# Variability-Aware Dark Silicon Management in On-Chip Many-Core Systems

Muhammad Shafique[1], Dennis Gnad[1], Siddharth Garg[2], Jörg Henkel[1]
[1]Chair for Embedded Systems, Karlsruhe Institute of Technology, Germany
[2]Electrical and Computer Engineering, New York University, NY, USA
Corresponding Authors: muhammad.shafique@kit.edu; sg175@nyu.edu

***Abstract*—Dark Silicon refers to the constraint that only a fraction of on-chip resources (cores) can be simultaneously powered-on (running at full performance) in order to stay within the allowable power budget and safe temperature limits, while others remain 'dark'. In this paper, we demonstrate how these 'dark cores' can be leveraged to improve the temperature profile at run-time, thus providing opportunities to power-on more cores at the nominal voltage than the number allowed when strictly obeying the conventional Thermal Design Power ($TDP$) constraint. In this paper, we propose a computationally efficient dark silicon management technique that determines the best set of cores to keep dark and the mapping of threads to cores at run-time, while also accounting for the impact of process variations. We have developed a light-weight temperature prediction mechanism that determines the impact of different candidate solutions on the chip thermal profile. Experimental evaluation of the proposed techniques on a simulated 8×8 many-core processor, and across a range of chips to account for process variations, show that the total instruction throughput is increased by $1.8\times$ on average while keeping the temperature within the safe limits, when compared with state-of-the-art approaches.**

## I. Introduction and Related Work

In the nanometer era, the exponential dependence of leakage power consumption on threshold voltage has constrained further threshold- and supply-voltage scaling. Consequently, the power density is increasing with technology scaling at a rate that it cannot be sustained. This gives rise to the so-called *Dark Silicon* problem that constraints the maximum number of transistors that can be simultaneously powered-on in the *nominal* operating mode for a given $TDP$ budget in order to ensure safe operation (i.e., where the maximum chip temperature $T_{max}$ is below the thermal safe temperature, $T_{safe}$), thus leaving a fraction of the chip *'dark'* [1, 2]. In case the $TDP$ is exceeded, $T_{max}$ will escalate beyond the cooling capacity (and surpass $T_{safe}$), thus resulting in thermal run-away unless the chip is throttled. Based on the technology scaling data from ITRS [3] and Intel, analytical studies in [1] have predicted that more than half the transistors on a chip may be dark on many-core systems executing massively parallel workloads. *The key challenge that we target in this paper is: can the abundance of 'potentially dark' cores still be harnessed to improve performance within thermal constraints, and if so, how?*

### A. Related Work and Scientific Challenges

The related work addresses the dark silicon problem mainly based on the following design philosophies:

*(1) Application-Specific Accelerators and Architectural Heterogeneity:* Accelerator-based architectures for increasing energy efficiency are proposed in [7]. The work in [8] introduces energy-efficient specialized cores for energy-intensive sections of an application, which are selectively activated on demand. Several papers have also explored micro-architectural heterogeneity to combat the dark silicon problem. The work in [9–11] targets architectural synthesis of heterogeneous dark silicon processors from performance and reliability aspects under area/power constraints. These design time approaches are orthogonal to the work presented in this paper, since our work focuses on run-time techniques.

*(2) Dynamic Power and Thermal Management:* Recent studies have explored the problem of performance maximization for dark silicon processors [12] but not incorporating thermal constraints. More recently, the work in [13] has focused on power-budgeting under thermal constraints, i.e., determining the power that can be allocated to each core for a given pattern of active/dark cores and a given thread-to-core mapping, but the authors do not discuss which cores to keep active/dark and how threads are mapped to cores, which is the goal of our paper. Furthermore, the work in [13] does not account for the impact of manufacturing process variations or the leakage-temperature feedback loop, as we do in this work.

A related idea is that of computational sprinting [14], that activates dark cores for short periods of time (hundreds of miliseconds) with the goal of minimizing response time. The additional heat generated during this period is dissipated by a phase-change material. Similarly, boosting techniques (like Intel's Turbo Boost [15] and AMD's Turbo CORE [16]) leverage the temperature headroom to sprint a core (executing the high-ILP application) for a short time by allocating the complete power budget to it. Our work is orthogonal to the sprinting and boosting approaches discussed above, since we determine spatial (and not temporal) patterns that maximize performance while aiming at keeping the temperature below $T_{safe}$ A diametrically opposite approach to utilizing dark cores is *near-threshold computing* [17] that allows cores to operate at a supply voltage near the threshold voltage, thus allowing several otherwise dark cores to be turned on. Near-threshold computing techniques are orthogonal to our approach since, in this paper, we only consider nominal voltage and frequency modes of operation.

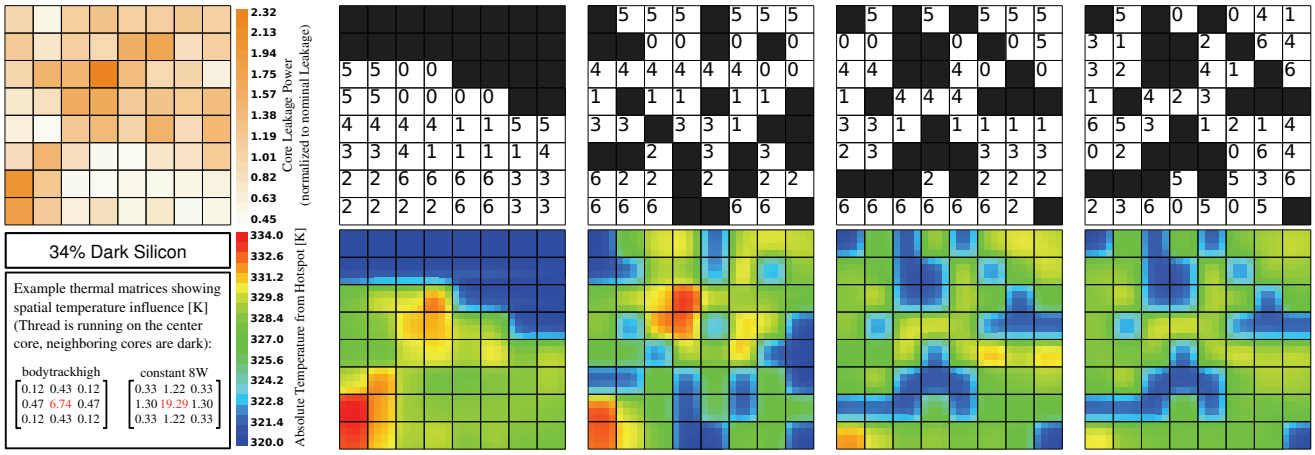
*The broad open question that is unaddressed in literature is the run-time optimization of the subset of cores to be kept dark and the thread-to-core mapping on the thermal profile of the chip, while taking into account the impact of process variations. We refer to this as variability-aware dark silicon management.* We will now illustrate our concept of dark silicon management using a motivational case-study.

### B. Motivational Case Study

In conventional many-core processors that dont have dark silicon constraints, there exists only one $TDP$ mode, i.e., powering-on all cores at nominal voltage/frequency to maximizing performance within a $TDP$ constraint. In contrast, dark silicon many-core processors exhibit a multitude of $TDP$ modes, each resulting in a starkly distinct thermal and leakage power profile. The resulting thermal headroom can be used to improve performance by switching-on additional core(s), while keeping the maximum chip temperature below $T_{safe}$. To illustrate this more clearly, we first define the term *dark core patterning* to refer to the subset of cores that are kept dark in order to reduce peak temperature under the $TDP$ constraint. The case study in Fig. 1 for a 64-core chip with a process variation map shown in Fig. 1a, 22 'dark' cores, (more experimental setup details in Fig. 1 and Section IV-A) shows different thermal profiles as a result of different dark core patterns. In each setting, the processor is executing the same set of multi-threaded applications, each with the same number of threads and same number of powered-on cores at full voltage/frequency.

*Case–1: Contiguous Dark Cores:* A naïve approach is to power-on a set of contiguous cores (as targeted in [18]), as shown in Fig. 1b.

978-3-9815370-4-8/DATE15/©2015 EDAA




(a) Subthreshold leakage power variation coefficient map of chip #10 example.

(b) Temperature after $13.2ms$ resulting from *contiguous pattern*. $T_{max} = 333.95K$

(c) Temperature after $13.2ms$ resulting from *random-1 pattern*. $T_{max} = 333.30K$

(d) Temperatures after $13.2ms$ resulting from *random-2 pattern*. $T_{max} = 331.98K$

(e) Temperature after $13.2ms$ resulting from *random-2 pattern* but with *optimized task placement*. $T_{max} = 330.77K$

**Experimental Setup:** 8x8 Alpha 21264 with 2MB L2 cache, 3GHz, 1.13 V, 22nm technology data scaled to 11nm as per ITRS-provided factors [3] to reflect dark silicon, McPAT v1.1 [4], Size of single core: $1.70 \times 1.75mm^2$, Gem5 [5], HotSpot [6] ($400\frac{W}{mK}$ Heatsink heat transfer, $100W\frac{W}{mK}$ silicon heat transfer), Multi-threaded Applications of Parsec ("bodytrackhigh", "x264" with 5 HD-video sequences); see further details in Section IV-A.

Fig. 1: Leakage Power variation, Different dark core patterns, and Thermal profile depending on the mapped tasks with 34% dark silicon; multiple instances of seven different tasks executing; Heat influence to the neighboring cores is due to spatial influences through both silicon and the heatsink on top of that.

However, this leads to temperature hot spots in some on-chip regions due to increased power density. Fig. 1b shows that a short time after tasks starts executing, a maximum temperature of $T_{max} = 333.95K$ is reached and the resulting thermal hotspot is visible near 'bottom-left' of the chip.

*Case–2: Random Dark Core Patterns:* A random dark core pattern alleviates the power density problem through spatial heat dissipation[1]. The random pattern shown in Fig. 1c results in $T_{max} = 333.30K$, and second random pattern (Fig. 1d) results in an even lower $T_{max} = 331.98K$. An efficient patterning algorithm would provide even more reductions in the temperature; as we will show in the results (Section IV-C).

*Case–3: Impact of Application Mapping for the same Pattern:* For a given pattern, a different mapping of threads-to-cores results in a different temperature due to heterogeneity in thread characteristics. Fig. 1e shows a further reduction of $1.21K$ in the peak temperature (i.e., $T_{max} = 330.77K$), showing that it may be beneficial to allocate more 'dark' cores around the core executing a workload-intensive thread.

*Accounting for Process Variations:* It is also important to note that due to process variations, the same dark core patterning on different chips will result in different thermal profiles as a result of leakage power variations. As a result, the best dark core patterning and mapping solution will vary from chip-to-chip, and therefore, the dark core patterning algorithm needs to consider a *superposition* of the variability and temperature maps. For instance, when considering the process variation map shown in Fig. 1a, the two cores in the bottom-left corner should be kept dark to lower $T_{max}$.

*Summary of Motivation Discussion:* Inefficient dark core patterning decisions (for instance the contiguous pattern discussed above) may lead to temperature hot spots that not only result in an increased leakage power but also throttling/shutting-down of cores in vicinity of the hotspot, thus degrading performance. Run-time **dark silicon management** through joint consideration of (a) dark core patterning, (b) thread-to-core mapping, and (c) accounting for the impact of manufacturing process variations, introduces new opportunities to optimize the temperature profile by selecting amongst one of many available $TDP$ modes, which can then be leveraged to increase performance while keeping the chip temperature below $T_{safe}$.

[1]this means all paths of heat conduction, not only direct paths from silicon → silicon, but also silicon ↔ heatsink ↔ silicon

### C. Our Novel Contributions and Concept Overview

In this paper, we present a novel *variability-aware dark silicon management* (DaSiM) technique that performs dark core patterning and thread-to-core mapping at run-time while accounting for thread heterogeneity and leakage power variations due to manufacturing variability. Furthermore, we leverage the reduction in peak temperature that ensues from the dark core patterning and mapping decisions to power on additional cores while keeping $T_{max}$ below $T_{safe}$.

To minimize the run-time overhead of the proposed technique, we also propose a *computationally lightweight* yet *accurate* temperature prediction mechanism that predicts the thermal profile that would result from a particular patterning and mapping decision. Finally, a computationally efficient heuristic is proposed that leverages the light-weight temperature prediction to optimize the patterning and mapping decisions. We show the accuracy of our temperature prediction mechanism when compared to *Hotspot* [6] simulations, and show that our proposed dark silicon management approach outperforms other approaches in terms of the resulting thermal profiles, performance, performance-per-$T_{max}$, and the number of task migration events.

## II. System Model and Preliminary

**Hardware Architecture Model:** We consider a tiled many-core processor $\mathscr{C} = \{C_1, C_2, ..., C_N\}$ with $N$ homogeneous cores organized as grid of $W \times H$ cores on the chip. $x_i$ and $y_i$ ($x \in [1, W]$, $y \in [1, H]$) denote the coordinates of the core $i$. Each core $C_i \in \mathscr{C}$ has a private L1-I and L1-D cache — in this paper we focus only on dark silicon management for cores and assume fixed area and power budgets for the uncore components. $f$, $p_i$, and $ps_i$ denote the nominal frequency, nominal total power consumption (this will be refined shortly), and power state (active or dark) of core $C_i$, respectively. Note that although the cores are micro-architecturally homogeneous, their leakage power consumption values can differ due to process variations, as will be discussed below. In a dark silicon processor, only a subset of cores

are powered-on; $N_{ON} = \sum_{i=1}^{N} ps_i$ and $N_D = N - N_{ON}$ denote the number of powered-on and dark cores, respectively.

$$\mathbf{ps}_i = \begin{cases} 0, \text{if core } C_i \text{ is dark/power-gated} \\ 1, \text{if core } C_i \text{ is power-on} \end{cases}.$$

**Application Model:** We assume a set of $M$ executable multi-threaded applications $\mathscr{P} = \{P_1, P_2, \ldots, P_M\}$. Each application $P_j \in \mathscr{P}$ has a set of threads $P_j = \{\tau_1, \tau_2, ..., \tau_{K_j}\}$, where $K_j$ is the number of threads of application $P_j$. We use the malleable application model [19, 20], which allows $K_j$ to be configurable based on the number of available active cores. The total number of active executing threads is given as: $n_\tau = \sum_{j=1}^{M} K_j$. Furthermore, $IPS_{(i,j,k)}$ denotes the throughput, measured in Instructions-Per-Second (IPS) of every thread and $p_{i,j,k}$ represents its power consumption. Threads execute on dedicated cores and the thread-to-core mapping function is given as:

$$\mathbf{m}_{i,j,k} = \begin{cases} 1, \text{if thread } \tau_k \text{ of program } P_j \text{ is executing on } C_i \\ 0, \text{otherwise} \end{cases}.$$

**Process Variation Model:** In this paper, we only deal with core-to-core leakage power variations and consider frequency variations are factored out via well-established guard-banding, i.e., all cores on a chip execute synchronously at a conservatively determined clock frequency. To model leakage power variations, we use existing models as proposed by [21, 22], where the chip area is partitioned into $N_{chip} \times N_{chip}$ grid points overlayed over cores. Each grid point $(u, v) \in [1, N_{chip}]^2$ has a process parameter $\vartheta_{u,v}$ which is modeled as a Gaussian random variable with mean $\mu_\vartheta$ and standard deviation $\sigma_\vartheta$, and spatially correlated with parameter $\alpha$ (see [21, 22] for more details on the spatial correlation model). The process variation affects threshold voltage ($V_{th}$) and has an exponential impact on leakage power. The total power consumption of a core $C_i$ executing thread $k$ of application $P_j$ is given as:

$$p_{i,j,k} = p^{dyn}_{i,j,k} + \sum_{(u,v)\in C_i} p^{leak}_{u,v} \times e^{V_{th}\vartheta_{u,v}/V_T} \quad (1)$$

where: $p^{dyn}_{i,j,k}$ is the thread dependent dynamic power consumption of thread $k$ of application $P_j$ on core $C_i$, $p^{leak}_{u,v}$ is the nominal leakage power consumption of grid point $(u, v)$, and $V_T = KT_i/q$ is the thermal voltage ($T_i$ is the temperature of core $C_i$). As it can be seen from Equation 1, core-to-core differences in dynamic power arise as a consequence of heterogeneity in thread characteristics, while the leakage power variability is due to manufacturing process variations. Temperature dependence of leakage power is captured by the thermal voltage $V_T$ which is temperature dependent. Finally, we assume that each core $C_i$ has at least one soft thermal sensor $T_i$.

## III. Variability-Aware Dark Silicon Management

Fig. 2 illustrates the basic flow of our variability-aware dark silicon management (DaSiM) technique, showing the connections between different components. It consists of two main components (detailed in the following sub-sections).

(1) The *dark core patterning and thread mapping technique* (Section III-C) that proactively determines the power-state control of cores in conjunction with mapping of multi-threaded applications, such that, the sum of $IPS$ of all concurrently executing applications is maximized while keeping $T_{max}$ below $T_{safe}$.

(2) The *thermal prediction technique* (Section III-B2) estimates the chip thermal profile for a candidate solution (i.e., a dark core pattern and a corresponding thread mapping) at run-time. To enable this, we derive the so-called *thermal matrix* **TM** for each application through thermal profiling (Section III-B1), which provides the temperature influence of this thread on the neighboring cores, taking into account the monitored temperature from the thermal sensors. The prediction technique also accounts for leakage power variations between cores, and for the temperature-leakage feedback loop. It may occur that in case of thermal mispredictions, the actual chip temperature exceeds $T_{safe}$, in which case the chip's fail-safe dynamic thermal management (DTM) controller will throttle the chip or migrate threads (thus sacrificing performance). It is crucial to minimize the number of such events and therefore, we evaluate this metric in our experimental results.

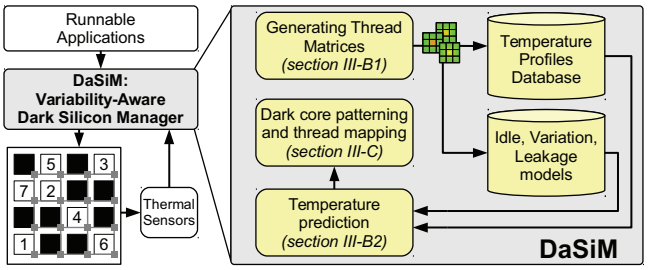


Fig. 2: Flow of Patterning, Mapping, Prediction

### A. Problem Formulation

The goal is to **find** joint patterning and mapping:

$$m_{i,j,k}\, \forall P_j \in \mathscr{P}, \tau_{j,k} \in P_j, s.t., C_i \in \mathscr{C}\ executes\ \tau_{j,k} \quad (2)$$

*that fulfills the following* **constraints:**
(1) The temperature $T_i$ of any core $C_i \in \mathscr{C}$ on the chip must be below $T_{safe}$ (Eq. 3);

$$T_i \leq T_{safe};\ \forall i \in [1, N] \quad (3)$$

and (2) each core $C_i \in \mathscr{C}$ executes only one thread $\tau_{i,j,k}$ (Eq. 4);

$$\sum_{j=1}^{M} \sum_{k=1}^{K_j} m_{i,j,k};\ \forall i \in [1, N] \leq 1 \quad (4)$$

*while optimizing for the following* **goals:**
(1) the sum $IPS$ all applications $P_j \in \mathscr{P}$ is maximized (Eq. 5);

$$\max_m \Big( \sum_{\forall C_i \in \mathscr{C}} \sum_{\forall P_j \in \mathscr{P}} \sum_{\forall \tau_k \in P_j} m_{i,j,k} \times IPS_{i,j,k} \Big) \quad (5)$$

and (2) for two equally valid solutions with same $IPS$, the one with the $T_{minmax}$ is selected (Eq. 6).

$$\min\left(\max_i (T_i)\right). \quad (6)$$

The problem can be formulated efficiently as an integer linear programming problem (ILP), but that would result in unacceptably high computational overhead, since this mechanism needs to be executed at run-time once every scheduling step. In what follows, we will present our solution.

### B. Prediction of Thermal Profile for a Given Set of Applications

Our temperature prediction technique uses superposition to predict temperature for a given patterning and thread-to-core mapping, by overlaying different (spatio-temporal) chip temperature profiles considering individual thread executions. At the same time, we also account for a factor that incorporates temperature-dependent leakage increase of a core due to the neighboring cores' temperature. Our prediction technique operates in two main steps: (1) offline temperature profile generation for different application threads (Section III-B1); and (2) online chip thermal profile prediction (Section III-B2) based on the offline profiling.

*1) Temperature profiling:* We profile each thread's temperature offset and store it in the form of a 2D *thermal matrix* **TM** which provides the thermal impact of a core $C_i$ executing a thread on its surrounding cores

(and itself). In particular, the **TM** for thread $\tau_k$ from application $P_j$ executing on core $C_i$ is given as follows:

$$\mathbf{TM}_{\tau_{(i,j,k)}} = \begin{bmatrix} \Delta T_{x_i-R,y_i+R} & \cdots & \Delta T_{x_i+R,y_i+R} \\ \ddots & \iddots & \\ \vdots & \Delta T_{x_i,y_i} & \vdots \\ & \iddots & \ddots \\ \Delta T_{x_i-R,y_i-R} & \cdots & \Delta T_{x_i+R,y_i-R} \end{bmatrix}$$

where $\Delta T_{x_i,y_i}$ is the temporal temperature difference at core $C_i$ located at $(x_i, y_i)$ between the steady-state temperature after executing the thread for time period $t_{epoch}$ and the initial temperature. Ideally, the radius $R$ should be large enough to encompass all the cores on the chip, but $R$ can be reduced to lower the computational cost of the prediction mechanism, albeit at the cost of reduced accuracy. In addition to the case when a core is executing a thread, we also determine $\mathbf{TM}_{idle}$ for the idle case, i.e., when a core is in the active idle state and not executing a thread. For large-sized chips, to reduce the complexity, we can restrict the spatial granularity of **TM** by a radius $R$.

Algorithm 1 presents the procedure to determine the entries of the **TM** matrix. Each thread is executed on a core in the center of the chip and its temperature along with the temperature of its neighboring cores within the window of radius $R$ (i.e., $\forall h \in \mathscr{C}, s.t., x_h, y_h < R$) are obtained to compute the **TM** entries. This process is repeated for all application threads. To lookup a corresponding entry for a core $C_i$ in the matrix **TM** of its neighbor $C_h$, we define a lookup function **TM.get**$(\tau_k, i, h)$. similarly we define a function **TM.update**$(\tau_k, i, \Delta T_{x_i,y_i}(h))$ to update an entry in the **TM** matrix with a new temperature difference value corresponding to the neighbor $C_h$. Note that we subtract $\mathbf{TM}_{idle}$ because it will later be considered in Algorithm 2 with temperature- and variation-dependent leakage power.

**Algorithm 1** Profile the temperature influence matrices

1: **for all** $\forall \tau_k \in P_j \in \mathscr{P}$ **do**
2: Run thread $\tau_k$ for time $t_{epoch}$ alone on the system on a core $i$ at the center of the chip
3: **for all** cores $C_h \in \mathscr{C}$ at positions $x_h, y_h$ in window around $C_i$, including itself **do**
4: $\Delta T_{x_h,y_h} \leftarrow T(t_0)_h - T(t_0 - t_{epoch})_h -$ **TM.get**$(idle, h, i)$
5: **TM.update**$(\tau_k, i, \Delta T_{x_i,y_i}(h))$
6: **end for**
7: **end for**

*2) Temperature prediction:* In order to predict the chip thermal profile (for evaluating a candidate solution by our dark silicon management technique in Section III-C), we apply superposition, such that different **TM**s of the concurrently executing threads are overlaid.

In the absence of temperature dependent leakage, the final temperature of any core can be obtained by summing up the temperatures of the **TM**s that overlap with that core.

In the presence of temperature dependent leakage however, simply using a superposition is not sufficient and the impact of increased temperature (due to the temperature dependent leakage) needs to be incorporated in the predicted thermal profile.

Algorithm 2 illustrates the procedure for predicting the chip thermal profile. It iterates overall neighbors' **TM**s that will influence the temperature of the selected active core. Afterwards, it estimates (1) the variation-dependent static temperature influence $T_{varstatic}$; and (2) the updated temperature $T_{static,now}$ incorporating the effects of temperature-dependent leakage (calculated through the function **tempDepLeakage**). This function is obtained using a power simulator; ; see Section IV-A. In case a new thread starts executing, or a thread finishes its execution, or a thread is replaced by another thread, our algorithm updates the temperature profiles and temperature-dependent leakage (lines 8-14).

**Algorithm 2** Calculate the predicted temperature of $C_i$

1: $T_{pred} \leftarrow T(t_0)_i$
2: **for all** $\forall$ cores $h$ to be overlayed around core $C_i$ to be predicted **do**
3: $T_{varstatic} \leftarrow$ **TM.get**$(idle, i, h) \cdot v_h$
4: $T_{static,prev} =$ tempDepLeakage$(T_{varstatic}, T(t_0 - 1)_i)$
5: $T_{static,now} =$ tempDepLeakage$(T_{varstatic}, T(t_0)_i)$
6: $T_{staticdiff} = (T_{static,now} - T_{static,prev})$
7: **if** new thread starting **then**
8: $T_{pred} \leftarrow +$**TM.get**$(\tau_{new}, i, h) + T_{varstatic} + T_{staticdiff}$
9: **else if** thread quitting and core powergating **then**
10: $T_{pred} \leftarrow -($**TM.get**$(\tau_{current}, i, h) + T_{varstatic} + T_{static,now})$
11: **else if** thread gets replaced by another thread **then**
12: $T_{pred} \leftarrow$ **TM.get**$(\tau_{new}, i, h) -$ **TM.get**$(\tau_{current}, i, h) + T_{staticdiff}$
13: **else**
14: $T_{pred} \leftarrow T_{staticdiff}$
15: **end if**
16: **if** if thread present and longer-time estimate required **then**
17: recursively increase $T_{static,now}$ and $T_{pred}$ with dependence on each other
18: **end if**
19: **end for**
20: **if** no thread is running on any core in radius **then**
21: $T_{pred} \leftarrow$ ambient_trend$(T_{predict}, T_{ambient})$
22: **end if**
23: **return** $T_{pred,i} = T_{pred}$

Furthermore, in case of estimation over multiple epochs, the temperature-dependent leakage is increased recursively (line 17). If no thread is executing in the overlaying radius, we apply an ambient decline function: **ambient_trend**$(T_{core}, T_{ambient}) \leftarrow T_{ambient} + (T_{core} - T_{ambient}) \cdot exp(-\gamma \cdot \Delta t)$.

### C. Dark Silicon Patterning and Application Mapping

Algorithm 3 presents our proposed heuristic for joint dark core patterning and thread mapping. We iterate through the threads starting with the overall application with highest temperature at the middle entry in the respective **TM** first (line 2). Next, we iterate through the cores starting from the coldest free core $C_i$ and assigned the thread currently under consideration to it (line 4-18). For each assignment, we make a prediction for all cores in the system and save the maximum temperature in $T_{max}$ (lines 7-12). Finally, the thread-to-core mapping with the lowest $T_{max}$ is picked assuming it is lower than $T_{safe}$ (lines 14-20). In case no such mapping is found, we continue iterating through other threads because valid mappings for these threads might still exist.

Fig. 3 presents an example scenario illustrating the intermediate steps of our algorithm functioning: (A) thermal map at the start, i.e., mapping the first thread; (B) thermal map at an intermediate solution with 'dark' and 'powered-on' cores and 26 mapped threads; (C) thermal map at the final solution with the final dark core pattern and 52 mapped threads.

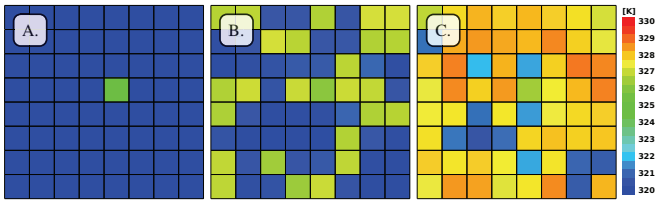
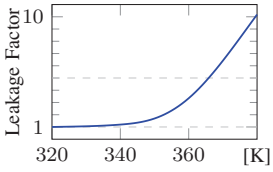


Fig. 3: Temperature profile from prediction of the whole chip after the first placed threads (A.), after 26 threads (B.), and final 52 threads (C.) to stay below $T = 329K$, $T_{max} = 328.872K$

**Algorithm 3** DaSiM

$\mathscr{M}' \leftarrow \mathscr{M}$
**for all** $\forall \tau_k \in P_j \in \mathscr{P}$, hottest to coldest **do**
  $T_{minmax} \leftarrow \infty$
  **for all** dark or idle cores $i$, coldest to hottest core **do**
    $\mathbf{m}'_{i,j,k} \leftarrow 1$
    $T_{max} \leftarrow 0$
    **for all** cores $h = 1..N$ **do**
      $T_{temp} \leftarrow T_{pred,h}(t_0 + 1)$
      **if** $T_{temp} > T_{max}$ **then**
        $T_{max} \leftarrow T_{temp}$
      **end if**
    **end for**
    $\mathbf{m}'_{i,j,k} \leftarrow 0$
    **if** $T_{max} < T_{minmax} \wedge T_{max} < T_{safe}$ **then**
      $T_{minmax} \leftarrow T_{max}$
      $Pos_{minmax} \leftarrow \{i\}$
    **end if**
  **end for**
  **if** $T_{minmax} < T_{safe}$ **then**
    $\mathbf{m}_{Pos_{minmax},j,k} \leftarrow 1$
  **else**
    No cold enough placement for this thread found, but test further (possibly colder) threads $\rightarrow$ *next*
  **end if**
**end for**




| Application | Threads | | Threads | |
|---|---|---|---|---|
| foreman | 20% dark silicon | 7 | 40% dark silicon | 5 |
| sunflower | | 8 | | 5 |
| blue | | 8 | | 5 |
| bodytrackhigh | | 7 | | 5 |
| eledream | | 7 | | 6 |
| station | | 7 | | 6 |
| football | | 7 | | 6 |

Fig. 4: **Left:** Leakage power factor at elevated temperatures over baseline ($320K$) according to McPAT, **Right:** Mixes of concurrently executing threads for two cases of dark silicon.

## IV. Results and Discussion

### A. Experimental Setup

We perform cycle-accurate simulations on Gem5 [5] tightly integrated with McPAT (v1.1 March 2014) [4] to obtain performance and power traces, respectively, for different multi-threaded applications from the *Parsec* benchmark suite. These traces are fed into an in-house many-core simulator for simulating large-scale chips. A number of process variation maps are generated based on the model of Section II and overlaid on the chip floorplan to obtain the leakage power. For temperature simulations, we have integrated Hotspot [6] as a library into our simulator. Several experimental settings (technology, chip configuration, architecture, etc.) have already been explained in Fig. 1. Here, we only provide the additional information. The nominal subthreshold leakage for each core in active mode is $1.18W$, and $0.019W$ left in power-gated mode. For temperature-dependent leakage, we generated an interpolated model from the *McPAT* simulator [4] (Fig. 4) estimating the temperature-dependent increase in the leakage power after a given time-period. In our experiments, we adopted time-period= $6.6ms$, $\lambda = 0.036$, $\xi = 3$, and $\gamma = 0.000525$. This factor is then added to the variation-dependent leakage power to obtain the total leakage power.

We emulated the *virtual thermal sensors* (used by our dark silicon management technique) on our manycore simulator using the integrated Hotspot library. We use an initial temperature of $T_{initial} = 340K$ for the whole chip, and $T_{ambient} = 320K$. In our experiments, we also model a fail-safe DTM module that triggers when $T_{max}$ exceeds $T_{safe}$ (e.g., because of temperature mis-prediction) and halts threads on hot cores / migrates from hot to cold cores. Our workloads consist of several *Parsec* application mixes with different number of threads per application.

### B. Temperature Prediction Accuracy

To illustrate the accuracy of our temperature prediction, Fig. 5 (left) shows the actual (Hotspot) and predicted temperatures for one of our simulations, while Fig. 5 (right) shows the boxplot[2] of absolute temperature errors over all our experimental simulations. Over several process variation maps, the maximum absolute error over all simulations is only $0.597K$.

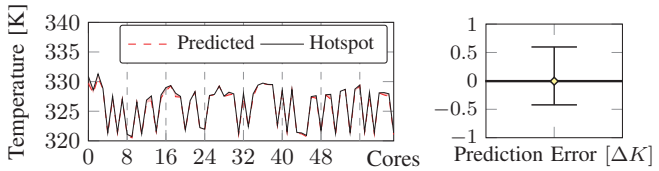



Fig. 5: **Left:** Example of thermal predicted vs. actual temperature, **Right:** Absolutes of temperature prediction errors

### C. Comparison with State-of-the-Art and Different Decision Policies

We compare the following dark silicon management algorithms for speedup, power, $T_{max}$, performance-per-power, performance-per-$T_{max}$, and DTM events using two dark silicon budgets (20%, 40%) and over 50 chips with different process variation maps.

- **C**: Contiguous mapping, placing threads closer to each other as adopted in [18, 23].
- **R**: Random patterning and mapping onto the cores; same random pattern for all simulations.
- **V**: Variance patterning; choose a pattern depending on the cores variance, prefer colder cores first, map threads in random order.
- **D**: Our DaSiM technique as described in subsection III-C, our $T_{safe}$ is set to $343.15K (70^{\circ}C)$

In all results that follow, the results from other techniques are normalized to the contiguous (**C**) approach. We make several key observations from our experimental results as discussed in the following. Fig. 6 plots the maximum temperatures for 20 % and 40 % dark silicon, and we observe that DaSiM outperforms all techniques, achieves the lowest $T_{max}$ for almost all the simulations (as illustrated by the concentrated boxplot distribution), and shows an improvement of up to 3% (on average) in temperature (in relation to a $320K$ offset) versus the *contiguous* baseline. This reduced temperature also results in substantially improved thermal efficiency (defined as the throughput per unit peak temperature), i.e., 1.6-fold for 20% and as much as 1.89-fold for 40% dark silicon (see Fig. 10).

The lower temperature allows DaSiM to power-on more cores that results in an increased power (see Fig. 7), which is still desirable as this additional power is used to improve the performance without violating the thermal constraints, i.e., staying below $T_{safe}$. Therefore, a more relevant metric is performance-per-$T_{max}$ efficiency for dark silicon chips. Despite high power input, DaSiM improves the average power efficiency (defined as the throughput per unit power consumption) by 5%–7% (see Fig. 9) for 20% and 40% of minimum dark silicon, respectively, due to a significant performance improvement while operating within safe thermal limits.

As a result, DaSiM provides a *significant* increase of 1.59-fold for 20% dark silicon and 1.84-fold for 40% dark silicon in the average throughput (IPS) compared to the competing approaches (see Fig. 8).

[2]A boxplot provides results summary in form of: minimum (botom line), maximum (top line), three quartiles representing 25%, 50%, and 75% of the results within this range, and the average results as the diamond symbol.

This is in part because it is able to turn on more extra cores than other techniques, and in part because it triggers fewer DTM events where the chip needs to be throttled or tasks migrated. As seen in Fig. 11, while there are zero DTM events for DaSiM, as it does not turn on cores when the thermal headroom is not enough for a sustainable execution time of a thread, other algorithms need to rely on DTM and the increased overheads from throttling and migrating threads. In particular, this is the case with the *variance* patterning that power-on cores and places threads in close vicinity due to the existance of less-leaky cores in groups as a result of correlation in the process parameter. Therefore, the *variance* patterning results in thermal hotspot that consequently leads to more DTM events (approx. 2x compared to the *contiguous* pattern) as shown in **V** boxplot of Fig. 11.

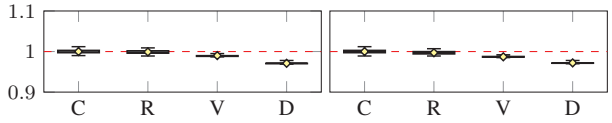



Fig. 6: $T_{max}$ per simulation across 50 chips in relation to contiguous and $T_{ambient}$ offset, **Left:** min. 20 % dark silicon **Right:** min. 40 %

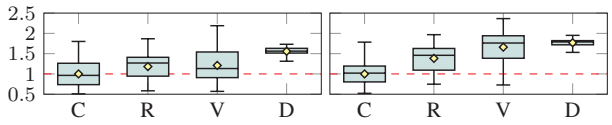



Fig. 7: Total power use per simulation across 50 chips, **Left:** min. 20 % dark silicon **Right:** min. 40 %

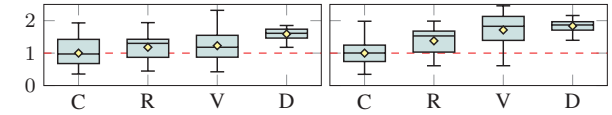



Fig. 8: Speedup across 50 chips in relation to contiguous, **Left:** min. 20 % dark silicon **Right:** min. 40 %

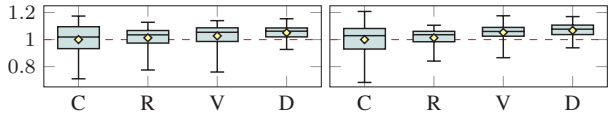



Fig. 9: Performance / Power in relation to contiguous, **Left:** min. 20 % dark silicon **Right:** min. 40 %

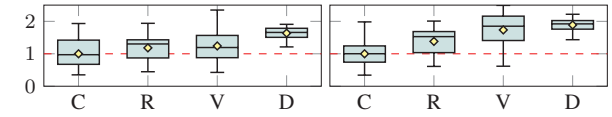



Fig. 10: Performance / Max. Temperature in relation to contiguous, **Left:** min. 20 % dark silicon **Right:** min. 40 %

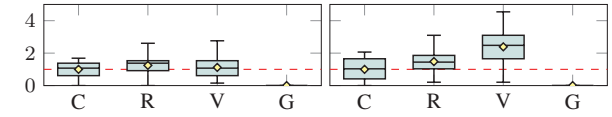



Fig. 11: DTM events (shutdown, migrate) across 50 different chips in relation to contiguous, **Left:** min. 20 % dark silicon **Right:** min. 40 %

## V. Conclusion

In this paper, we have proposed a dark silicon management (DaSiM) technique that employs joint dark silicon patterning and thread-to-core mapping to optimize the thermal profile of a many-core processor with dark silicon constraints. Consequently, we propose to leverage the reduction in peak chip temperature that enables our technique to power-on additional cores, thereby increasing performance. At run-time, an efficient heuristic is employed to jointly optimize the dark silicon pattern and thread-to-core mapping, which is enabled by a light-weight temperature prediction mechanism that provides an estimate of the chip temperature profile resulting from a candidate solution. In addition, our proposed technique synergically accounts for the core-to-core variations in leakage power dissipation that arise as a consequence of manufacturing process variability while determining the best solution. Our experimental results demonstrate the benefits of the proposed technique compared to state-of-the-art and other less-sophisticated decision policies, in terms of performance, performance per unit maximum temperature, performance/Watt without incurring any DTM events. Our DaSiM technique is the first to leverage dark transistors/cores on the chip to optimize the chip's thermal profile, and paves the road for further research into the opportunities offered by the dark silicon.

## Acknowledgments

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89); http://invasic.de.

## References

[1] H. Esmaeilzadeh, E. Blem, R. St.Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *International Symposium on Computer Architecture (ISCA)*, pages 365–376, 2011.

[2] M. Shafique, S. Garg, J. Henkel, and D. Marculescu. The eda challenges in the dark silicon era: Temperature, reliability, and variability perspectives. In *Design Automation Conference (DAC)*, 2014.

[3] International technology roadmap for semiconductors, http://public.itrs.net/reports.html.

[4] Li et al. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Symposium on Microarchitecture*, pages 469–480, 2009.

[5] N. Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, pages 1–7, August 2011.

[6] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *ACM SIGARCH Computer Architecture News*, volume 31, pages 2–13. ACM, 2003.

[7] M. J. Lyons, M. Hempstead, G.-Y. Wei, and D. Brooks. The accelerator store: A shared memory framework for accelerator-based systems. *ACM Trans. Archit. Code Optim.*, 8(4):48:1–48:22, 2012.

[8] G. Venkatesh et al. Conservation cores: reducing the energy of mature computations. In *Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 205–218, 2010.

[9] J. Allred et al. Designing for dark silicon: a methodological perspective on energy efficient systems. In *ISLPED*, 2012.

[10] Y. Turakhia et al. Hades: Architectural synthesis for heterogeneous dark silicon chip multi-processors. In *DAC*, 2013.

[11] F. Kriebel et al. Aser: Adaptive soft error resilience for reliability-heterogeneous processors in the dark silicon era. In *Design Automation Conference (DAC)*, 2014.

[12] T. Muthukaruppan et al. Hierarchical power management for asymmetric multi-core in dark silicon era. In *(DAC)*, pages 174:1–174:9, 2013.

[13] X. Hu et al. Thermal-sustainable power budgeting for dynamic threading. In *Design Automation Conference (DAC)*, pages 1–6. ACM, 2014.

[14] A. Raghavan et al. Computational sprinting. In *Symposium on High-Performance Computer Architecture (HPCA)*, pages 1–12, 2012.

[15] Efi Rotem et al. Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Micro*, 32(2):20–27, 2012.

[16] S. Nussbaum. Amd trinity apu. HotChips '12, 2012.

[17] U.R. Karpuzcu et al. Energysmart: Toward energy-efficient manycores for near-threshold computing. In *Symposium on HPCA*, 2013.

[18] M. Fattah et al. Smart hill climbing for agile dynamic mapping in many-core systems. In *Design Automation Conference (DAC)*, 2013.

[19] H. Shojaei et al. A parameterized compositional multi-dimensional multiple-choice knapsack heuristic for cmp run-time management. In *Design Automation Conference (DAC)*, pages 917–922, 2009.

[20] Gerald S. and Matthew L. Moldable parallel job scheduling using job efficiency: An iterative approach. In *(JSSPP), ACM SIGMETRICS*, 2006.

[21] J. Xiong, V. Zolotov, and L. He. Robust extraction of spatial correlation. *Computer-Aided Design of Integrated Circuits and Systems, Transactions on*, 26(4):619–631, April 2007.

[22] Raghunathan et al. Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors. In *Conference on Design, Automation and Test in Europe*, pages 39–44, 2013.

[23] M. Fattah, M. Palesi, P. Liljeberg, J. Plosila, and H. Tenhunen. Shifa: System-level hierarchy in run-time fault-aware management of many-core systems. In *Design Automation Conference (DAC)*, 2014.