

# adBoost: Thermal Aware Performance Boosting through Dark Silicon Patterning

Anil Kanduri, *Student Member, IEEE*, Mohammad-Hashem Haghbayan, *Student Member, IEEE*, Amir M. Rahmani, *Senior Member, IEEE*, Muhammad Shafique, *Senior Member, IEEE*, Axel Jantsch, *Member, IEEE*, Pasi Liljeberg, *Member, IEEE*

**Abstract**—Increasing power densities of many-core systems leaves a fraction of on-chip resources inactive, referred to as dark silicon. Efficient management of critical interlinked parameters - power, performance and temperature can improve resource utilization and mitigate dark silicon. In this paper, we present a run-time resource management system for thermal aware performance boosting using a dark silicon aware run-time application mapping strategy. The mapping policy patterns inactive cores among active cores for relatively lower and even distribution of operating temperatures. This provides enough thermal headroom for boosting the frequency of active cores upon performance surges and allows sustained boosting periods, improving the performance further. We design a controller for thermal aware performance boosting that decides on efficient allocation utilization of power budget and thermal headroom obtained from patterning. Our strategy yields up to 37% better throughput, 29% lower waiting time and up to  $2 \times$  longer boosting periods, in comparison with other state-of-the-art run-time mapping policies.



**Index Terms**—Dark Silicon; Run-time Mapping; Frequency Scaling;

## 1 INTRODUCTION

Power densities of many-core systems are increasing due to disproportional voltage and technology node scaling, contributing to higher on-chip temperatures [1], [2]. As a result, a section of the chip has to remain inactive to function within safe thermal limits. This inactive circuitry of the chip is termed as Dark Silicon [3]. Computer systems cap the power consumption at a fixed upper bound to ensure that the on-chip temperature due to power dissipation is within a safe limit. Thermal design power (TDP) is a widely used design time estimate of the upper bound on power consumption that ensures thermal safety. TDP is a conservative estimate with the assumption that the chip is operating at worst case voltage and frequency levels, and workloads [4]. An efficient alternative to TDP is Thermal Safe Power (TSP), a variable run-time estimate of upper bound on power, calculated dynamically as a function of number of active cores and their spatial alignment [5]. Considering the mutual thermal effect of neighboring active and inactive cores, application mapping can determine the maximum utilizable power budget. Using TSP instead of TDP presents a scenario where utilizing the same power budget can result in different temperatures for different mappings.

State-of-the-art application mapping policies that handle thread-to-core allocation prefer i) a contiguous mapping scheme to minimize inter-core communication latency [6] and ii) the set of cores chosen for mapping an application to be regular geometric structures such that potential incoming applications do not suffer with a dispersed set of available cores and to avoid inter-application congestion. At lower technology nodes, conventional mapping approaches result in denser mappings which accumulate temperature faster, limiting the maximum utilizable power budget within safe thermal limits. In contrast, sparse mappings that compromise on contiguity can provide higher utilizable power budget before reaching the critical temperature. With shrinking transistor sizes and increasing power densities, mapping policies have to be adapted to maximize performance through efficient power budget utilization.

### 1.1 Implications of Mapping on Power Density

We demonstrate the effect of mapping on temperature accumulation through the following example. Consider an application with 8 threads to be run on a 16-core system, arranged in a  $4 \times 4$  mesh, as shown in Figure 1 (a). Each core in this system is modeled based on Niagara-2 architecture at 22nm technology, power and performance metrics are extracted from the analytical framework of Lumos [7]. The application is executed on our in-house cycle accurate simulator (see Section 4). All the 8 threads of the application are mapped in a dense manner on 8 cores and the remaining cores are power gated. Each thread of the application consumes an average power of 5.95 W, resulting in total active power of 47.6 W ( $5.95 \times 8$ ). We assume the critical temperature of the chip to be 80 °C. The final steady state temperature of the system after running the application for 100 ms is shown in Figure 1 (a). This mapping configuration has resulted in the chip reaching its critical temperature after consuming 47.6 W of power. For the same application and set up, consider

- Anil Kanduri, Mohammad-Hashem Haghbayan and Pasi Liljeberg are with University of Turku, 20500 Turku, Finland. Muhammad Shafique and Axel Jantsch are with TU Wien, 1040 Vienna, Austria. Amir M. Rahmani is with University of California, Irvine, CA 92617 USA.  
E-mail: spakan@utu.fi, mohhag@utu.fi, pakrli@utu.fi, muhammad.shafique@tuwien.ac.at, axel.jantsch@tuwien.ac.at, amirr1@uci.edu
- Amir M. Rahmani is also with TU Wien, 1040 Vienna, Austria.

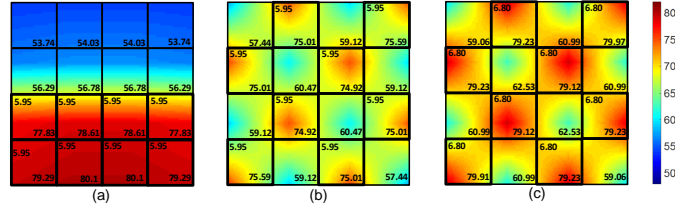


Fig. 1. Effect of mapping on temperature and power budget. Each block represents a tile in a 16-core system arranged in 4x4 mesh. Top left corner of a tile shows power consumption in watts and bottom right corner shows final steady state temperature (a). Dense mapping, (b) Sparse mapping that consumes same power as dense mapping at lower temperature, (c). Sparse mapping that utilizes more power than dense mapping at same temperature

the mapping configuration in Figure 1 (b). Here, the same application is mapped sparsely, such that each active core has an idle neighbor. As the neighboring cores are relatively cool, temperature accumulation of active cores is slowed down. With the same amount of power consumption, the sparse mapping has resulted in almost 4°C lower final steady state temperatures, showing the effect of mapping on temperature.

With critical temperature set to 80°C, available thermal headroom can be utilized i) to power up more cores, ii) to run active cores at a higher frequency i.e., boosting, iii) to lower overall on-chip temperatures. This allows both per-application performance to be improved by boosting, and per-chip throughput by activating more cores. Consider the final steady state temperature of the same application with sparse mapping configuration as discussed earlier in Figure 1 (c), which is boosted within thermal safe limits. With the sparse mapping configuration, the chip has reached its critical temperature after each core consuming 6.8 W of power. The total power that can be consumed with sparse mapping ( $6.8 \times 8 = 54.4$  W) is higher than that of the dense mapping, yielding a 6.8 W of surplus power budget. The sparse mapping configuration in Figure 1 (b) has 5.96 % lower temperature and the one in Figure 1 (c) has 14.2 % of surplus power budget, over the dense mapping configuration. Assuming that each thread consumes 5.95 W of power as in Figure 1 (a), the surplus budget can be used to activate at least one more core on the chip, increasing the utilization to 12.5%, thus mitigating dark silicon. We establish that the performance and utilization of applications can be improved by considering sparsity in application mapping, which provides power headroom for activating more cores and/or thermal headroom for boosting the frequency of active cores. We refer to the sparse mapping approach which aligns cool inactive cores among hot active cores as Dark silicon patterning [8].

## 1.2 Impact of Dark Silicon Aware Mapping and Boosting on Performance

The thermal headroom provided with dark silicon patterning can be leveraged to boost the frequency over short bursts of time to improve performance while minimizing the likelihood of thermal violation. We refer to this technique as boosting, distinguishing from conventional frequency scaling for performance enhancement [9]. We define boosting period as the time taken for a power violation to translate into a thermal violation. Boosting period largely depends on available thermal headroom, on-chip temperatures, the rate at which temperature accumulates and frequency to be boosted to. In

addition to these factors, power-performance characteristics of an application and the combination of other concurrent applications also effect the boosting period. With hot active cores neighbored by cool inactive cores, dark silicon aware mapping is likely to result in higher thermal headroom and importantly a lower rate at which temperature accumulates with power. Dark silicon aware mapping thus improves the performance gains from boosting, which necessitates considering mapping and boosting together at run-time. We provide insights into the compound effect of dark silicon aware mapping on boosting with the following example.

**Motivational Example:** Consider a synthetic application of 8 threads, executed on a similar set up mentioned in Section 1.1 with both dense and sparse mappings. Figure 2 shows the rate at which transient temperature accumulates for the application with dense and sparse mappings over 100 ms of execution time. The thermal profile was extracted using HotSpot 6.0 [10], with its default configuration. In both cases, the application started execution with same initial temperature of 70 °C. However, temperature accumulates faster with dense mapping, due to each active core's temperature affecting its neighboring core's temperature. In contrast, temperature accumulates relatively slower with sparse mapping, due to the interleaved inactive cores reducing the compounded effect of temperature among active cores. After 100 ms of execution, there is about 2 °C of difference between the temperatures of both the mappings. Assume a performance surge requirement at 100 ms, which is met by boosting the frequency of active cores from 3 GHz to 3.75 GHz. The sudden increase in frequency causes increase in power and temperature, as shown in Figure 3. At time  $t=100$  ms, the dense mapping is operating at about 76 °C, while sparse mapping is operating at 74 °C. With dense mapping being at a relatively higher temperature when boosting is invoked, subsequent temperature accumulation towards critical temperature is faster, lowering the period of boosting. The dense mapping reaches the critical temperature of 80 °C at 131 ms, after sustaining a boosting period of 31 ms. With potential thermal violation, the frequency is reduced back to 3 GHz to reduce the on-chip temperature. The application finished execution eventually after 189 ms. On the other hand, the sparse mapping, which is already at a relatively lower temperature when boosting is invoked, sustains a longer period of over-boosting of 75 ms (i.e. an increase of about 2x in the boosting period). As a result, there is a significant performance gain during the period of boosting and the application finishes execution by 175 ms. The sparse mapping configuration could sustain 41% longer period of over-boosting which reflects in 8% of performance

as compared to the dense mapping. Further, the sparse mapping remained well below the critical temperature of 80 °C, giving enough thermal headroom for either longer periods and/or higher levels of over-boosting. The longer period of boosting in case of sparse mapping can be attributed to both slower accumulation of temperatures and essentially the cores operating at relatively temperatures at the time boosting is invoked. Our analysis demonstrates that mapping applications sparsely at first which is then followed by boosting, allows to extract better performance. Further, this avoids frequent oscillation between frequency up and down scaling that densely mapped applications suffer with, due to faster temperature accumulation.

### 1.3 Our Contributions

In our previous work [8], we proposed dark silicon patterning, a run-time mapping technique to balance the heat distribution evenly across the chip to increase power budgets. We used the surplus power budget to service more applications, improving the overall chip throughput. Dark silicon patterning also reduces the effect of temperature accumulation among neighboring active cores, leaving thermal headroom for active cores to consume relatively more power before reaching critical temperatures. In the context of this work, we focus on improving both per-application latency along with per-chip throughput, within thermal constraints by utilizing the thermal headroom from dark silicon patterning to boost the frequency of active cores upon performance requirements. We propose *adBoost*, a thermal aware adaptive boosting technique that considers both mapping and boosting decisions together for creating enough thermal headroom for sustainable periods of boosting. We split our approach into 2 steps viz., 1) dark silicon patterning based mapping and power management, followed by 2) thermal aware adaptive boosting. We augment our previous run-time controller for power management that supports dynamic application mapping and power allocation with the proposed boosting controller, *adBoost*. The *adBoost* controller receives feed-back on instantaneous temperature from the system to make decisions frequency scaling decisions upon performance requirements for efficient utilization of available thermal headroom obtained with patterning. Existing works on power budgeting [11], dark silicon aware power management [12] [13], dark silicon aware interconnect designs [14] [15], frequency boosting [16] have considered each of their focus aspects and scenarios independently, without considering the effect of each technique on subsequent inter-related aspects. In our work, we integrate application mapping - to maximize power budgets and minimize on-chip temperatures, power allocation - to efficiently utilize the surplus budget provided with mapping, and boosting - to leverage thermal headroom provided with sparse mappings, thereby improving performance while honoring power and thermal constraints.

**Our contributions are as follows:**

- A run-time mapping technique for dark silicon patterning to evenly distribute heat across the chip that improves power budget.
- Design, analysis and implementation of boosting mechanism to accelerate performance through fre-

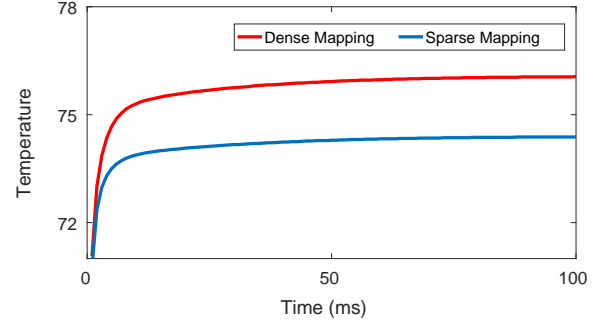


Fig. 2. Effect of mapping on temperature accumulation

quency scaling subject to availability of thermal headroom.

- Controllers with power and temperature feed-back for efficient allocation of surplus power budget to improve per-chip throughput and/or per-application latency.
- Design and implementation of a system manager that integrates the run-time mapping unit - for surplus power budget, power management - for power allocation decisions and boosting mechanism - for utilizing the thermal headroom available.

## 2 DARK SILICON AWARE MAPPING

Applications with high performance requirements functioning at full throttle lead to high power consumption and on-chip temperatures. Dynamic thermal management techniques such as clock and power gating, voltage and frequency down scaling, increased fan speed etc., are triggered to limit the power consumption and subsequently lower the on-chip temperatures over a period of time. Such reactive techniques manage thermal violations, while performance of the chip is degraded. We consider run-time mapping by patterning the dark cores along with working cores to be a better choice as opposed to the reactive strategies - for even distribution of heat and the surplus budget from patterning can be utilized for performance boosting. This requires a dark silicon aware mapping solution, which considers spatial alignment of active and dark cores with an objective of providing surplus budgets. In the following sections, we elaborate on design of our mapping strategy.

### 2.1 Inter-application Sparsity

Application mapping techniques intend to find suitable cores to bind tasks of an application, while optimizing one or more parameters among communication latency, on-chip traffic, geometric structure - contiguity and dispersion. While conventional mapping approaches prefer contiguity for performance, we propose to avoid contiguity for even heat distribution and to avoid possibility of hot spots. Two factors that distinguishes a sparse mapping from a contiguous mapping are spatial distribution of applications and sparsity among tasks of each individual application. Consider the heat dissipated by a core  $C_i$ , which is a 3-tuple  $(P_i, T_n, T_{amb})$ , where  $P_i$  is the power dissipated by the core,  $T_n$  is the temperature of neighbouring cores and  $T_{amb}$  is the ambient temperature. Contiguous mappings accumulate heat faster with neighboring cores' temperature effecting the active cores (i.e., higher  $T_n$ ), where as the sparse mapping

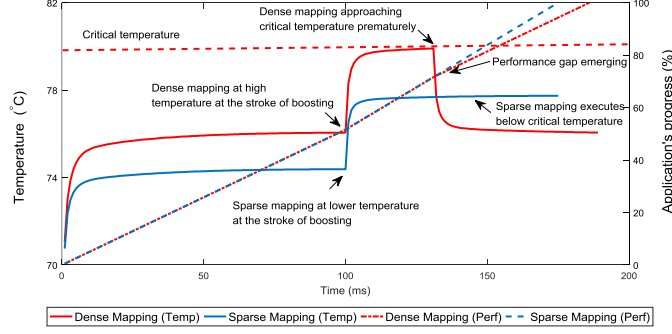


Fig. 3. Effect of mapping on boosting and performance. Thermal profile and progress of application's execution are shown for dense and sparse mappings.

we propose benefits from lower temperature of inactive neighboring cores (i.e., lower  $T_n$ ).

Concurrent applications running on a many-core system effect temperatures of neighboring applications' cores and lead to potential hot spots. We demonstrate the effect of inter-application spatial alignment on temperature and power budget through another example, presented in Figure 4. Three applications App1, App2 and App3 with 12, 8 and 4 tasks respectively are assumed to be running on the system. Conventional mapping approaches map all the three applications contiguously to avoid dispersion and limit packet latency. Thermal profiles of these 3 applications on a 144-core system with a contiguous mapping is shown in Figure 4(a). The power budget (TSP) of this mapping configuration computed using TSP library [5] is 66W. Thermal profiles of the same applications running on the same system with sparsely distributed mapping configuration (at both inter-application and intra-application levels) is shown in Figure 4(b). This mapping configuration provides a power budget (TSP) of 74.6W, as calculated by TSP library. An improvement of 8.6W in power budget can be observed for the spread-out, patterned mapping as opposed to tightly packed and contiguous mapping. Although contiguous mapping avoids dispersion, it leads to poor thermal profile of the chip due to prorogation of heat among neighboring applications and tasks. Assuming that some of the application require performance surges at runtime, boosting them for sustained periods depends on both neighboring active cores within the applications as well as other (spatially closer) concurrently running applications. In view of these factors, we split our mapping approach into two phases viz., selecting a region that is spatially dispersed from current set of applications running on chip and mapping tasks of the application sparsely such that active and inactive cores are patterned in the selected region.

**Latency Vs Performance Gain:** The contiguous mapping presented in Figure 4(a) has a budget of 66W, which is used to power 22 cores, giving a per-core budget of 3W. Assuming this per-core budget of 3W as the baseline, the sparse mapping configuration in Figure 4(b) could power  $74.6/3 = 24.7$  cores. The surplus budget we could gain through patterning based mapping could be used to power up 2.7 more cores, reducing dark silicon by 12.2%. However, the sparse mapping configurations also lead to dispersion among applications and increased inter-task communication latency within an application. The average packet latency in

each of the patterned applications App1, App2 and App3 is 1%, 3.5% and 3.8% (shown in Figure 4(b)) more than that of the contiguous mappings respectively. Yet, the power budget gains (about 12%) and thermal headroom for potentially boosting the applications outweighs the odds of latency induced by distributed mapping configurations. It is also to be noted that the inactive cores are the inevitable dark cores - instead of leaving them out naively, we use them to balance out heat distribution and gain power budget.

## 2.2 First Node Preferences

We split our mapping approach into two phases - first node selection, followed task allocation, as proposed in [6]. First node selection corresponds to choosing a free core on the chip around which the tasks of an application can subsequently be mapped to fulfill power, performance and/or latency objectives. From our analysis presented in Figure 4, we prefer nodes that are far from currently active cores as suitable candidates for first node, in order to minimize the affect of temperature among active cores. Precisely, our objective is to find a *free* and *sparse* first node which has: i) enough number of free nodes around it to allocate tasks of the incoming application with minimal internal congestion and ii) minimal effect of temperature due to neighboring (active) cores.

**Free First Node:** We modify the first node selection strategy MapPro, presented in [17] in order to meet the first objective of finding a *free* first node. MapPro uses Vicinity Counter (VC) as a metric to represent number of free cores around a given node within a given region. The node with a relatively higher VC value is preferred as the first node such that it can fit all the tasks of an application within the region around it. It should be noted that VC of node is relative to the size of the region i.e., the same node has different VC values over different regions. For example, a node within a  $5 \times 5$  (25 cores) sized region of free cores has 24 free cores around it, whereas the same node within a  $3 \times 3$  (9 cores) sized region has 8 free cores around it. The VC parameter also represents internal congestion around a given node by considering the number of occupied nodes around it - intuitively reflecting the likeliness of congestion, if mapped around that node. Implementation details of MapPro are elaborated in [17].

**Sparse First Node:** To meet our second objective of minimizing mutual heat effect among active cores of concurrently running applications, we prefer nodes that are farther from (if any) currently active cores as candidates



for first node. The effect of temperature of an active core on neighboring cores (and all cores on the chip) gradually decreases as we move towards physically farther cores. Inspired by the surface tension phenomenon [18], where energy distribution of a flat surface turns into a curved surface with applied surface pressure, we model the effect of temperature of an active core on other cores of the chip in a similar fashion. The temperature effect of every active core decreases exponentially, but not linearly, with distance from the active (hot) core. In other words, the greater the distance from an active core, the lesser the effect of heat from it. In order to minimize this heating effect from active cores of other applications, we prioritize nodes that are as far as possible from such active cores. We define Distance Factor (DF) to quantify the effect of temperature of an active cores on other cores on the chip, as follows: **Definition:** Distance Factor,  $DF_{i,j}$ , for a node located at  $(i,j)$  is the weighted sum of impact of distance from all the other occupied nodes located at  $(x,y)$  such that  $(x,y) \in Mesh$ .

$$DF_{i,j} = \sum Wn_{i,j} \times (e^{-\alpha(d_{ij}-xy)}) \quad (1)$$

where  $Wn_{i,j}$  is the weight of node  $n_{i,j}$ ,  $d_{ij}-xy$  is distance from nodes located at  $(i,j)$  and  $(x,y)$  and  $\alpha$  is the mesh size. We augment MapPro's choice of *free first node* with the *sparse first node* to find a suitable candidate for the *first node*.

The algorithmic flow of first node selection is shown in Algorithm 1 which is inspired by our previous work presented in [17]. Whenever a new application enters the system, the average power budget required for its execution is estimated. Subject to the availability of power budget, the application is serviced by finding a suitable first node based on VC and DF values (lines 1-3). Number of tasks of the application determines the region size over which VC values will be computed. The node with maximum VC value,  $maxVC$ , among all nodes is chosen as the first node, and thus also selecting the polygon (preferably square) centered at  $maxVC$ . The chosen first node and the application are passed to patterning phase (line 4). Mapping the tasks based on patterning is explained in the following section and Algorithm 2. Mapping a new application alters the VC and DF values for all the nodes on the chip, which need to be updated. After mapping an application, we calculate the new VC and DF values for all the nodes, to be used for choosing the first node for subsequent applications. VC values for different sizes of regions are updated (lines 6-23) to determine the new  $maxVC$  node. Pro-active VC and DF calculation eases first node selection for next incoming application (based on number of tasks of the application) and reduces first node selection overhead. Conflict between nodes having same VC value is resolved by considering corresponding DF values (lines 16-18). Status of the nodes running an application is reset to *free*, once the application leaves after finishing its execution. The VC and DF values are once again updated as per the altered mapping configuration.

### 2.3 Dark Silicon Patterning

We prefer mapping tasks of the application in a sparse pattern in order to reduce the probability of heat accumulation that could potentially lead to hot-spots. We define the likeliness of an active core effecting the temperature of

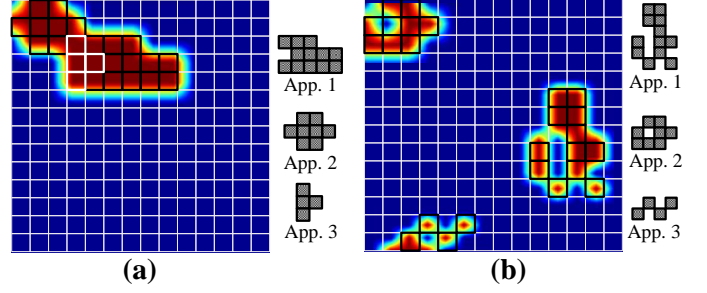


Fig. 4. Effect of inter-application mapping on temperature accumulation. (a) Contiguous mapping of 3 applications, with their individual mapping configurations. (b) Sparse mapping of 3 applications, with their individual mapping configurations and percentage of packet latency penalty for mapping sparsely, normalized against contiguous mappings.

its neighbors through its sparsity. Sparsity of a node  $n_{i,j}$  represents the number of free nodes that are neighboring it in all four cardinal directions (North, East, West, South). The Sparsity Factor ( $SF_{i,j}$ ) for a node  $n_{i,j}$  located at  $(i,j)$  is expressed as:

$$SF_{i,j} = \sum_{i'=1}^4 \sum_{j'=1}^4 F(i+i', j+j') \quad (2)$$

where  $i' = [0, 1, 1, -1]$ ,  $j' = [-1, 0, 1, 1]$ .  $F(i,j)$  denotes if a node located at  $(i,j)$  is free or not, such that

$$F(i,j) = \begin{cases} 1 & \text{if } n_{i,j} \text{ is unoccupied} \\ 0 & \text{if } n_{i,j} \text{ is occupied} \end{cases}$$

Following the first node selection, we build a polygon  $P$  (preferably a square), which is a set of all the un-occupied nodes around the selected first node. Intuitively, mapping the tasks of an application in this region  $P$  would have relatively lesser effect of temperature from other applications. Within the selected region, we prioritize the nodes that are more sparse to map tasks of the application. As a result, sparse nodes are occupied first, leaving out the denser nodes to be patterned in a way that they provide necessary cooling effect needed by their active neighbors. While mapping, we sort the tasks of the incoming application as per their communication volume. We choose the task with highest communication volume and map it on to the first node, the most central and sparse node such that relative proximity with other inter-communicating tasks is higher. We proceed to the task with next highest communication volume by sorting all the remaining tasks as per their total communication volume and map it onto the node with the highest  $SF$  among the nodes in the selected region  $P$ . We continue a similar order of tasks - based on communication volume and similar order of nodes - based on sparsity, such that tasks with higher communication gets mapped onto nodes with higher sparsity, until all tasks of the application are mapped. Mapping tasks with higher communication volume in the order of sparsity allows corresponding communicating tasks to be in proximity of the high communication volume tasks, minimizing the weighted Manhattan distance for inter-task communication. Throughout the mapping procedure, dense nodes have least priority, which would remain un-occupied amongst the other occupied nodes of the region. This provides the necessary cooling effect for improved power budget utilization.

---

**Algorithm 1** The mapping algorithm

**Inputs:** *newApp*: New application, *budget*: Available power budget;  
**Outputs:** *Q*: Mapping;  
**Constants:** *M*: Size of the mesh, *groups*: Number of square groups =  $\lceil (\sqrt{M} - 1)/2 \rceil$ , *maxRadius*: Maximum radius of the square  $((\sqrt{M} - 1)/2)$ ,  $\alpha$ : Mesh size parameter  $\sqrt{M}$ , *P<sub>avg</sub>*: Average power consumption per node;  
**Global Variables:** *VC*: Vicinity Count of a node, *maxVC*: Node with the maximum VC, *firstNode*: Selected first node for mapping, *DF*: Distance factor;

**Body:**

```

1: appPredictedPower  $\leftarrow |newApp| \times P_{avg}$ ;
2: if appPredictedPower  $\leq budget$  then
3:   firstNode  $\leftarrow maxVC_{\lceil (\sqrt{appSize}-1)/2 \rceil}$ 
4:   Q  $\leftarrow pattern(firstNode, newApp)$ ;
5:   //Updating VC and DF values after mapping
6:   for each nxy  $\in newApp$  do
7:     for each core nij located in Row i and Column j do
8:       r' = maximum( $|i - x|, |j - y|$ );
9:       DFij =  $e^{-\alpha r'}$ ;
10:      for r = 1 to maxRadius do
11:        if r - r'  $\geq 0$  then
12:          VCijr = r - r';
13:          if VCijr > maxVCr then
14:            maxVCr  $\leftarrow VC_{ij}^r$ ;
15:          else
16:            if VCijr = maxVCr and DFij > DFmaxVCr then
17:              maxVCr  $\leftarrow VC_{ij}^r$ ;

```

---

The algorithmic flow of patterning is shown in Algorithm 2. Mapping starts at *firstNode*, selected from Algorithm 1 by choosing square region (*Square<sub>firstNode</sub><sup>size</sup>*) such that application (*App*) can fit within this region (line1). Tasks (*Tasks*) of the application are sorted as per their communication volume (line 2). The most expensive task as per communication is mapped onto the node with maximum sparsity factor (*SF*), *maxSF* (lines 5-6). The mapped task is removed from the list of tasks to be mapped and the mapped node is removed from the list available nodes in the square (lines 7-8). This procedure is repeated until all the tasks of the application are mapped. The *SF* value for nodes in selected square changes with every occupied and thus new *maxSF* is computed for every unmapped task.

### 3 SYSTEM DESIGN

Our system design consists of three phases - i) run-time mapping that maximizes utilizable power budget, ii) power controller that utilizes surplus power budget to activate more cores (if possible) and iii) boosting controller that increases frequency beyond the base frequency subject to thermal safety. The hierarchical view of the system architecture is shown in Figure 5. We first present the work flow of the framework, followed by details on individual modules in the following sections.

**Work flow:** The run-time mapping unit (RMU) is invoked upon on arrival of new applications and is responsible for task-to-core allocation of the incoming applications. The RMU chooses appropriate first node and subsequently finalizes task-to-core mapping. The TSP Calculator estimates safe upper limit for power consumption, based on

---

**Algorithm 2** Patterning

**Inputs:** *App*: Application, *firstNode*: Selected First Node.  
**Global Variables:** *S* Selected square, *SF*: Sparsity Factor for each node in a square, *maxSF*: Node with the maximum SF in a square, *currentNode*: Node onto which current task is being mapped;  
**Global Constants:** *Tasks*: Vector of tasks of the application  
*App, size*: Radius of square close to size of *App*

**Body:**

```

1: S  $\leftarrow Square_{firstNode}^{size}$ 
2: Tasks = sort(App);
3: while App  $\neq \emptyset$  do
4:   for each ti  $\in Tasks$  do
5:     currentNode  $\leftarrow maxSF$ ;
6:     map (ti)  $\rightarrow maxSF$ ;
7:     App = ti;
8:     S = currentNode;

```

---

number of simultaneously active cores using the mapping configuration. Once a new application is mapped and/or a currently running application exited the system, upper bound on power is re-evaluated using the TSP calculator. The upper bound is set until further variation in mapping due subsequent entry/exit of an application. The power controller handles power actuation decisions by monitoring current power consumption and comparing against safe limit and communicates the DVFS and power gating settings to the central manager. Power controller is invoked over the parametric power epoch *E<sub>power</sub>*. The boost controller makes boosting decisions upon performance requirements of applications and sends the corresponding frequency level and application information to the central manager. The boost controller is invoked over the boosting epoch *E<sub>boost</sub>*. It is to be noted that power controller decisions are based on power headroom available whereas boost controller decisions are based on thermal headroom. DVFS settings from the power controller are proportional to the surplus power budget available. Boost controller's decisions rely on temperature feedback, irrespective of power headroom. We set *E<sub>power</sub>* as  $4 \times E_{boost}$ , such that boosting decisions are made conservatively to avoid thermal violation. This minimizes the oscillation of voltage and frequency scaling decisions among power and boost controllers. Both power controller and boost controller act independently over their respective epochs and communicate their decisions to the central manager. The central manager enforces voltage and frequency levels and power gating settings as per the decisions made by power controller and the boost controller by arbitrating between them. Figure 6 shows modular work flow of different control units of our approach. Individual modules of the system are detailed in following sections.

#### 3.1 Run-time Mapping Unit

Applications arriving dynamically are serviced by the Run-time Mapping Unit (RMU) by finding active cores around a suitable first node, followed by task-to-core mapping. The RMU presented in Figure 5 uses the pro-active approach, MapPro [17], for first node selection and dark silicon patterning based mapping (PAT) [8] for application mapping. The objectives and algorithmic flow of patterning based

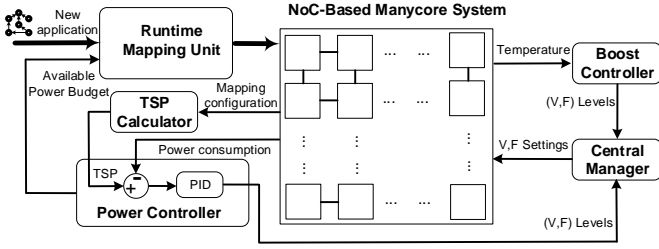


Fig. 5. System architecture

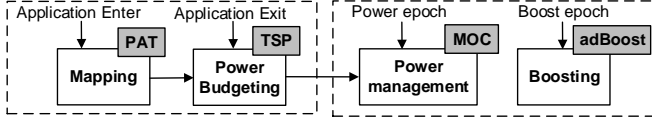


Fig. 6. Workflow of proposed framework

mapping have been elaborated in Section 2.3. More details on first node selection and dark silicon patterning based mapping are elaborated in our previous works [17] and [8] respectively. Each task of an incoming application holds a computation factor and a communication factor, indicating their relative amount of computation and inter-task communication volumes. RMU estimates the average power consumption of an incoming application as accumulated sum of power consumption of task-per-core, under nominal conditions. RMU receives the available power budget from the power controller, and compares it against the estimated average power consumption of the incoming application. RMU decides to map the new application with sufficient power budget being available, while the application waits in the queue in case of restricted power budget. A possibility of avoiding longer application waiting time is by scheduling a sub-set of independent tasks that would fit within available power budget. Although, such power allocation requires application level information on inter-task dependencies, and availability and choice of appropriate set of independent tasks that would fit within the limited power budgets.

### 3.2 Power Controller

The power controller is responsible for power management and allocation decisions, as shown Figure 5. We use the light weight open source TSP library to calculate appropriate power budget to be allocated, as a function of number of active cores [11]. The TSP Calculator (in Figure 5) receives current mapping configuration to dynamically calculate the thermal safe power (TSP), the safe upper bound. TSP is evaluated whenever there is a change in mapping configuration i.e., upon arrival of new application or exit of a currently running application. The power controller estimates available power budget, calculated as the difference between the monitored power consumption and the safe upper bound determined by TSP Calculator. We use a PID controller to determine voltage and frequency level settings of cores, such that they are proportional to the available power budget, estimated by comparing instantaneous power consumption with TSP. In case of a power violation, i.e., power consumption exceeding TSP, the power controller scales down the (V,F) levels proportionally enough to reduce the power consumption to a safe limit. With power constraints

honored i.e., power consumption below TSP, the power controller allocates the surplus budget in two possible ways, i.e., either to map new applications (if any), or to throttle currently running applications. When there is a new application waiting in the queue for execution, the surplus budget is used to map the new application, provided that the available power budget is enough to accommodate the new application. With no new application arrival or insufficient amount surplus budget to fit the new application, the surplus budget is used to throttle currently running applications. In this scenario, the power controller scales up the (V,F) levels of the running application(s), corresponding to the amount of power budget available. In case of throttling current applications, the power controller also considers applications' priority level and network characteristics to determine suitable candidates among concurrently applications to actuate DVFS. In case of upscaling or downscaling the (V,F) levels, voltage and frequency levels determined by the power controller are received by the central manager, which enforces DVFS settings for power actuation. Algorithmic implementation, power controller's decisions on power allocation and choice of suitable candidates for DVFS are detailed in our previous work [19].

**Dark and Dim Silicon Considerations:** We present a scenario demonstrating varying DVFS levels using the power controller strategy under dynamic workloads in Figure 7. We assume dynamic entry and exit of applications that create a variable workload that lead to both dark silicon and dim silicon scenarios. Figure 7 (a)-(d) shows subsequent entry of four applications with 7, 4, 4 and 4 tasks respectively. In Figure 7 (a), one application (App1) of 7 tasks being mapped at high (V,F) levels. As another application (App2) of 4 tasks enters, power budget is estimated and has to be shared among the two applications. This results in downscaling the App1 to accommodate both App1 and App2 within the safe limits. As new applications continue to arrive, the power budget is estimated and is allocated accordingly using DVFS. Figure 7 (d) shows the scenario where there are 4 concurrent applications are running on the chip. Available power budget has to be shared among relatively a higher number of applications, which leads to throttling down certain applications, subjective to their priority requirements and network characteristics. With a higher number of concurrent applications, the system execution is inclined towards dim silicon, whereas relatively smaller number of concurrent applications presents dark silicon scenario. Power budget gains from patterning the applications is higher with dark silicon, as compared to a dim silicon baseline.

### 3.3 adBoost Controller

The boost controller is responsible for boosting the frequency of active cores, subject to application requirements and thermal safety. The controller is invoked over an epoch  $E_{boost}$ . Transient and steady state per-core temperature values are calculated using HotSpot 6.0 with its default configuration over every epoch  $E_{boost}$ . Trace file containing per-core power values is obtained from the power controller. In every epoch, it monitors on-chip per-core temperatures to estimate the thermal headroom  $\Delta T$ , calculated as the difference between critical temperature ( $T_{crit}$ ) and maximum temperature ( $T_{max}$ )

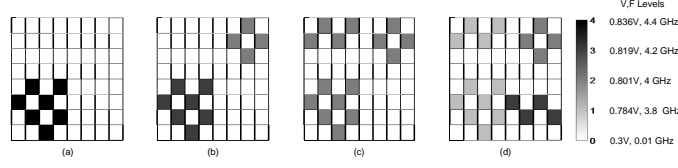


Fig. 7. Varying DVFS levels using the power controller with dynamic workloads. (a)-(d): Mapping and DVFS level of applications, ranging from 1 to 4 concurrent applications of 7, 4, 4 and 4 tasks per application respectively.

among active cores. When an application requests for performance improvement, the controller evaluates  $\Delta T$  to handle boosting process. The algorithmic flow of boosting is presented in Algorithm 3.

$\Delta T > 0$  When there is enough thermal headroom ( $T_{crit} > T_{max}$ ), the controller can boost the frequency upon an application's request for performance. With a specific application's request boosting, the controller chooses to increase the frequency of cores running the application by one *level*. Frequency of each core among the set of cores running the application is boosted within in one epoch (line 9, Algorithm 3). If the performance requests still persist, frequency of chosen cores is boosted progressively in steps of one *level* during the subsequent epochs, provided that there is thermal headroom. Boosting stops upon reaching the maximum possible frequency ( $F_{max}$ ) through successive steps of boosting up to  $B$  levels. When more than one application makes a request for boosting, the application running on cores with relatively lower temperatures is chosen.

$\Delta T < 0$  When there is no thermal headroom ( $T_{crit} < T_{max}$ ), the controller has to lower the frequency of active cores and any applications' requests for performance cannot be met. The controller chooses application running on cores with maximum temperature (i.e., hottest cores on the chip) to downscale the frequency by one *level*. Lowering the frequency continues progressively in steps of one *level* over the subsequent epochs, until the temperature violation is under control. Similar to the boosting, frequency of each core among the set of cores running the chosen application is scaled down (line 14, Algorithm 3). Once there is enough thermal headroom, the boosting process resumes normally upon application requirements. Increase in frequency due to boosting in one epoch can increase the temperature over the following epochs. To avoid thermal violation, the boost controller always evaluates thermal headroom in every epoch to determine the possibility of boosting. Figure 8 shows the control flow of the boost controller subject to the thermal headroom available. The number of epochs over which temperature does not reach  $T_{crit}$  with boosting represents period of boosting. Intuitively, a dense mapping with temperature of neighboring cores affecting mutually would approach the  $T_{crit}$  sooner when compared to a sparse mapping. The critical temperature, epoch of boosting ( $E_{boost}$ ), number of boosting frequency levels ( $B$ ) and the size of each level (*level*) are parameterizable. Figure 9 shows the rate at which temperature accumulates upon increasing the frequency from 4 GHz to 4.2 GHz. We observe that temperature accumulation is steeper in the initial phases and eventually saturates over a period of 5 ms. We set our boosting epoch length to half of the time period over which temperature would saturate, for a 200

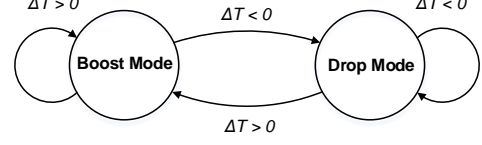


Fig. 8. Control flow of boosting.  $\Delta T$  is the thermal headroom. Boost and Drop modes represents up and down scaling of frequency

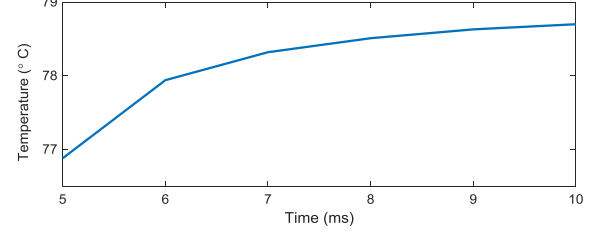


Fig. 9. Rate of temperature accumulation

MHz rise in frequency, for a conservative and fine-grain control. We boost one application within one boosting epoch, and conservatively boost only by 200 MHz, also by prioritizing relatively cooler cores, in order to minimize the probability of potential thermal violation with boosting.

### Algorithm 3 Boosting

**Inputs:**  $Temp[]$  : Per-core temperature vector,  $APPS$ ,  $APPS_{boost}$  : Applications - currently running on the chip and requested for boosting  
**Global Variables:**  $T_{max}$  Maximum temperature on the chip,  $F$  : Frequency,  $c_i$  : Core  $i$   
**Global Constants:**  $T_{crit}$  : Critical temperature,  $E_{boost}$  : Boosting epoch, *level* : Step size of boosting frequency,  $F_{max}$ ,  $F_{min}$  : Maximum and minimum operating frequencies

#### Body:

```

1: for every  $E_{boost}$  do
2:    $Temp[] \leftarrow \text{updateTemperature}()$ ;
3:    $T_{max} \leftarrow \max(Temp[])$ ;
4:    $\Delta T \leftarrow T_{crit} - T_{max}$ ;
5:   if  $\Delta T > 0$  then
6:     for  $APPS_{boost}$  do
7:        $targetApp \leftarrow \minTemp(APPS_{boost})$ ;
8:       if  $targetApp \rightarrow F \neq F_{max}$  then
9:         for  $c_i \in targetApp$  do
10:          Boost( $c_i, level$ );
11:     else
12:       for  $APPS$  do
13:          $targetApp \leftarrow \maxTemp(APPS_{boost})$ ;
14:         if  $targetApp \rightarrow F \neq F_{min}$  then
15:           for  $c_i \in targetApp$  do
16:            Down( $c_i, level$ );

```

## 4 EVALUATION

The system model and specifications are summarized in Table 1 and are elaborated as follows. **Application Model:**



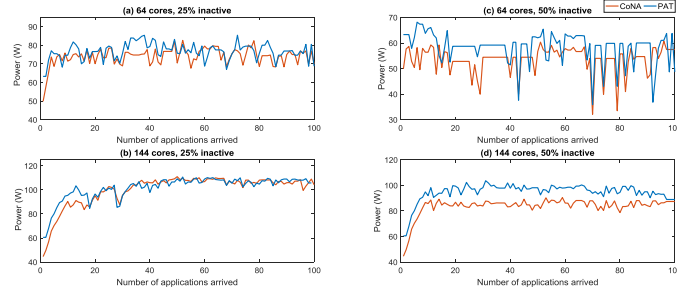


Fig. 10. Power budgets of CoNA and PAT with varying number of active cores. Power budget is calculated using TSP calculator upon arrival/exit of an application, over 100 applications that enter and leave the system. (a). Power budgets for a 64-core system with 25% inactive cores. (b). Power budgets for a 144-core system with 25% inactive cores. (c). Power budgets for a 64-core system with 50% inactive cores. (d). Power budgets for a 144-core system with 50% inactive cores.

We evaluated the proposed mapping and boosting approach over applications that are modeled as task graphs, generated using TGG [20]. Each application is a directed graph of tasks, where each task holds information on computation and communication with other tasks, and priority factor to indicate need for performance surges. The communication volume between any two given tasks is considered while mapping tasks to cores, by sorting all the tasks as per their communication volume with a specific task to be mapped. Specifically with sparse mapping algorithm, this avoids mapping tasks with higher communication volume on farther cores.

**System Model:** We simulated traffic patterns of applications using our in-house cycle-accurate many-core platform implemented in SystemC. The specifications of Niagara-2 like in-order cores obtained from McPAT [21] are used as the baseline for processing elements. The communication network infrastructure between processing elements is provided by a pruned version of Noxim [22] that uses mesh topology and XY routing. Parameters related to technology node scaling are extracted from Lumos framework [7], an open source framework which quantifies power-performance characteristics of many-core systems with technology node scaling. We used TSP library [5] to calculate the Thermal Safe Power. In the many-core platform we implemented, (overview as in Figure 5), a random sequence of applications enter the system and are buffered into a FIFO. A Central Manager (*CM*) assigned to node  $n_{(0,0)}$  of the mesh in the many-core platform implements all the required system resource control and allocation operations including run-time mapping, power budget calculation, power controller and boost controller functions. Each of the control routines are implemented as software modules, requiring no additional specialized hardware. Our framework is designed to run as a user-space process which can communicate with the operating system to enforce resource management decisions. Applications are serviced in a first-come-first-serve policy, subject to availability of enough power budget. A suitable first node for mapping the incoming application is chosen by the (*CM*), followed by mapping all the remaining tasks of the application.

Assuming a base frequency of 4 GHz in our evaluation, we set the step size of each boosting frequency level to 5% rise on the base frequency, resulting in a step size of 200 MHz. Figure 9 shows the rate at which temperature accumulates upon boosting from 4GHz to 4.2GHz. Steep rise followed

TABLE 1  
Simulation Details

Parameter	Specification
Application Model	Synthetic task graph, 4-18 tasks [20]
Core Model	Niagara-2 like, in-order
Power Model	McPAT [21] and Lumos [7], 16nm
Thermal Model	HotSpot 6.0 [10]
Network Model	Noxim NoC, mesh topology, X-Y routing [22]

by a saturation in temperature is noticeable over a time period of 5 ms. We sampled this time taken, to avoid sharp rise in temperature before making a consecutive boosting decision. It should be noted that the rate of temperature accumulation varies with initial conditions viz., temperature of chosen cores at the instance of boosting, simultaneously active cores neighboring the boosted cores and change in level of frequency. In our design, we allow the boosting epoch and boosting frequency level as parametric values for adaptability of the policy. For evaluation purposes, we chose an epoch  $E_{boost}$  of length 2.5 ms, with 4 levels of boosting such that each level corresponds to 200MHz, assuming the critical temperature  $T_{crit}$  as 80 °C. A dynamic setting of boosting epoch and boosting length would be possible, provided with fine-grained monitoring and control of on-chip temperatures. However, dynamic setting of boosting epoch and level for optimal boosting period requires exhaustive search over concurrent applications, making it NP-hard problem. Within the scope of our work, we considered a static setting which is a light weight working solution.

**Evaluation Metrics:** We evaluate our dark silicon patterning approach *PAT* that prefers sparsity against *CoNA*, that prefers contiguity [23]. Since we relaxed the constraint on contiguity among individual applications, we chose to compare *PAT* against *CoNA* to quantify the effect of our patterning approach against contiguity. As a baseline, we consider the system architecture presented in Section 3 which uses the power controller presented in Section 3.2 [19] for power actuation using traditional DVFS and power gating knob. For comparison of dense and sparse mappings, we use *CoNA* and *PAT* mapping policies respectively in the run-time mapping unit (RMU) module. We also compare the same mapping strategies from a boosting perspective, simulating them with the boost controller enabled (from here on referred as *PAT++* and *CoNA++*). In this case, we augmented the system baseline comprising of power controller using traditional DVFS and power gating with the adBoost controller as the policy for boosting module. We simulate

the system over a period in which 100 applications enter and leave the system. The entry sequence of applications is maintained the same for different mapping approaches for a fair comparison. We evaluate both the mapping strategies with and without boosting, over different sizes of the network (64-core and 144-core) and different number of simultaneously active cores. We use a combination of higher and lower number of applications arriving for execution, resulting in variable number concurrent applications and thus active cores and dark cores. This leads to two scenarios of increasing dark silicon, precisely emulating 25% and 50% dark cores, for evaluation. Throughout the evaluation, we refer to these combinations of sizes and active cores - (a) 64 cores, 25% inactive, (b) 144-cores, 25% inactive, (c) 64-cores, 50% inactive, (d) 144-cores, 50% inactive. Given the fixed number and sequence of applications, we compare power budgets, average waiting time (AWT), thermal profiles and throughput offered by CoNA and PAT. With the boosting controller enabled, we also compare average period of boosting for CoNA++ and PAT++. In every case, we simulated the platform at 16 nm technology and a critical temperature of 80 °C.

#### 4.1 Power Budgets and Performance

Figure 10 shows the power budgets offered by CoNA and PAT for 64-core and 144-core systems over different number of active cores. The power budgets are calculated with the entry of an application using the TSP library [5]. Figure 10 (a) and 10 (b) have 25% of cores always inactive while 10 (c) and 10 (d) have 50% inactive cores. Power budgets offered by PAT are higher than that of CoNA due to the sparsity of patterned mapping. At lower sizes of the network and relatively higher number of active cores, power budget gains from PAT are relatively lower due to limited number of dark cores that can be patterned across higher number of active cores. The amount of power budget gain with PAT is higher with lower number of active cores i.e., with higher percentage of dark silicon. For instance, gains for both 64-cores and 144-core systems with 50% inactive cores is significant when compared to that of 64-cores and 144-cores systems with 25% inactive cores. This indicates that as the amount of dark silicon increases with technology node scaling, PAT provides higher power budgets. The surplus gain obtained at every instance of an application's entry in turn would be utilized to accommodate incoming applications if possible, which reflects in higher performance and better power budget utilization. As we increase the number of cores, power budget gains will increase with larger area and more number of active cores. However, a relative power budget gain compared between two different mappings depends on number of dark cores among total cores that can be patterned for power budget gains.

Applications that entered the system when there is not enough power budget are forced to wait in the queue until sufficient power budget becomes available. We refer to this time elapsed between application entering the system and starting its execution as the average waiting time (AWT). With CoNA offering relatively lower power budgets compared to PAT, the system often does not enough budget to service incoming applications, increasing the AWT. In contrast, PAT offers power budget which can be utilized to

accommodate incoming applications more frequently. Thus, PAT minimizes the AWT of applications, reducing the time spent by the applications waiting in the queue. Figure 11 shows the normalized AWT of CoNA, PAT, CoNA++ and PAT++. The gain in AWT for PAT over CoNA is evident across different sizes of network and active cores due to the higher power budget. CoNA++ has a better AWT than PAT as it benefits from the frequency boosting which provides higher performance. PAT++ has better AWT compared to all the other strategies with both higher power budget gain and longer periods of boosting due to patterned mapping.

Lower AWT implies applications' request for execution being serviced quicker, reflecting a lower overall execution time. Figure 12 shows normalized throughput gains for CoNA and PAT - with and without boosting. Throughput is calculated as the time taken for 100 applications to enter and leave the system. PAT and PAT++ have higher throughput compared to CoNA and CoNA++, as expected due to the surplus power budgets. Similar to the power budget gains, throughput gains also are significant with increase in dark silicon. Throughput gain of PAT over CoNA can be attributed to two factors - i) higher power budget offered for a given application, which improves per-application performance and ii) higher power budget allocation for the chip over several concurrent applications, which improves per-chip throughput. The surplus power budget gained with entry of a sequence of applications is accumulated and utilized whenever possible to accommodate new applications waiting in the queue. Within the same amount of time, PAT would be able to execute more applications than CoNA and/or execute the same number of applications as CoNA within a lesser time. It should be noted that for same sequence of incoming applications, AWT decreases with increase in number of active cores, as there are higher amount of resources available. However, increase in number of active cores can be a result of either larger mesh sizes or a relative increase in number of active cores i.e., lower number of dark cores. The gains for PAT over CoNA and PAT++ over CoNA++ also follows the trend of AWT in respective cases, reflecting the effect of AWT on throughput. In addition to these factors, throughput gain in case of PAT++ over CoNA++ with boosting can be attributed to more frequent thermal violations of CoNA++. Every time the temperature approaches 80°C, frequency of active cores has to be lowered for thermal safety, which in turn reduces some performance. With PAT++, thermal constraints are honored, retaining a higher period of boosting and high performance. CoNA++ however violates thermal constraints occasionally, triggering frequency downscaling and thus lowering the performance.

#### 4.2 Thermal Profile and Boosting

Figure 13 shows the thermal profiles of CoNA++ and PAT++ for different mesh sizes and active cores. Precisely, we present the maximum temperature observed on the chip among all the cores. PAT++ is relatively cooler when compared to CoNA++ due to the sparsity of active cores. CoNA++ on the other hand is more susceptible to accumulate temperature and form hot-spots with tightly packed active cores. Since the surplus power budget from PAT++ is utilized either to map new incoming applications or to boost the frequency (or both), the difference between temperatures

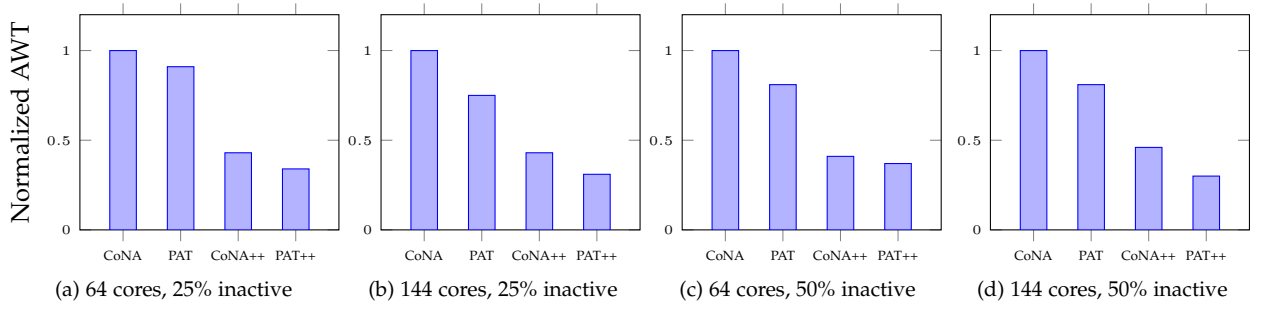


Fig. 11. Average waiting times (AWT) of CoNA and PAT with varying number of active cores. AWT is calculated as the average time spent by each application in the queue before it starts execution, over 100 applications that enter and leave the system. (a). AWT for a 64-core system with 25% inactive cores. (b). AWT for a 144-core system with 25% inactive cores. (c). AWT for a 64-core system with 50% inactive cores. (d). AWT for a 144-core system with 50% inactive cores.

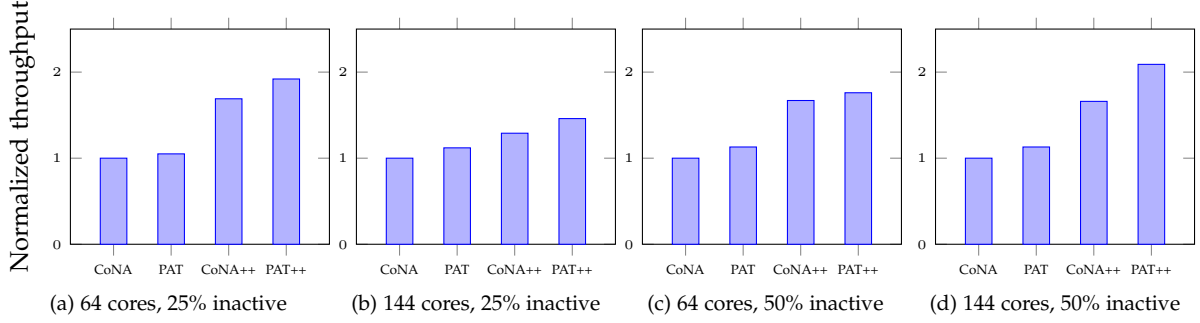


Fig. 12. Throughput gain of PAT over CoNA with varying number of active cores. Normalized throughput is calculated as time taken for 100 applications to enter and leave the system. (a). 64-core system with 25% inactive cores. (b). 144-core system with 25% inactive cores. (c). 64-core system with 50% inactive cores. (d). 144-core system with 50% inactive cores.

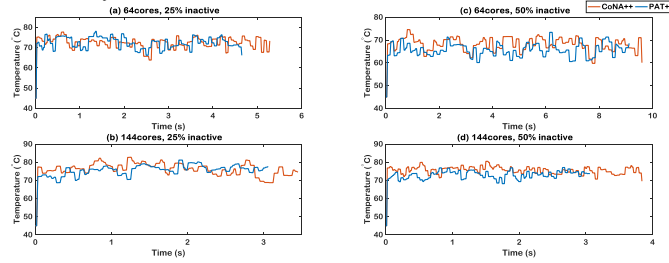


Fig. 13. Temperature profiles of CoNA++ and PAT++ for simulation time of 100 applications entering and leaving the system. (a). 64-core system with 25% inactive cores. (b). 144-core system with 25% inactive cores. (c). 64-core system with 50% inactive cores. (d). 144-core system with 50% inactive cores.

is relatively small. Operating the chip exclusively at lower temperatures with the same power budget instead of utilizing the surplus budget to increase performance could be a designer's choice.

Figure 14 shows periods of boosting for CoNA++ and PAT++ over different mesh sizes and active cores. Boosting period is calculated as the accumulated sum of the number of boosting epochs ( $E_{boost}$ ) elapsed during the overall simulation time. PAT++ has a relatively longer period of boosting compared to CoNA++ in all the cases. PAT++ follows sparse mapping of an application and also disperses concurrent applications which result in lower temperatures. Boosting active cores that are already operating lower temperatures allows the increase in frequency to be sustained for longer intervals, before the temperature eventually might approach critical temperature. Both the higher power budget offerings and lower operating temperatures from dark silicon patterning favor PAT++ in sustaining the boosting period for longer periods. Throughput gains from PAT++ over CoNA++ can also be attributed for the

additional performance extracted from the longer boosting periods. Boosting in case of CoNA++ results in a relatively lower gain, given the contiguous nature of mapping that accumulates temperature, resulting in active cores that are relatively hot. Boosting the cores which are already at higher temperatures would further contribute to approach the critical temperature faster. As on-chip temperatures approach critical temperature, frequency of boosted cores has to be scaled down towards normal mode of execution, limiting the effective boosting period and thus performance gains.

Another consideration to be made for power budget utilization is the benefits obtained from boosting. Figure 15 shows the normalized period of boosting per overall execution time. It is to be noted that higher period of boosting might not necessarily translate into equivalent higher performance. For instance, the period of boosting is higher in case of (b) 144-cores, 25% inactive - with higher number of active cores, and lower in case of (c) 64-cores, 50% inactive - with lower number of active cores. With higher number of active cores, the incidence of thermal violation

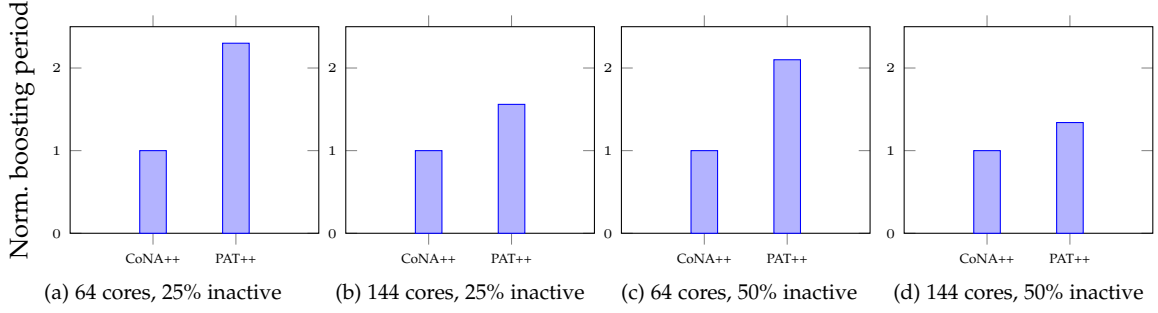


Fig. 14. Boosting period gains for PAT++ over CoNA++. Boosting period is calculated as time the simulation ran in boosting mode per overall execution time for 100 applications to enter and leave the system. (a). 64-core system with 25% inactive cores. (b). 144-core system with 25% inactive cores. (c). 64-core system with 50% inactive cores. (d). 144-core system with 50% inactive cores.

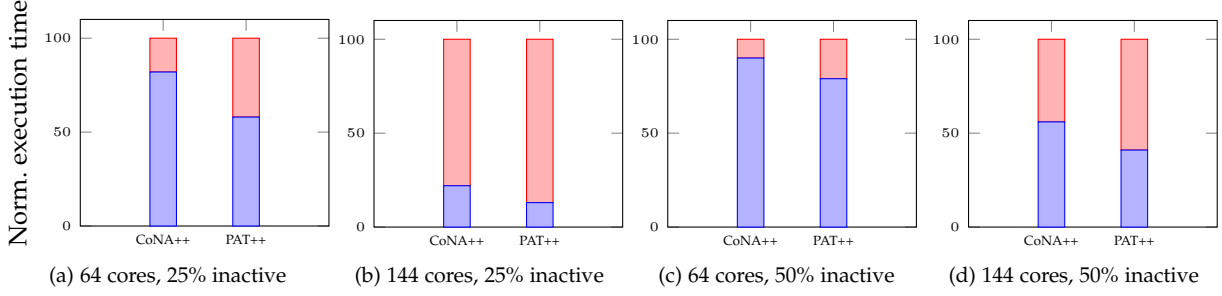


Fig. 15. Boosting period for CoNA++ and PAT++. The stack shows normalized execution time split into normal execution (in blue) and boosted execution (in red) for (a). 64-core system with 25% inactive cores, (b). 144-core system with 25% inactive cores, (c). 64-core system with 50% inactive cores, (d). 144-core system with 50% inactive cores.

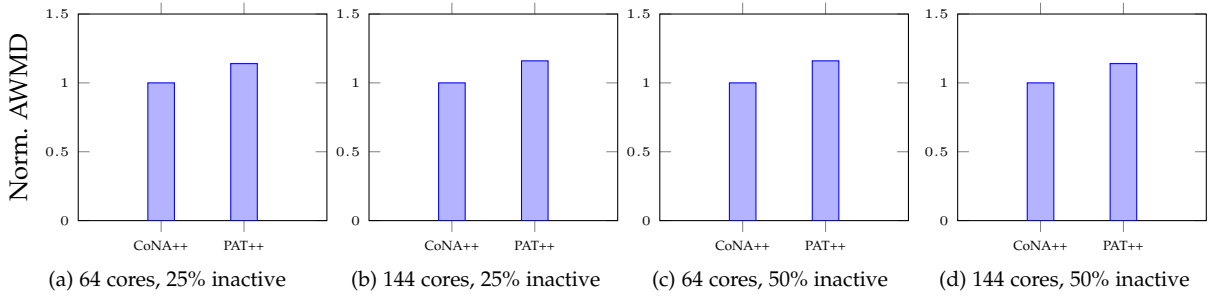


Fig. 16. Average weighted manhattan distance (AWMD) of PAT over CoNA. (a). 64-core system with 25% inactive cores. (b). 144-core system with 25% inactive cores. (c). 64-core system with 50% inactive cores. (d). 144-core system with 50% inactive cores.

is more frequent. The controller thus rolls between normal and boost modes of execution. There are several phases of - boosting followed by frequency downscaling, increasing the period of boosting. In contrast, with fewer active cores, the temperature barely approaches critical temperature. This gives an uninhibited boosting scenario, such that maximum operating frequency is reached sooner. Despite not having a longer boosting period, gain in performance is achieved through effective utilization of boosting.

#### 4.3 Overhead with Sparse Mapping and Controllers

The total number of hops traversed by all packets sent and received by tasks of an application, referred as Weighted Manhattan Distance (WMD) represents the penalty levied by inter-core communication in an application. Although sparse mapping provides high power budget and low operating temperatures, dispersed tasks of an application increases the WMD, compared to contiguous mappings such as CoNA. Figure 16 shows the average weighted Manhattan distance (AWMD) of PAT over CoNA for different mesh sizes and active cores. The normalized AWMD is higher for PAT when compared to CoNA, indicating a higher latency. Despite the latency, PAT still offers better performance over

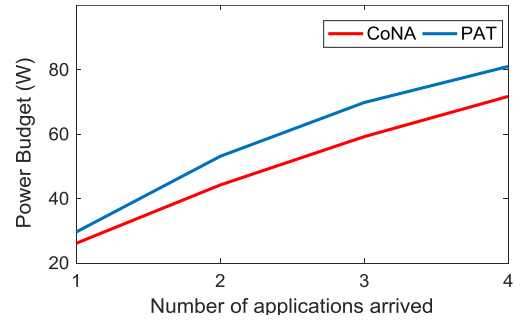


Fig. 17. Power budgets with CoNA and PAT mappings. Applications concurrently run: streamcluster, ferret, bodytrack and blackscholes

CoNA, with higher power budgets that compensate for the additional packet latency. Our timing model includes additional packet latency induced with sparse mappings for fair comparison. With the same sequence of workloads, patterned mapping configurations per each application remain similar in each of the cases in Figure 16 (a)-(d), although the location of first node might vary subject to concurrently running applications. This results in a similar amount of ad-

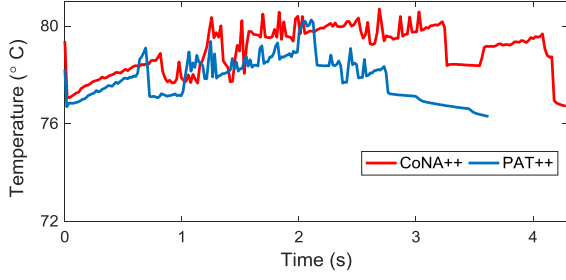


Fig. 18. Temperature profiles with boosting using CoNA and PAT mappings. Applications concurrently run: streamcluster, ferret, bodytrack and blackscholes

ditional hops traversed and thus similar AWMD values with sparse mappings. The additional energy overhead incurred in inter-task communication due to additional hops with sparse mappings is 14%, 16%, 14% and 16% respectively, which is also proportional to the performance overhead (AWMD). The proactive first node finding uses a look up table to find appropriate first node for any possible sized incoming application by continuously keeping track of density factors of each node. While first node selection is proactive, updating the density factors is of  $O(n \times M \times \sqrt{M})$  complexity, where  $n$  is the number of tasks of an application (or application size) and  $M$  is the mesh size. The patterning algorithm has complexity of  $O(n^2)$ , where  $n$  is the size of the application. The complexity of power controller and boost controllers are  $O(M)$  and  $O(n)$  respectively, where  $M$  is the mesh size and  $n$  is number of tasks within an application. The run-time overhead induced with the controller includes latency elapsed in transmitting the frequency settings from the central manager to the respective cores. This depends on the Manhattan distance between the central manager and the active core, such that the worst case latency is equal to  $\sqrt{2} \times M \times \text{hopLatency}$ , where  $M$  is the size of the network and  $\text{hopLatency}$  is the per-hop latency of a packet. The overhead of finding first node, mapping, power management and boosting decisions and enforcement of voltage and frequency settings are included in the execution timing model.

#### 4.4 Gains with other workloads

We also evaluate our proposed approach on real workloads, apart from the synthetic task graphs. We chose 4 applications from PARSEC benchmark suite viz., blackscholes, bodytrack, ferret and streamcluster and native small input data. We extracted execution traces of the applications using Intel Pin tool and power traces using McPAT. Technology node specifications are obtained from Lumos [24], validated against Niagara-2 in-order cores at 16nm technology node. The applications are concurrently run on a 64-core system over the same in-house platform platform used for evaluating the synthetic workloads, described earlier. Power budgets achieved at run-time with entry of each new application using CoNA and PAT are shown in Figure 17. With sparse mapping providing necessary cooling effect among active cores, PAT offers upto 17% more power budget, compared to CoNA. Thermal profiles of the same set of applications using the adBoost controller over CoNA++ and PAT++ are shown in Figure 18. It can be

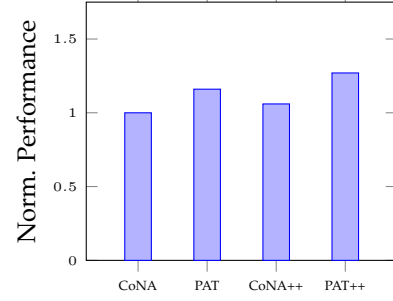


Fig. 19. Normalized performance with CoNA and PAT mappings with and without boosting. CoNA and PAT represent performance using power controller alone, CoNA++ and PAT++ represent performance also using boost controller. Applications concurrently run: streamcluster, ferret, bodytrack and blackscholes.

observed that PAT++ maintains a lower temperature profile with patterned mapping, which subsequently provides more thermal headroom for PAT++ compared to CoNA++, which has a limited scope for boosting. The subsequent performance gains with PAT++ over CoNA++ along with PAT over CoNA are shown in Figure 19. The additional power budget and relatively lower temperatures offered by sparse mapping allows PAT++ to be boosted for upto 18% longer compared against CoNA++.

## 5 RELATED WORK

**Application Mapping:** State-of-the-art dynamic application mapping strategies have focused on performance aspects through minimizing congestion and spatial alignment [25][6][26], without any considerations on power budgets. The emphasis of mapping techniques was largely constrained to inter-task communication and contiguous mapping with performance objectives alone [27][28][29]. Precisely, these algorithms do not consider maximizing performance within given power budgets. The limitations on future many-core systems with dark silicon challenge requires re-structuring of mapping objectives towards improving power budgets by considering dark silicon, and thus performance. Contiguous mapping avoiding dispersion and fragmentation for better performance was proposed in [23]. However, power budget limitation with dark silicon changes the implications of dispersion and fragmentation on system performance. A non-contiguous mapping through geometrical partitioning of the network is presented in [30], minimizing performance penalties by mapping communicating tasks on nearby cores and the rest in proximity. This establishes that non-contiguous mappings not necessarily affects system performance, which can be exploited to attack dark silicon. Patterning inevitable dark cores around active neighbors for improving power budget utilization was presented in our previous work [8]. In the similar vein, maximizing performance through adaptive mapping within a given power budget, by exploiting process variations, reliability and thermal considerations were presented in [31] [32] [33] [34] [35] [15]. This emphasizes the significance of application mapping on maximizing performance within fixed power budgets.

**Dark Silicon and Power Budgeting:** Dark silicon phenomenon and its implications on performance are well established in [3][36]. Precisely, power densities of computer systems are rising with technology scaling, which leads to



thermal violations. Common practise of ensuring thermal safety is to cap the power consumption at a fixed upper bound - thermal design power (TDP) [4]. A slightly flexible option for power budgeting were AMD and Intel Corporations' CPUs with configurable TDP options, however without any fine grained control [37][38]. Selectively mapping applications to restrict power consumption to a fixed limit was proposed in [13], using a feedback controller. There are other reactive strategies that trigger dynamic power and thermal management techniques to honor TDP [39] [40] [41] [19] [12] [15] [42]. Pagani *et al.* have proposed more realistic and dynamic estimate of power budget by expressing it as a function of simultaneously active cores i.e., mapping configuration [5]. They provide a light weight C library to estimate thermal safe power (TSP) for a given mapping configuration along with the possible worst case TSP for a given number of active cores. The effect of application mapping on power budget, by aligning dark cores among active cores have been presented in [43]. It shows a two fold gain from patterning - better power budget utilization and lower operating temperatures. [44] quantified the advantages dark silicon patterning which results in higher power budgets and lower temperature profiles of the chip. However, principles for dark silicon patterning based mapping, utilizing surplus power budget from patterning, allocation of the same for boosting performance have remained unexplored. To summarize, most of the strategies mentioned above were reactive to power/thermal violations but do not directly address the performance penalties or maximizing performance within given power budgets.

**Boosting:** The initial idea of throttling an application's frequency to maximize performance per watt was proposed by Murali et al. [45]. The goal is to choose appropriate pareto-optimal space among instructions per second and power per instructions, depending on amount of parallelism in an application. This feature was later implemented in Intel's architecture as the TurboBoost feature which increases the frequency of active cores beyond the base frequency in short bursts of time for performance gains [46]. Raghavan et al. have proposed the concept of computational sprinting - exceeding the power limitations of a chip over short bursts of time by activating cores which are otherwise dark [47]. They focus on accelerating particularly the parallel sections in applications which can have performance gains with sprinting. Further, they also use frequency sprinting on top of parallel sprinting, relying on heat dissipation during a cool down period that follows the boosting period. Selectively boosting an application while scaling down the frequency of other concurrent applications is proposed in [16]. Most of these works emphasize on boosting the frequency, while having enough thermal headroom and power budget remain the key requirements to facilitate boosting. In our work, we first address the issue of creating surplus power budget through dark silicon patterning, followed by efficient allocation of the available thermal headroom to then boost the frequency of active cores.

## 6 CONCLUSION

We have presented *adBoost* controller for thermal aware performance boosting in many-core systems. *adBoost* ben-

efits from dark silicon patterning (PAT), a mapping strategy to evenly distribute temperature across the chip in order to improve utilizable power budget. PAT maps tasks and applications in a sparse manner by patterning inevitable cooler dark cores around hotter active cores, allowing them to utilize power efficiently before reaching critical temperature. Our power controller efficiently allocates the surplus power budget from patterning to map new applications or throttle current applications while *adBoost* controller scales the frequency upon performance requirements. Lower on-chip temperatures with PAT provides sustainable periods of boosting, improving both per-application latency and per-chip throughput. In comparison with the state-of-the-art contiguous mapping strategies, the combination of *adBoost* and PAT offers higher power budget, lower temperatures and sustain longer periods of boosting. These put together reflect in better throughput and efficient utilization of power budget. Our strategy can also be extended for heterogeneous architectures, provided the prior knowledge on performance-per-watt of every application/task. This allows the choice of more suitable set of heterogeneous cores at the mapping phase, which reflects on lower on-chip temperatures using power efficient cores. This can be leveraged to boost the frequency of low-power cores for improved performance and/or use the thermal headroom created with the low-power cores to boost the performance of other power hungry cores.

## ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Curie grant agreement No 705617.

## REFERENCES

- [1] A. Kanduri *et al.*, "A perspective on dark silicon," in *The Dark Side of Silicon*. Springer International Publishing, 2017, pp. 3–20.
- [2] A. Rahmani and others., *The Dark Side of Silicon*, 1st ed. Springer, Switzerland, 2016.
- [3] H. Esmaeilzadeh *et al.*, "Dark silicon and the end of multicore scaling," *IEEE Micro*, no. 3, pp. 122–134, 2012.
- [4] "Intel Xeon Processor - Measuring Processor Power, revision 1.1," in *White paper, Intel Corporation*, April, 2011.
- [5] S. Pagani *et al.*, "Tsp: thermal safe power: efficient power budgeting for many-core systems in dark silicon," in *Proc. of CODES+ISSS*, 2014, p. 10.
- [6] M. Fattah, M. Daneshmand, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *Proc. of DAC*, 2013.
- [7] L. Wang and K. Skadron, "Dark vs. Dim Silicon and Near-Threshold Computing Extended Results," in *University of Virginia Dept. of CS Technical Report TR-2013-01*, 2012.
- [8] A. Kanduri *et al.*, "Dark silicon aware runtime mapping for many-core systems: A patterning approach," in *Proc. of ICCD*, 2015, pp. 573–580.
- [9] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*. IEEE, 2008, pp. 123–134.
- [10] W. Huang *et al.*, "Hotspot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Tran. on VLSI Systems*, vol. 14, no. 5, pp. 501–513, 2006.
- [11] S. P. et. al., "Thermal safe power (tsp): Efficient power budgeting for heterogeneous manycore systems in dark silicon," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 147–162, 2017.
- [12] A. M. Rahmani *et al.*, "Reliability-aware runtime power management for many-core systems in the dark silicon era," *IEEE Tran. on VLSI Systems*, vol. 25, no. 2, pp. 427–440, 2017.

- [13] M.-H. Haghighbayan *et al.*, "Dark silicon aware power management for manycore systems under dynamic workloads," in *Proc. of ICCD*, 2014, pp. 509–512.
- [14] J. Zhan *et al.*, "Designing energy-efficient noc for real-time embedded systems through slack optimization," in *Proc. of DAC*. ACM, 2013, p. 37.
- [15] J. Z. *et al.*, "NoC-Sprinting: Interconnect for Fine-Grained Sprinting in the Dark Silicon Era," in *DAC*, 2014.
- [16] S. Pagani *et al.*, "seboost: Selective boosting for heterogeneous manycores," in *Proc. of CODES+ISSS*, 2015, pp. 104–113.
- [17] M.-H. Haghighbayan *et al.*, "Mappro: Proactive runtime mapping for dynamic workloads by quantifying ripple effect of applications on networks-on-chip," in *Proc. of NOCS*, 2015, pp. 1–8.
- [18] H. E. White, *Modern College Physics*. Van Nostrand, 1948.
- [19] A.-M. Rahmani *et al.*, "Dynamic power management for manycore platforms in the dark silicon era: A multi-objective control approach," in *Proc. of ISLPED*, 2015, pp. 219–224.
- [20] "TGG: Task Graph Generator," URL: <http://sourceforge.net/projects/taskgraphgen/>, 2010.
- [21] S. Li *et al.*, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. of MICRO*, 2009, pp. 469–480.
- [22] F. Fazzino, M. Palesi, and D. Patti, "Noxim: Network-on-chip simulator," URL: <http://sourceforge.net/projects/noxim>, 2008.
- [23] M. Fattah *et al.*, "Cona: Dynamic application mapping for congestion reduction in many-core systems," in *Proc. of ICCD*, 2012, pp. 364–370.
- [24] Lumos Framework, <http://liangwang.github.io/lumos/>. Accessed: 2014-05-20. [Online]. Available: <http://liangwang.github.io/lumos/>
- [25] C.-L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," in *Proc. of ICCD*, 2008, pp. 164–169.
- [26] M. Fattah, P. Liljeberg, J. Plosila, and H. Tenhunen, "Adjustable contiguity of run-time task allocation in networked many-core systems," in *Proc. of ASP-DAC*, 2014, pp. 349–354.
- [27] C. Chen-Ling *et al.*, "Energy- and Performance-Aware Incremental Mapping for Networks on Chip With Multiple Voltage Levels," *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, 2008.
- [28] M. A. Bender, D. P. Bunde, E. D. Demaine, S. P. Fekete, V. J. Leung, H. Meijer, and C. A. Phillips, "Communication-aware processor allocation for supercomputers: Finding point sets of small average distance," *Springer Algorithmica*, vol. 50, no. 2, pp. 279–298, 2008.
- [29] E. Carvalho, N. Calazans, and F. Moraes, "Heuristics for dynamic task mapping in noc-based heterogeneous mpocs," in *Proc. of RSP*, 2007, pp. 34–40.
- [30] M. Deveci *et al.*, "Exploiting geometric partitioning in task mapping for parallel computers," in *Proc. of PDP*, 2014, pp. 27–36.
- [31] B. Raghunathan, Y. Turakhia, S. Garg, and D. Marculescu, "Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors," in *Proc. of DATE*, 2013, pp. 39–44.
- [32] N. Kapadia and S. Pasricha, "Varsha: Variation and reliability-aware application scheduling with adaptive parallelism in the dark-silicon era," in *DATE*. IEEE, 2015, pp. 1060–1065.
- [33] M.-H. Haghighbayan *et al.*, "Performance/reliability-aware resource management for many-cores in dark silicon era," *IEEE Tran. on Computers*, 2017.
- [34] Y. Liu, Y. Ruan, Z. Lai, and W. Jing, "Energy and thermal aware mapping for mesh-based noc architectures using multi-objective ant colony algorithm," in *Proc. of IEEE International Conference on Computer Research and Development*, vol. 3, 2011, pp. 407–411.
- [35] M. Bao, A. Andrei, P. Eles, and Z. Peng, "Temperature-aware task mapping for energy optimization with dynamic voltage scaling," in *Proc. of DDECS*, 2008, pp. 1–6.
- [36] M. Taylor, "Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse," *Proc. of DAC*, 2012, pp. 1131–1136.
- [37] "Intel Corporation. Fourth Generation Mobile Processor Family Data Sheet," in *White paper, Intel Corporation*, July, 2014.
- [38] "AMD Kaveri APU A10-7800," accessed: 2015-02-28. [Online]. Available: [http://www.phoronix.com/scan.php?page=articleitem&amd\\_a\\_45watt&num=1](http://www.phoronix.com/scan.php?page=articleitem&amd_a_45watt&num=1)
- [39] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Temperature management in multiprocessor socs using online learning," in *Proc. of DAC*, 2008, pp. 890–893.
- [40] L. Thiele, L. Schor, H. Yang, and I. Bacivarov, "Thermal-aware system analysis and software synthesis for embedded multi-processors," in *Proc. of DAC*, 2011, pp. 268–273.
- [41] T. Komoda *et al.*, "Power capping of cpu-gpu heterogeneous systems through coordinating dvfs and task mapping," in *Proc. of ICCD*, 2013, pp. 349–356.
- [42] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A case for dynamic frequency tuning in on-chip networks," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*. IEEE, 2009, pp. 292–303.
- [43] M. Shafique *et al.*, "Dark silicon as a challenge for hardware/software co-design: Invited special session paper," in *Proc. of CODES+ISSS*, 2014, p. 13.
- [44] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The eda challenges in the dark silicon era," in *Proc. of DAC*, 2014, pp. 1–6.
- [45] M. Annamaram, E. Grochowski, and J. Shen, "Mitigating amdahl's law through epi throttling," in *Proc. of ISCA*, 2005, pp. 298–309.
- [46] E. o. Rotem, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, 2012.
- [47] A. Raghavan *et al.*, "Computational sprinting," in *Proc. of HPCA*. IEEE, 2012, pp. 1–12.



**Anil Kanduri** received M.Sc (Tech) in Embedded Computing from University of Turku, Finland in 2014. Since then, he has been working towards PhD degree in Computer Systems at the department of Information Technology, University of Turku. His research interests are in energy efficient computer architectures, run-time management and approximate computing.



**Mohammad-Hashem Haghighbayan** received the BS degree in computer engineering from Ferdowsi University of Mashhad, Iran, the MS degree in computer architecture from University of Tehran, Iran, and PhD in information and communication technology from University of Turku, Finland. He is currently a post-doctoral researcher in department of future technologies in University of Turku, Finland. His research interests include high-performance energy-efficient architectures, power management techniques, machine learning, and online/offline testing. He has several years of experience working in industry as well as developing research tools before starting his PhD.



**Amir M. Rahmani** received his Master's degree from Department of ECE, University of Tehran, Iran, in 2009 and Ph.D. degree from Department of IT, University of Turku, Finland, in 2012. He also received his MBA jointly from Turku School of Economics and European Institute of Innovation & Technology (EIT) ICT Labs, in 2014. He is currently Marie Curie Global Fellow at University of California Irvine (USA) and TU Wien (Austria). He is also an adjunct professor (Docent) in embedded parallel and distributed computing at the University of Turku, Finland. His research interests span Self-aware Computing, Energy-efficient Many-core Systems, Runtime Resource Management, Healthcare Internet of Things, and Fog/Edge Computing. He has served on a large number of technical program committees of international conferences, such as DATE, GLSVLSI, DFT, ESTIMedia, CCNC, MobiHealth, and others, and guest editor for special issues in journals such as JPDC, FGCS, MONET, Sensors, Supercomputing, etc. He is the author of more than 150 peer-reviewed publications. He is a senior member of the IEEE.



**Muhammad Shafique** (M'11, SM'16) is a full professor at the Institute of Computer Engineering, Dept. of Informatics, TU Wien, Austria. He is directing the Group on Computer Architecture and Robust, Energy-Efficient Technologies (CARE-Tech). He was a senior research group leader at Karlsruhe Institute of Technology (KIT), Germany for more than 5 years. He received his Ph.D. in Computer Science from KIT in January 2011. Before, he was with Streaming Networks Pvt. Ltd. developing video coding algorithms and

optimizations for VLIW-based Processors. His research interests are in computer architecture, power-/energy-efficient systems, robust computing covering various dependability/reliability/resilience aspects, emerging computing trends like neuromorphic and approximate computing, neurosciences, emerging technologies and nanosystems, hardware security, adaptive and self-learning, FPGAs, MPSoCs, and embedded systems. Dr. Shafique received the prestigious 2015 ACM/SIGDA Outstanding New Faculty Award, six gold medals in educational career, and several best paper awards and nominations at prestigious conferences like CODES+ISSS, DATE, DAC, and ICCAD, Best Master Thesis Award, and Best Lecturer Award. He has given several Invited Talks, Tutorials, and 2 Keynotes. He has also organized many special sessions at premier conferences (like DAC, ICCAD, DATE, and Codes+ISSS) and served as the Guest Editor for IEEE Design and Test Magazine (DT) and IEEE Transactions on Sustainable Computing (T-SUSC). He has served as the TPC co-Chair of ESTIMedia and LPDC, General Chair of ESTIMedia, and Track Co-Chair at DATE 2017-18 and FDL 2017. He has served on the program committees of several IEEE/ACM conferences like ISCA, ICCAD, DATE, CASES, FPL, and ASPDAC. He is a senior member of the IEEE and IEEE Signal Processing Society (SPS), and a member of ACM. He holds one US patent and over 150 papers in premier journals and conferences.



**Axel Jantsch** received the diploma and doctorate degrees from the Technical University of Vienna, Vienna, Austria, in 1988 and 1992, respectively. He was with Siemens Austria, Vienna, Austria, as a system validation engineer from 1995 to 1997. Since 1997, he has been an associate professor with the Royal Institute of Technology (KTH), Stockholm, Sweden. Since 2000, has been a Docent, and since December 2002, a full professor of Electronic System Design with the Department of Electronic Systems. He has published more than 200 papers in international conferences and journals.

He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES+ISSS, SOC, NOCS, and others, and one book in the areas of VLSI design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks on chip. He received the Alfred Schrodinger Scholarship from the Austrian Science Foundation while a guest researcher with KTH between 1993 and 1995. He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES+ISSS, SOC, NOCS, and others. He has been the TPC Chair of SSDL/FDL 2000, the TPC Co-Chair of CODES+ISSS 2004, the General Chair of CODES+ISSS 2005, and the TPC Co-Chair of NOCS 2009. From 2002 to 2007, he was a subject area editor for the Journal of System Architecture. He is a member of the IEEE.



**Pasi Liljeberg** received MSc and PhD degrees in electronics and information technology from the University of Turku, Turku, Finland, in 1999 and 2005, respectively. He received Adjunct professorship in embedded computing architectures in 2010. Currently he is working as a full professor in University of Turku in the field of Embedded Systems and Internet of Things. At the moment his research interests are computer architectures, fog computing, energy efficient architectures, approximate and adaptive computing, biomedical engineering, health technology and Internet of Things.

In that context he has established and leading the Internet-of-Things for Healthcare (IoT4Health) research group. Liljeberg is the author of more than 260 peer-reviewed publications. He is a member of the IEEE.