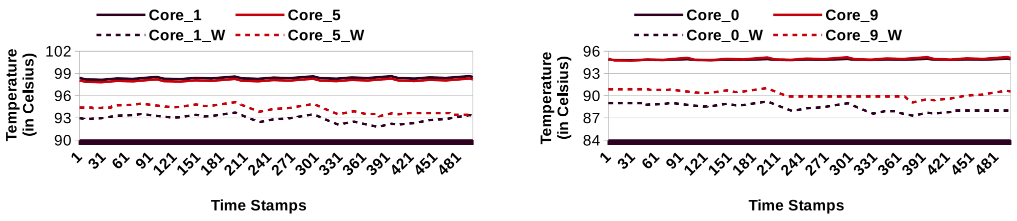


Fig. 18. Temporal Changes in LLC-size.



(a) *Body* (b) *Stream*

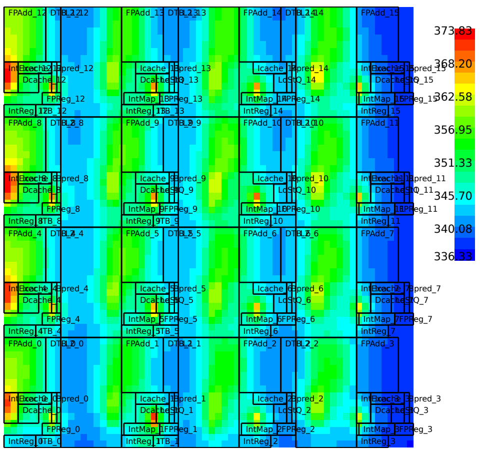
Fig. 19. Temporal variation in Temperature at the cores.

in cache workloads across the execution phases with heavy computational overheads. The overall maximum reduction in peak temperature varies between 6.5°C and 8.4°C while applying *WaFFLe*.

Figure 17 shows that, for computational workloads (like *Black* and *Swap*), the average chip temperature is reduced more in the case of *WS_OPT*, as a larger number of cache-ways are kept gated for longer time-intervals, without much degradation in performance, as seen in Figure 18. Such large number of gated LLC ways reduces the temperature of larger on-chip areas and generates thermal buffers that assist in reducing temperature of their adjacent cores due to significant amount of heat transfer. However, for memory-intensive workloads (like *Stream* and *Freq*), our way-shutdown strategy was unable to maintain a reduced cache size for long time-intervals, which results in almost a similar reduction in average chip temperature as its core-based counterparts. Figure 18 further depicts, for computational workloads (like *Black* and *Swap*) *WaFFLe* was able to maintain a lower LLC size for 3-4 epochs[4], that leads to more reduction in average chip temperature. However, by combining both FG-DVFS and LLC way-shutdown, *WaFFLe* achieves a maximum reduction in average chip temperature ranging between 5.0°C and 6.2°C.

Temporal and Spatial Thermal Effects: Figure 19 shows the temporal changes in temperature for two OoO cores each for *Body* (mixed) and *Stream* (memory-intensive), where cores are considered one from the edge of the chip and one from the central location. In this figure, the *Core_i* represents the temporal change in temperature of the core ID *i* in baseline (the solid lines in the figure) and *Core_i_W* implies the temporal change in temperature for the same core (the dotted lines in the figure), while applying *WaFFLe*. We have taken the statistics over 500 time-stamps with an interval of 0.33μs. For *Body*, the core experienced a moderate amount of isolated misses, but the heavy cache requirement limits the reduction in core-temperature. But, many adjacent cache-ways were gated that significantly reduces the core temperature, at *Core*_1. For *Stream*, the cores experience a lot of

[4]The epochs are contiguous within the RoI portions of the individual simulations.


(a) *Baseline*

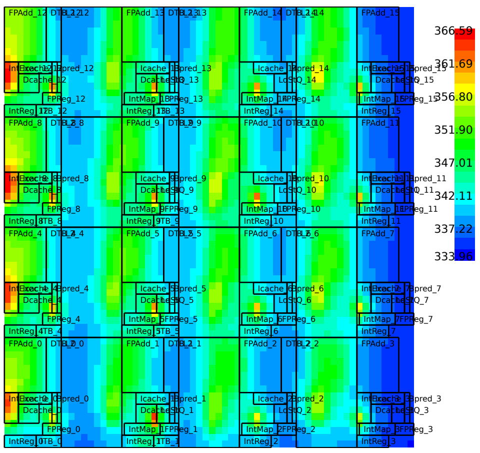

(b) *WaFFLe*

Fig. 20. Spatial variation in Temperature (for *Black*).

Table 2. Maximum reduction in spatial thermal variance.

Applications	Reduction (in °C)	Applications	Reduction (in °C)
Black	5.6	**Freq**	3.6
Body	4.3	**Stream**	4.6
Can	3.9	**Swap**	5.9
Ded	4.0	**X264**	4.7
Fluid	4.3	**Gmean**	4.5

isolated misses, and LLC contains a large amount of temporal data. Hence, for a long time-span, a large set of ways were gated that reduces core temperature for both *Core*_0 and *Core*_9, remarkably.

Figure 20 shows spatial thermal variation for *Body* at a particular time-stamp. Cache banks at the edges are the coolest on-chip regions, both in *baseline* and *WaFFLe*. For *WaFFLe*, the reduction in temperature across the chip is prominent, and the peak temperature is reduced by 7°C. Shutting down of inner cache-ways of the chip drastically reduces the temperature at the cores. Furthermore, by applying FG-DVFS during the stall-intervals, *WaFFLe* reduces the effective temperature at the cores, significantly. To show the effectiveness of *WaFFLe* in managing spatial thermal imbalance on-chip, we further report the maximum changes in spatial thermal variance (the difference between the highest and the lowest chip-temperature) across the chip for nine PARSEC applications. Table 2 shows the respective reductions for the individual benchmarks over baseline, where both memory intensive and compute intensive applications experience a reduction within a range of $4-6$°C with an average of 4.5 °C.

5.2 Change in Energy Usage

In our simulation setup, we consider the temperature values of the individual on-chip components at different time-stamps to calculate their respective leakage power consumption by adopting piece-wise linear approximation [23]. Figure 21 shows the reduction in leakage consumption as a result of the reduction in temperature by *WaFFLe*. Unlike the cache-ways, cores are not shutdown completely, hence, the leakage reduction is lesser than for the caches, but is noticeably high. Compute-bound

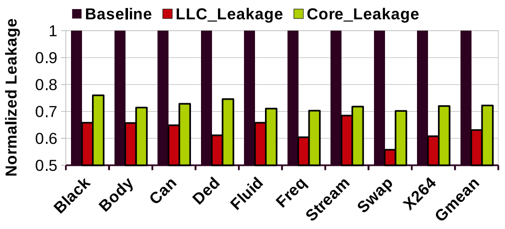


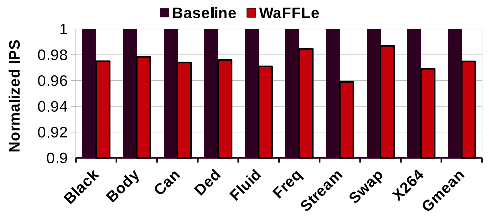
Fig. 21. Leakage Energy of LLC and Cores.

applications were able to maintain smaller LLC-sizes for a longer time (like *Black* and *Swap*) and show more leakage reduction at the LLC than at the cores. The leakage reductions at cores and caches are close to each other for *Stream*, a memory-intensive application, that allows *WaFFLe* to apply FG-DVFS during its large number of off-chip accesses. Additionally, the presence of a large amount of temporal data (during some execution-phases) further assists *WaFFLe* to gate a significant number of LLC-ways. For mixed and memory-intensive applications (like *Body* and *Fluid*), smaller LLC-size was not maintained for a long time-span due to performance constraints, hence, reduction in LLC-leakage is close to the core-leakage reduction. Overall, *WaFFLe* managed to reduce core leakage energy by 28% (mean) along with 38% (mean) reduction in LLC leakage for nine PARSEC applications.

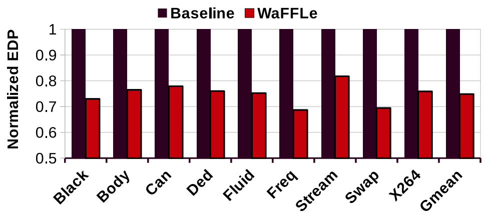
5.3 Change in Performance and EDP

As FG-DVFS in *WaFFLe* is applied at memory stalls, it hardly incurs any performance overhead. Additionally, on-chip VRs incorporate smaller voltage switching overhead than its off-chip counterparts. The performance cognizant way-shutdown incurs small overhead during sorting out of clean data inside the individual cache sets and swapping operations. Extra stall cycles can also be incurred due to a reduction in cache associativity that increases conflict misses. Our simulation framework considers all of these issues and respective changes in performance for all the nine PARSEC applications are shown in Figure 22a. The overall average performance aggravation is less than 3% for eight benchmarks, but for *Stream*, performance degradation is higher due to its diverse cache requirements across the different execution-phases. Note that, for performance, we consider IPS (Instructions Per Second) instead of IPC (Instructions Per Cycle), due to changes in core-frequency.

We consider component-wise temperature values (over time) while computing leakage power of the individual on-chip components. For all applications, reduction in core leakage is more than 22%, whereas cache leakage reduction is close to 38%, on average. The respective reductions in EDP for our applications are shown in Figure 22b. Note that, the component-wise total power consumption includes cores, caches and NoC (links and routers) towards calculating EDP. However, the more performance degradation along with lesser energy savings in *Stream* results in slightly higher EDP. For *Freq* and *Swap*, EDP gains are more due to lesser performance degradation with more savings in leakage. Note that, more off-chip accesses in case of memory-intensive workloads increases the chances of applying FG-DVFS at the cores, that further reduces core-temperature, hence, the respective leakage consumption. On an average, *WaFFLe* achieves an EDP gain of 25.2% with a sound reduction in chip temperature.

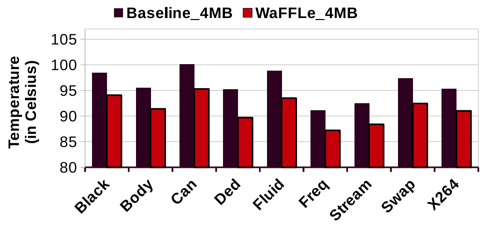


(a) Changes in IPS with respect to Baseline.

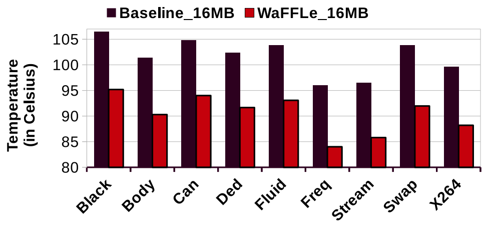


(b) Reduction in EDP with respect to Baseline.

Fig. 22. Changes in Performance and EDP.



(a) LLC-size: 4MB

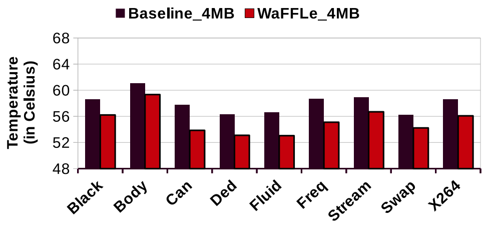


(b) LLC-size: 16MB

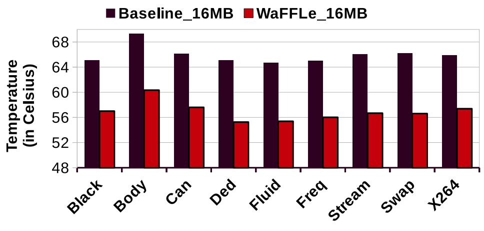
Fig. 23. Effects on Peak Temperature for different LLC-sizes.

5.4 WaFFLe with Different LLC-sizes

We further studied the effects of *WaFFLe* on smaller (4MB) and larger (16MB) LLCs, without changing the other parameters. A smaller LLC provides fewer chances for dynamic changes in cache sizes to maintain performance, hence, smaller cache portions are turned off, which generates smaller thermal buffers. But a smaller LLC increases the number of LLC misses, which results in an increase in memory stalls and the opportunities for performing FG-DVFS. *WaFFLe* reduces peak temperature within a range from 4 to 5.5°C (see Figure 23a). The reduction in peak temperature is increased with a larger LLC, as it provides more chances to maintain a smaller sized cache for long periods of time. The larger thermal buffers in case of a larger LLC assist in reducing peak temperatures. The overall reduction in peak temperature lies in the range of 10.5 – 12°C (see Figure 23b).



(a) LLC-size: 4MB



(b) LLC-size: 16MB

Fig. 24. Effects on Avg. Temperature for different LLC-sizes.

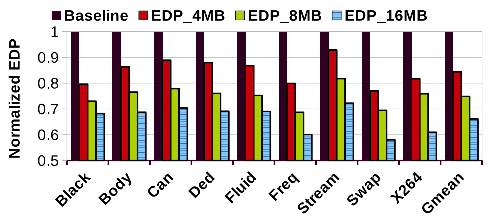


Fig. 25. EDP gains with different LLC sizes.

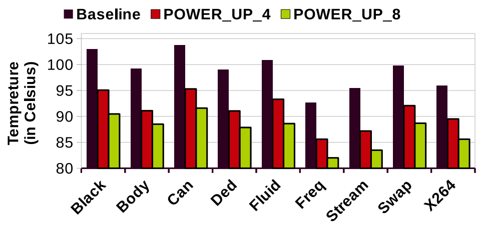


Fig. 26. Peak temperature: *POWER_UP* = 4% vs. *POWER_UP* = 8%.

Figure 24a and Figure 24b show the reductions in average chip temperature for the individual applications, with smaller and larger LLCs, respectively. For smaller LLC-size of 4MB, reduction in average chip temperature lies between 1.8 and 4°C, whereas the range is 8.0 – 9.9°C for larger caches. Overall leakage reduction is more with *WaFFLe* in case of larger LLC and trivially lesser with smaller LLC. Hence, EDP gains are more with the larger LLC over its smaller counterparts. Figure 25 shows the reductions in EDP for different LLC sizes across the applications.

5.5 WaFFLe: Synergy effects

We further studied the synergy effects of both the core and cache based policies by using a larger value of *POWER_UP* (in Algorithm 1). This larger value of *POWER_UP* will actually keep more LLC ways turned off for a longer time-span, hence more reduction in chip temperature than its smaller counterpart. Additionally, maintaining a reduced LLC size for a longer time-span drastically increases the LLC misses that offers more chances for Algorithm 2 to apply FG-DVFS at the cores at the cost of higher performance degradation. Figure 26 and Figure 27 show the reduction in peak and average chip temperature, respectively, for *POWER_UP* = 8% (*POWER_UP*_8) compared to the baseline and *POWER_UP* = 4% (*POWER_UP*_4). The reduction in peak temperature is greater in case of *POWER_UP*_8 as a larger number of LLC ways were kept turned off than for *POWER_UP*_4, that provides more opportunities for applying FG-DVFS by incurring increased memory stalls. The reduction in average temperature is also greater for *POWER_UP*_8 than *POWER_UP*_4 due to the longer duration for which a larger gated LLC portion is maintained. However, the increased LLC miss-count for *POWER_UP*_8 drastically aggravates overall performance, which is illustrated in Figure 28. As expected, the aggravation is more (up to 9.3%) in case of the memory intensive ones (e.g. *Body, Can, Stream*), whereas compute-intensive workloads (e.g. *Black, Swap*) experience comparatively lower aggravation (between 3 – 4%). The overall average performance aggravation is around 5.1% for *POWER_UP*_8, which is 2.5% in *POWER_UP*_4. Hence, we conclude that, with higher values of *POWER_UP*, *WaFFLe* can enhance thermal benefits by keeping many LLC ways[5] gated for a longer time, but this can severely degrade the overall performance especially for the memory intensive workloads.

5.6 Comparison Against Prior Work

In a recent exploration [27], authors proposed a thermal management technique for a 16-core based homogeneous CMP by combining DVFS and task migration. While trying to maintain the core temperature within a preset threshold, this technique applies per cores DVFS along with migrating tasks from hotter to colder regions where colder but faster cores are prioritized. But, effectiveness

[5]without violating Way_{Limit} in Algorithm 1

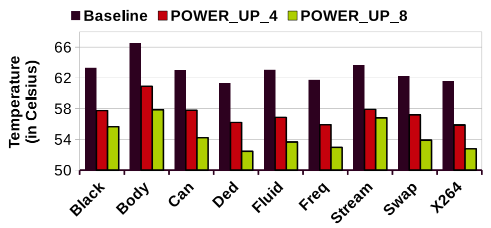


Fig. 27. Average temperature: *POWER_UP* = 4% vs. *POWER_UP* = 8%.

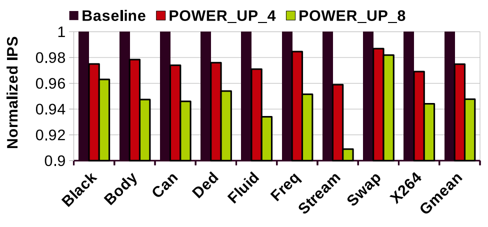
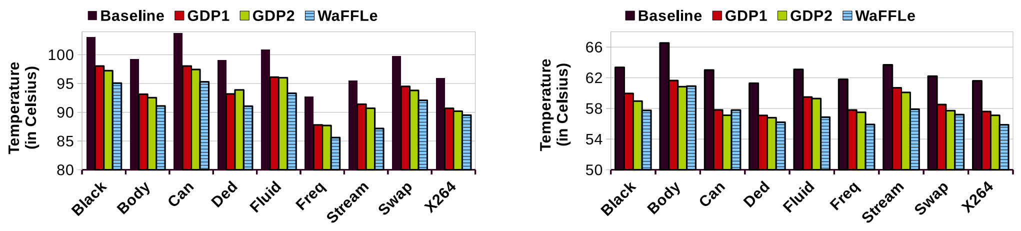


Fig. 28. Performance (IPS): *POWER_UP* = 4% vs. *POWER_UP* = 8%.



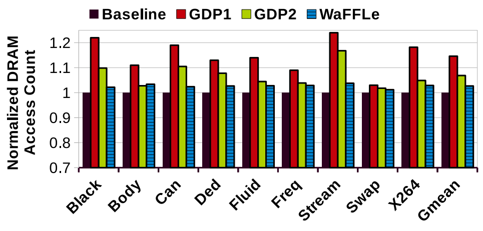
(a) Peak Temperature (b) Average Temperature

Fig. 29. Temperature comparison with prior policies.

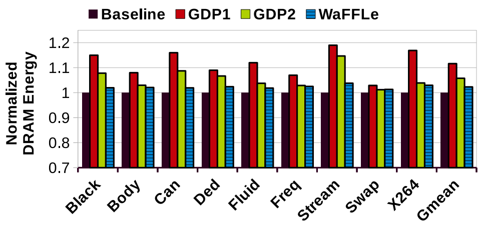
of this task migration is limited by the number of available spare cores, and temporal thermal status of the CMP was not discussed, contrary to *WaFFLe*.

We compared *WaFFLe* against two recent prior thermal management techniques (*GDP1* and *GDP2*) [12], where authors shutdown cache banks for temperature reduction along with a greedy DVFS policy [36]. In *GDP1*, cache banks are turned off at the hottest regions, whereas *GDP2* targets underutilized cache banks to generate on-chip thermal buffers. Both policies individually applied greedy DVFS, which reduces V/F settings of the cores if the local temperature reaches a threshold value. Figure 29a shows that *WaFFLe* outperforms these prior policies in terms of peak temperature reduction. But turning off a full LLC bank generates larger thermal buffers, hence, reduction in average temperature is almost similar to our policy in cases of some applications like, *Body*, *Can* and *Ded* (see Figure 29b). Unlike *WaFFLe*, *GDP1* and *GDP2* use system-wide IPC for performance monitoring, which may lack some scopes for cache resizing, as cache performance is not the only parameter that causes IPC-change. Additionally, during the bank shutdown process, both *GDP1* and *GDP2* suspend servicing of the cache requests, that further aggravates performance. In fact, invalidation of CN blocks in our Algorithm 1 also speeds up the LLC resizing process. Thus, frequent switching of core-frequency along with LLC-bank shutdown in both *GDP1* and *GDP2* show more degradation in performance than *WaFFLe*.

All of these techniques, i.e., *WaFFLe*, *GDP1* and *GDP2*, improve on-chip thermal efficiency by amalgamating core and cache based strategies, but cache resizing in all cases might incorporate side effects. Such effects primarily include the change in DRAM access-count, so the DRAM energy, caused by an increase in the LLC misses. For *GDP1* and *GDP2*, shutting down of LLC banks incurs extra write-back in addition with LLC miss count that increases DRAM accesses, hence, the energy usage. On the other hand, *WaFFLe* attempts to invalidate CN blocks during resizing that significantly trims extra write-back operations. We observed the changes in DRAM access counts and energy,



(a) DRAM Access Count



(b) DRAM Access Energy

Fig. 30. Change in DRAM access count and access energy.

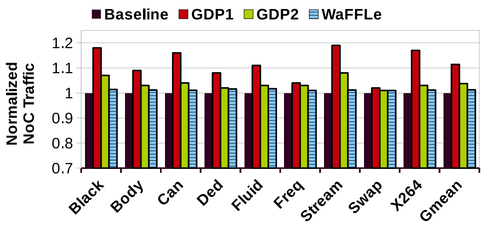


Fig. 31. Change in NoC Traffic.

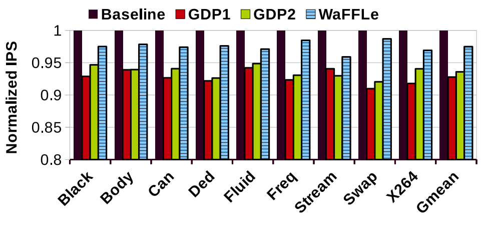


Fig. 32. Performance comparison with prior policies.

and report the same in Figure 30a and 30b. For memory intensive applications (e.g. *Stream*), the DRAM access counts as well as energy are increased due to additional misses caused by LLC resizing. However, for all of these applications, *WaFFLe* outperforms the prior techniques in terms of DRAM access counts and energy, and the improvement is significantly noticeable, especially in cases of memory intensive applications.

Another side effect of dynamic LLC resizing includes the changes in NoC traffic. The bank shutdown process involves transferring of the blocks to a target bank in addition with the remapping of the future requests that further increases NoC traffic. Figure 31 shows the changes in NoC traffic in case of *WaFFLe*, *GDP1* and *GDP2*, where NoC traffic significantly increases in case of *GDP1* as bank shutdown is not cognizant to the recent bank usage. In *GDP2*, usage cognizant bank shutdown does not increase NoC traffic like *GDP1*, as turning off least used bank will have lesser number of remapped request later. However, *WaFFLe* does not have any remapped requests, and hence, is free from such overheads. By considering all such costs, we analyzed the overall performance of these three policies. Figure 32 shows the effects on performance while applying *GDP1*, *GDP2*, and *WaFFLe*. For all nine PARSEC applications, performance degradation in case of *WaFFLe* is remarkably lower than both *GDP1* and *GDP2*. We summarize the comparison with the prior techniques in Table 3. The values clearly indicate that *WaFFLe* reduces both peak and average chip temperature more than the prior policies for our baseline 16-core based homogeneous CMP.

6 RELATED WORK

Most of the prior thermal management policies [15, 28] throttle the dynamic power of the cores in CMPs either by employing DVFS [43] or through task migration [14, 18, 19]. Donald and Martonosi [15] have classified the thermal management techniques into multiple groups based on their implementation strategies. These papers have shown the efficacy of several DVFS and task migration based policies in controlling temperature, where distributed (per-core) DVFS along with

Table 3. Summary of comparison with prior techniques by considering an 8MB LLC.

LLC-Size	GDP1	GDP2	WaFFLe
Reduction in Peak Temperature	6.1°C	6.7°C	8.4°C
Reduction in Average Temperature	5.2°C	5.9°C	6.2°C
IPS degradation	7.3%	6.4%	2.5%

task migration is claimed to be the best. But, underlying migration overheads at the caches were not considered, which might impose scalability issues in the recent large sized CMPs. V. Hanumaiah et al. [23] proposed a thermal efficient thread migration policy, which has also been integrated with DVFS to achieve active scalable cooling of homogeneous CMPs [22]. Moreover, combining DVFS and per-core power gating can enhance system throughput and reduce the temperature of large CMPs [30]. However, most of these techniques handled hotspots at some localized regions without considering spatio-temporal on-chip thermal variation. Furthermore, thermal management units (TMUs) have been implemented [1] to reduce the negative impacts of on-chip hotspots, but their limitations lead to significant performance losses [40].

Prior cache-based policies predominantly attempted to minimize the leakage energy [11, 25, 38, 51], without an extensive focus on the on-chip thermal status. But a few techniques such as the power density-minimized architecture [29] and shadow tag [48, 49] reduce on-chip power density through leakage minimization, which also decreases the temperature of the CMP. Other explorations [5, 17, 39] orchestrated cache block placements and their accesses to enhance the thermal efficiency of caches. In another work, FlexiWay [38], Mittal et al. effectively saved LLC leakage by turning off different number of cache-ways from different sets. FlexiWay might have scalability issues from implementation perspective especially in case of larger caches and the thermal aspect of the cache was also not considered. As on-chip power density escalates for smaller technology nodes, modern CMPs (built in 32 nm or smaller) need a combined thermal management policy that accounts for both cores and caches to reduce the spatio-temporal on-chip thermal imbalance, the primary challenge in designing state-of-the-art CMPs [24]. A few recent thermal management techniques [2, 4] considered both caches and cores, but these learning-based policies targeted a specific set of small-scale embedded systems.

***WaFFLe* over State-of-the-art.** In *WaFFLe*, our proposed performance cognizant cache-way shutdown mechanism:

- attempts to invalidate the CN blocks present in the LLC and swaps the data blocks on demand during the resizing process, thus limits additional write back to the main memory;
- keeps the normal cache operations during the resizing process (unlike the prior bank shutdown based thermal management approaches [12, 13]), that reduces performance degradation;
- is able to monitor the locality of reference in a more detailed manner, that enables more efficient LLC resizing, compared to the prior bank-level strategies.

On the other hand, LLC stall induced FG-DVFS (at the cores) of *WaFFLe* contributes in the following ways over the existing state of the art [16]:

- *WaFFLe* studies the potential of FG-DVFS in improving thermal efficiency;
- the implementation of FG-DVFS in multi-core environment by exploiting the coherence information about current misses in the LLC.

To the best of our knowledge, *WaFFLe* is the first power/thermal management mechanism proposed for CMPs that exploits coherence information in detecting apt memory stall cycles to apply FG-DVFS at the cores. With comprehensive temporal and spatial thermal analyses, we empirically validated the potential of *WaFFLe*, that integrates dynamic cache-way shutdown and LLC stall induced FG-DVFS at the cores, in improving thermal efficiency of the CMP with a slight average performance degradation of 2.5%.

7 CONCLUSIONS

The thermal management of modern CMPs has become a topic of paramount importance in recent chip design. Managing local hotspots and reducing average chip temperature are the primary objectives of thermal management techniques. This paper presents *WaFFLe*, an integrated thermal management approach that applies fine-grained DVFS at the cores to handle local hotspots and generates thermal buffers at the LLC by incorporating a performance cognizant way-shutdown policy to reduce the average chip temperature. The fine-grained DVFS is applied to the cores during long-latency memory accesses, which might otherwise stall the execution. To reduce write-backs of cache blocks, *WaFFLe* implements a way-shutdown technique that mostly invalidates clean blocks. *WaFFLe* outperforms a previous state-of-the-art cache-based technique and a greedy DVFS policy [12, 36] in terms of both thermal efficiency and performance.

WaFFLe reduces both peak and average temperature of a 16-core homogeneous CMP with up to 8.4 °C and 6.2 °C, respectively, with an average performance degradation of only 2.5 %. *WaFFLe* also achieves a maximum reduction in spatial thermal variance of 5.9 °C. To the best of our knowledge, this is the first thermal management approach for generic CMPs that considers both caches and cores to reduce chip temperature.

ACKNOWLEDGMENTS

This work was funded by the *European Research Consortium for Informatics and Mathematics (ERCIM)* and *Marie Curie Individual Fellowship (MSCA-IF), EU (Grant Number 898296)*. We are thankful to the anonymous reviewers for their constructive feedback and suggestions that helped us in improving the quality of the manuscript.

REFERENCES

[1] 2015. Intel Xeon Phi Coprocessor datasheet.
[2] T. Adegbija and A. Gordon-Ross. 2018. TaPT: Temperature-aware dynamic cache optimization for embedded systems. *Computers* 7, 1 (2018), 3.
[3] M. Alian, A. H. M. O. Abulila, L. Jindal, D. Kim, and N. S. Kim. 2017. NCAP: Network-Driven, Packet Context-Aware Power Management for Client-Server Architecture. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.
[4] M. H. Alsafrjalani and T Adegbija. 2018. TaSaT: thermal-aware scheduling and tuning algorithm for heterogeneous and configurable embedded systems. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI (GLSVLSI)*. 75–80.
[5] 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 (GLSVLSI)*. 365–368.
[6] G. Bhat, S. Gumussoy, and U. Y. Ogras. 2017. Power-temperature stability and safety analysis for multiprocessor systems. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 1–19.
[7] 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)*. 72–81.
[8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. 2011. The gem5 simulator. *ACM SIGARCH computer architecture news* 39, 2 (2011), 1–7.
[9] E. A. Burton, G. Schrom, F. Paillet, J. Douglas, W. J. Lambert, K. Radhakrishnan, and M. J. Hill. 2014. FIVR — Fully integrated voltage regulators on 4th generation Intel® Core™ SoCs. In *2014 IEEE Applied Power Electronics Conference*

and Exposition - APEC 2014. 432–439.

[10] S. Chakraborty and H. K. Kapoor. 2016. Static energy reduction by performance linked dynamic cache resizing. In *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. 1–6.

[11] S. Chakraborty and H. K. Kapoor. 2017. Performance linked dynamic cache tuning: A static energy reduction approach in tiled CMPs. *Microprocessors and Microsystems* 52 (2017), 221 – 235.

[12] S. Chakraborty and H. K. Kapoor. 2018. Analysing the Role of Last Level Caches in Controlling Chip Temperature. *IEEE Transactions on Sustainable Computing* 3, 4 (2018).

[13] S. Chakraborty and H. K. Kapoor. 2019. Exploring the role of large centralised caches in thermal efficient chip design. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 24, 5 (2019), 1–28.

[14] T. Chantem, R. P. Dick, and X. S. Hu. 2008. Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In *2008 Design, Automation and Test in Europe (DATE)*. 288–293.

[15] J. Donald and M. Martonosi. 2006. Techniques for Multicore Thermal Management: Classification and New Exploration. In *33rd International Symposium on Computer Architecture (ISCA)*.

[16] S. Eyerman and L. Eeckhout. 2011. Fine-Grained DVFS Using on-Chip Regulators. *ACM Trans. Archit. Code Optim. (TACO)* 8, 1 (2011).

[17] 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 (MEDEA)*.

[18] Y. Ge, P. Malani, and Q. Qiu. 2010. Distributed task migration for thermal management in many-core systems. In *Proceedings of the 47th Design Automation Conference (DAC)*. 579–584.

[19] Y. Ge, Q. Qiu, and Q. Wu. 2011. A multi-agent framework for thermal aware task migration in many-core systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20, 10 (2011), 1758–1771.

[20] B. Goel, S. A. McKee, and M. Själander. 2012. Chapter two - Techniques to Measure, Model, and Manage Power. Advances in Computers, Vol. 87. Elsevier, 7 – 54.

[21] V. Hanumaiah and S. Vrudhula. 2014. Energy-Efficient Operation of Multicore Processors by DVFS, Task Migration, and Active Cooling. *IEEE Trans. Comput.* 63, 2 (2014), 349–360.

[22] 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*.

[23] 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 (2011).

[24] J. Henkel, H. Khdr, and M. Rapp. 2019. Smart Thermal Management for Heterogeneous Multicores. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*.

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

[26] S. Khatamifard, L. Wang, W. Yu, S. Kose, and U. R. Karpuzcu. 2017. ThermoGater: Thermally-aware on-chip voltage regulation. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. 120–132.

[27] Y. G. Kim, M. Kim, J. Kong, and S. W. Chung. 2020. An Adaptive Thermal Management Framework for Heterogeneous Multi-Core Processors. *IEEE Trans. Comput.* 69, 6 (2020), 894–906.

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

[29] Ja Chun Ku, S. Ozdemir, G. Memik, and Y. Ismail. 2005. Thermal management of on-chip caches through power density minimization. In *38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'05)*.

[30] J. Lee and N. 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, 02 (2012), 225–235.

[31] D. Li and J. Wu. 2014. Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms. *IEEE Transactions on Parallel and Distributed Systems* 26, 3 (2014), 810–823.

[32] 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 *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 469–480.

[33] A. Mandke, B. Amrutur, and Y. N. Srikant. 2011. Adaptive Power Optimization of On-chip SNUCA Cache on Tiled Chip Multicore Architecture Using Remap Policy. In *Second Workshop on Architecture and Multi-Core Applications (WAMCA)*.

[34] M. Manivannan, V. Papaefstathiou, M. Pericas, and P. Stenstrom. 2016. RADAR: Runtime-assisted dead region management for last-level caches. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 644–656.

[35] M. Manivannan, M. Pericàs, V. Papaefstathiou, and P. Stenström. 2017. Runtime-Assisted Global Cache Management for Task-Based Parallel Programs. *IEEE Computer Architecture Letters* 16, 2 (2017), 145–148.

[36] 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 (2015), 1017–1030.

[37] S. Mittal. 2014. A survey of architectural techniques for improving cache power efficiency. *Sustainable Computing: Informatics and Systems* (2014).

[38] S. Mittal, Z. Zhang, and J. S. Vetter. 2013. FlexiWay: A cache energy saving technique using fine-grained cache reconfiguration. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*. 100–107.

[39] H. Noori, M. Goudarzi, and K. Inoue, K.and Murakami. 2008. Improving energy efficiency of configurable caches via temperature-aware configuration selection. In *2008 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 363–368.

[40] S. Pagani, H. Khdr, W. Munawar, J. Chen, M. Shafique, M. Li, and J. Henkel. 2014. TSP: Thermal Safe Power - Efficient power budgeting for many-core systems in dark silicon. In *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*.

[41] P. Petrides and P. Trancoso. 2017. Heterogeneous- and NUMA-aware Scheduling for Many-core Architectures. In *Proceedings of the 10th ACM International Systems and Storage Conference (SYSTOR)*.

[42] M. Powell, Se-Hyun Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. 2000. Gated-Vdd: a circuit technique to reduce leakage in deep-submicron cache memories. In *ISLPED'00: Proceedings of the 2000 International Symposium on Low Power Electronics and Design*. 90–95.

[43] R. Rao, S. Vrudhula, C. Chakrabarti, and N. Chang. 2006. An Optimal Analytical Solution for Processor Speed Control with Thermal Constraints. In *ISLPED'06 Proceedings of the 2006 International Symposium on Low Power Electronics and Design*. 292–297.

[44] S. Saha, Y. Lu, and J. S. Deogun. 2012. Thermal-Constrained Energy-Aware Partitioning for Heterogeneous Multi-core Multiprocessor Real-Time Systems. In *Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. 41–50.

[45] S. Sharifi, D. Krishnaswamy, and T. S. Rosing. 2013. PROMETHEUS: A Proactive Method for Thermal Management of Heterogeneous MPSoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 7 (2013), 1110–1123.

[46] M. Själander, M. Martonosi, and S. Kaxiras. 2014. *Power-Efficient Computer Architectures: Recent Advances*. Synthesis Lectures on Computer Architecture.

[47] K. Stavrou and P. Trancoso. 2005. TSIC: Thermal Scheduling Simulator for Chip Multiprocessors. In *Proceedings of the Panhellenic Conference on Advances in Informatics*, Vol. 3746. Springer-Verlag.

[48] 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 the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*.

[49] 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 (2012).

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

[51] W. Zang and A. Gordon-Ross. 2013. A Survey on Cache Tuning from a Power/Energy Perspective. *ACM Comput. Surv.* 45, 3 (2013).

[52] 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*.

[53] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma. 2016. Thermal-Aware Task Scheduling for Energy Minimization in Heterogeneous Real-Time MPSoC Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 8 (2016), 1269–1282.