Dark Silicon Aware Runtime Mapping for Many-core Systems: A Patterning Approach

Anil Kanduri[1], Mohammad-Hashem Haghbayan[1], Amir-Mohammad Rahmani[1,3],
Pasi Liljeberg[1], Axel Jantsch[2], and Hannu Tenhunen[1,3]
[1]Department of Information Technology, University of Turku, Turku, Finland
[2]TU Wien, Austria
[3]Department of Industrial and Medical Electronics, KTH Royal Institute of Technology, Stockholm, Sweden
Email: {*spakan, mohhag, amirah, pakrli*}*@utu.fi*, *axel.jantsch@tuwien.ac.at*, *hannu@kth.se*

***Abstract*—Limitation on power budget in many-core systems leaves a fraction of on-chip resources inactive, referred to as dark silicon. In such systems, an efficient run-time application mapping approach can considerably enhance resource utilization and mitigate the dark silicon phenomenon. In this paper, we propose a dark silicon aware runtime application mapping approach that patterns active cores alongside the inactive cores in order to evenly distribute power density across the chip. This approach leverages dark silicon to balance the temperature of active cores to provide higher power budget and better resource utilization, within a safe peak operating temperature. In contrast with exhaustive search based mapping approach, our agile heuristic approach has a negligible runtime overhead. Our patterning strategy yields a surplus power budget of up to 17% along with an improved throughput of up to 21% in comparison with other state-of-the-art run-time mapping strategies, while the surplus budget is as high as 40% compared to worst case scenarios.**

Keywords*—*Dark Silicon; Power Budgeting; Runtime Mapping

I. Introduction

Power density of many-core systems is alarmingly increasing for every technology node generation, attributed to slack voltage scaling that is not on par with technology node scaling and exponential rise in leakage power [1]. Increase in power density leads to thermal issues [2], forcing the chip's functionality to fail. To ensure a safe operation, it is critical for the chip to perform within a fixed upper bound on power budget [3]. In order to stay within in this limit, a certain section of the chip has to remain inactive - a growing phenomenon termed as Dark Silicon [4]. Compute intensity of future applications such as deep machine learning, virtual reality, big data etc., demands further technology node scaling, subsequently leading to further rise in power density and dark silicon. ITRS projections have predicted that by 2020, designers would face up to 90% of dark silicon, meaning that only 10% of the chip's hardware resources are useful at any given time [5]. Increasing dark silicon directly reflects on performance to a point that multi-core and many-core scaling provides zero gain [4].

Many-core systems face a new set of challenges at the verge of dark silicon to continue providing the expected performance and efficiency [6]. Run-time application mapping policy is one of the key factors that can influence performance and energy efficiency of many-core systems [7]. Mapping is the process of choosing a preferable set of cores on the chip to run tasks of an application, minimizing congestion and maximizing performance [8], [9]. With dynamic workload characteristics and un-predictable sequence and arrival of applications, mapping decisions had to be made at run-time for current and future many-core systems. Thus far, designers have been using mapping strategies assuming that all the cores of a chip are active and available. However with dark silicon scenario, this completely alters as not all the cores can be active at a given time and the number of cores that can be active varies depending on activity of other working cores [10]. Existing run-time mapping algorithms are dark silicon agnostic and will not be able to provide the high performance they used to, as they do not consider the availability of active cores and are likely to violate safe power budget. All of these factors necessitates a dark silicon aware run-time mapping strategy to continue achieving performance gain and energy efficiency through technology node scaling.

Thermal Design Power (TDP) is a standard design time metric that has been used to determine a safer upper bound on chip's power consumption. Safer operation of a chip is guaranteed as long as power consumption stays within TDP [11]. TDP is single fixed upper bound that is pessimistically estimated assuming that all the cores are active and are operating at a worst case voltage and frequency. With dark silicon phenomenon, a variable number of cores will be inactive (dark), depending on current set of applications running, ambient temperature and most importantly, number of simultaneously active cores. Thus, the safe upper bound on power budget varies in run-time, as opposed to the conservative upper bound of TDP which leads to under-utilization of available power budget resulting in dark silicon [10]. A sensible way to avoid conservative limit of TDP is to use a variable and realistic upper bound on power consumption, Thermal Saturation Power (TSP), as proposed by Pagani *et al.* [10]. TSP is modeled as a function of simultaneously active cores, their alignment, effect of temperature of a core on its neighbors and ambient temperature. At any given time instance, the amount of power the chip can consume to safely operate will depend on the alignment of active cores, which is determined by application mapping. It can be deduced that for same set of applications and same number of active cores, a certain mapping can result in a higher power budget over other mappings, based on appropriate alignment of active and dark cores. Subsequently, the surplus budget gained through mapping can be utilized to power up more cores, minimizing dark silicon and thus offering higher performance. This is explained through an example presented in Figure 1.

A contiguous mapping of an application with 6 tasks

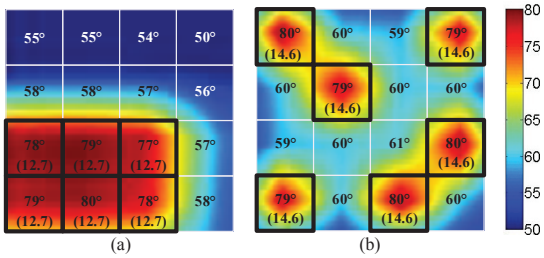


Fig. 1: Effect of mapping on Power Budget [10]

is shown in 1 (a). Since all the active cores are tightly packed, the heat dissipated by every active core affects its neighbors hazardously. As a result, the cores reach their critical temperature of (80°C) after consuming 12.7W of power. This configuration of mapping can effectively utilize 12.7×6 = 76.2W of power, which would be its power budget. Utilizing power beyond this budget would heat up the cores beyond their critical temperature leading to permeant failure. The mapping shown in Figure 1(b) is sparsely distributed in comparison with the previous mapping. In this case, as the cores are separated out, the heating effect of one active core on the other is minimized. Therefore, the cores can now consume 14.6W with reaching their critical temperature. Effectively, the chip's power budget in this case is 14.6×6 = 87.6W, which is 11.2W (14.9%) more than the previous case, establishing the impact of mapping on providing a better power budget. This surplus budget can in turn be used to: i) activate more cores, ii) run current tasks much faster, and iii) run more tasks without reaching critical temperatures.

Although it has been shown that optimal mapping can provide higher power budget [12][13], to the best of our knowledge, no methodical approach exists on how to pattern the active cores alongside dark cores such that they result in higher power budget. In this paper, we propose a dark silicon aware run-time mapping approach which aligns active cores along with dark cores that can evenly distribute heat dissipation across the chip. The surplus budget we gain through mapping rises the upper bound on power consumption which is used to activate more cores, directly mitigating the dark silicon phenomenon. This way, we maximize the utilization on power budget to improve performance and energy efficiency. To the best of our knowledge, ours is the first work to consider dark silicon scenario for run-time application mapping and to pattern the active and dark cores to improve the power budget. The key contributions of this work are as follows:

- A dark silicon aware runtime application mapping approach that aligns active cores with dark cores to offer higher power budget.
- A closed-loop power budgeting platform that keeps the maximum power consumption under safe operational power (i.e., TSP) which varies at runtime.

The rest of the paper is organized as follows: In Section II, related work on dynamic mapping and power budgeting is presented. Our proposed mapping strategy and overview of the system are detailed in Section IV. Experimental setup and results are presented in Section V. Finally, Section VI concludes the paper and discusses potential future work.

II. Related Work

State-of-the-art dynamic application mapping strategies have targeted benefits in terms of network performance, minimizing congestion, system throughput, power optimization, etc [14][8][15], without any fixed upper bound on power consumption. The main focus of mapping strategies so far is on inter-task communication and their objective is maximizing performance, forcing them to map contiguously [16][17][18]. Precisely, these algorithms do not consider the issue of dark silicon. However, power budget (and thus performance) limitations of future many-core systems emphasizes to re-structure the objective of mapping towards improving power budget by considering dark silicon. Conventional mapping policies advocate avoiding dispersion and fragmentation and do not consider the issue of dark silicon which changes the impact of dispersion and fragmentation on system performance [19]. A non-contiguity mapping through geometrical partitioning of the network is presented in [20], which shows that penalties on performance can be minimized by mapping communicating tasks on nearby cores and the rest in proximity, but not necessarily contiguous. They have established that non-contiguous not necessarily affects system performance, although they do not exploit this fact to attack dark silicon. A patterning approach to avoid congestion between packets routed from same row or column is proposed in [21]. They limit the number of tasks to one per row and one per column and do not consider any power budget.

A feedback based power management system is proposed in [3]. They are limited to restricting the violation of TDP through a PID controller and they use a conventional dynamic mapping approach [8], which is not dark silicon aware. An online learning approach is presented in [22] that employs various power management techniques whenever there are hot-spots identified in the system with changing workloads. It is restricted to multi-core systems and does not consider dark silicon. Thermal aware system analysis and calibration is presented in [23] at a lower level of abstraction, yet they do not propose effective task allocation based on their detailed analysis. Liu *et al.* [24] present an energy and thermal aware mapping strategy for NoC-based systems. Their approach is limited to design time (static mapping) which is based on heuristic that estimates temperature, resulting into a near exhaustive search to find optimal nodes. A proactive estimation of potential hot-spots through temperature sensors is presented in [25]. They mitigate identified hot-spots through thread migration and dynamic voltage scaling. This method suffers from performance degradation by voltage down scaling and overhead in migration. Bao *et al.* [26] present a case for balancing the heat dissipation of the chip evenly through a temperature aware mapping. Their mapping is based on voltage down scaling when needed as per critical temperatures of cores, putting a limit on its performance. Power capping of the cores through dynamic voltage and frequency scaling by identifying hot-spots at runtime is presented in [27]. The authors try to minimize the power consumption of a specific section of chip and thus to balance heat distribution, however, no run-time mapping strategy is either presented in their work. On the whole, thermal aware mappings stay within upper bound of power budget and avoid hot-spots, but do not utilize the available budget effectively, neither improve it.

The phenomenon of dark silicon and effects of utilization wall were identified in pioneering works in [4][28]. Thus far,

most common practice of ensuring safe chip functionality has been by estimating TDP [11] in a conservative manner. Recent upgrades on AMD and Intel Corporations' CPUs have the option of a configurable TDP, however they are limited to a maximum of 3 modes, without any fine grained control [29][30]. Pagani *et al.* have proposed an adaptive way of setting the upper bound on power consumption by expressing it as a function of simultaneously active cores in [10]. They provide a light weight C library to estimate TSP for a given mapping and also the worst case TSP for a given number of active cores. Despite these, they do not provide any insight on which mapping would offer a better power budget. The importance of spatial alignment of dark cores along with active cores and its effects on power budget have been presented in [12]. It shows gain of patterning in two perspectives viz., better power budget and lower operating temperatures. [13] quantitatively showed that different patterns of dark silicon result in different power budgets and temperature profiles of the chip. However, both these works do not propose any method on how to pattern the dark tiles to get such higher power budgets. Taking all these things together into perspective, it is necessary for a run-time mapping algorithm that patterns active and dark cores to maximize utilization of power budget, considering dark silicon.

III. Motivation

The heat dissipated by a core C_i 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. When active cores function at full throttle, they dissipate power and heat that is proportional to the power. If temperature goes beyond the safer limit, chip's functionality would fail permanently. Generally, dynamic thermal management techniques such as clock and power gating, voltage and frequency down scaling, increased fan speed etc., are triggered to manage any such sudden phases of chip's overheating. This would reduce activity inside the chip and let the chip back to steady state temperature over a period of time. Although uneven heat distribution can be managed with these, it hampers the performance by a great deal. A better way for even distribution of heat is to pattern the dark cores along with working cores through runtime mapping.

The impact of spatial alignment of active cores on power budget is explained through a motivational example, presented in Figure 2. Three applications App1, App2 and App3 with 9, 12 and 7 tasks respectively are assumed to be running on the system. Conventional mapping approaches offer lower inter-task communication latency by greedily mapping all the applications contiguously. Mapping of the 3 applications contiguously on a NoC-based many-core system with 144 cores is shown in Figure 2(a). The power budget (TSP) of this system as computed by TSP library is 66W. A non-contiguous and spread-out mapping of the same applications (as well tasks) is shown in Figure 2(b). This mapping 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. Contiguous mapping avoids dispersion, but it leads to poor thermal profile of the chip due to prorogation of heat among neighboring applications and tasks and thus resulting in lower power budget. Contrastingly, spatially distributed mapping of applications offers higher power budget as effect of heat among different applications is negligible. Also, active

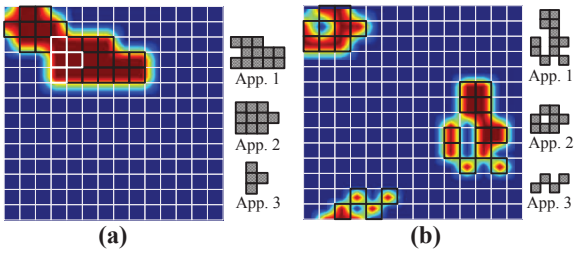


Fig. 2: Thermal profiles of contiguous and spatially distributed mappings

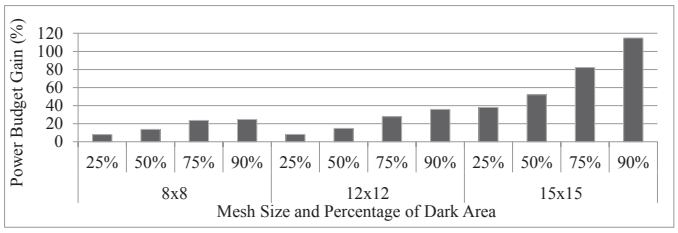


Fig. 3: Surplus power budget with increasing dark silicon

cores are patterned along with inactive cores such that heat effects of neighboring cores running the same application are minimized. Based on the spatial alignment and a minor compromise on dispersion, we could gain up to 13% of power budget. It is 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. In the mapping presented in Figure 2(a), the budget of 66W is used to power 22 cores, giving a per-core budget of 3W. Assuming a per-core budget of 3W, the mapping configuration in Figure 2(b) could power 74.6/3 = 24.7 cores. The surplus budget we could gain via patterning could thus be used to power up 2.7 more cores, reducing dark silicon by 12.2%. This benefit could be better realized with increased size of the many-core system, technology node scaling and more dark silicon which is expected in near future. The average packet latency in each of the patterned mappings for App1, App2 and App3 is 1%, 3.5% and 3.8% more than that of the contiguous mappings. Although there is a minor penalty in terms of latency, the gain in power budget outweighs the odds against it.

Figure 3 shows the surplus power budget that was gained for different mapping configurations over the worst case TSP budget. We ran random mappings with fixed amount of inactive (dark) cores ranging from 25% to 90% of darkness, over mesh sizes of 8x8, 12x12 and 15x15 with 22nm technology. For the same number of active cores, we collected the worst case TSP budgets. The difference between a random configuration and the worst case budget is presented as percentage gain in power budget. It is evident that the gain in power budget increases with increase in amount of dark silicon, to an extent where we can get a surplus of up to 114% when 90% of core is dark. It is to be noted that this surplus is purely from a random mapping generated to represent the power budget gain that can possibly be achieved, although the realistic gain could be variable.

IV. The Proposed Mapping and Patterning Approach

A. System Architecture

The top level abstraction of the system we implemented in our approach is shown in Figure 4. Application Repository

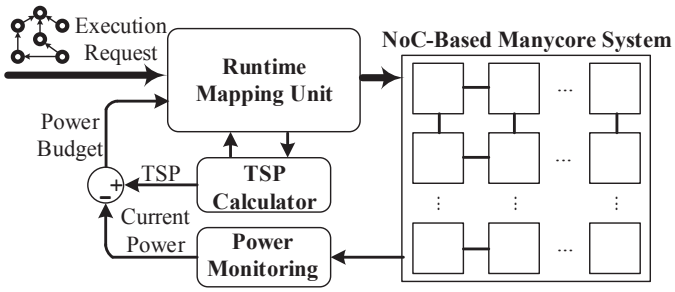


Fig. 4: Architecture Implementing the proposed mapping strategy

holds the applications modeled as task graphs and are released onto the system for execution over time. Runtime Mapping Unit (RMU) monitors the power profile of incoming applications issued by Application Repository, to ensure that upper bound on power budget is not violated by mapping a new application. RMU estimates the power of incoming application and checks if the chip currently has enough power budget to run the new application. An example of such estimation can be found in [3]. The application is forwarded onto the system if there is available budget. In case of un-availability, the application waits until the system can allocate enough budget, perhaps with currently running application(s) leaving the system after finishing their execution. TSP Calculator receives current mapping configuration of the system as input and computes the realistic upper bound on power budget, the thermal safe power (TSP). The RMU feeds this new budget value to the chip and updates the maximum power budget of the chip to the TSP provided by the TSP Calculator.

Application mapping is to find a free region for an application on the chip, which is of polynomial time complexity [31]. However, we have constraints on finding an optimal region such as power consumption, communication latency, dispersion, patterning the dark cores etc., making mapping an NP-hard problem. Surplus budget gained from mapping in Figure 2(b) can be attributed to the mapping policy. Two major factors that distinguishes it from contiguous mapping are spatial distribution of applications and sparsity among tasks of each individual application. 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.

Finding an optimal region for an application starts with finding an optimal node, the *first node*, around which an application can be mapped. In our approach, we prioritize nodes (thus regions) that are spatially far from currently active cores as *first nodes*. Tightly packed up active cores are the major reason for hop-spots which eventually lead to under utilization of power budget. Hence, after region selection, we map tasks of the application in a sparse pattern that will reduce the probability of heat accumulating at specific regions that potentially turn out as hot-spots. Our approach tries to attain an even distribution of heat at both region selection (through optimal first node selection) and mapping (through patterning).

B. First Node Selection

In contrast to reactive strategies, we exploit MapPro [9] which pro-actively calculates an optimal first node for incoming applications of every possible size by assigning the square that can fit the application. This way, we totally eliminate the time spent in first node selection, which automatically reflects in the overall execution time. The choice of a particular first node and thus a particular square region is based on our objective to find a region that has: i) free nodes to allocate for incoming application with minimal internal congestion and ii) minimal effect in terms of temperature on other regions. To quantify these objectives, we use Vicinity Counter (*VC*) defined in [9] which addresses the aforementioned first objective. We then combine it with a new parameter viz., Distance Factor (*DF*) which addresses the second objective.

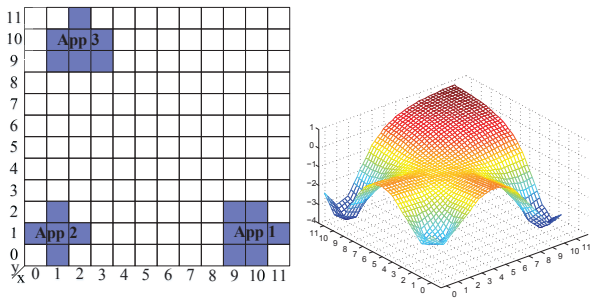
VC parameter expresses the number of free nodes, which also represents the number of occupied nodes. Using this parameter, we consider the affect of occupied nodes in the region on internal congestion, by quantifying the location of occupied nodes. For instance, a node that is occupied in the inner most square close to the first node has more affect on internal congestion than the ones that are occupied in outer squares, far from the first node. Therefore, we quantify this by pegging the weight of an occupied node with its distance from the central node, by assigning a higher penalty to occupied nodes closer to central node and relatively lower penalty to the ones that are far. The VC value represents the availability and congestion around a chosen node, and makes it easier to select between nodes with the same VC value, by considering congestion.

Definition: *Distance Factor, DF_{ij}, 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_{ij} is the weight of node n_{ij}, d_{ij-xy} is distance from nodes located at (i,j) and (x,y) and α is the mesh size. Thus the DF of a node represents the effect of heat from concurrently running applications on the chip. This helps in selection of a region that is less probable to generate any potential hotspots. In addition, the issue of two or more nodes having a same VC value can be resolved by examining their corresponding DF values, and the node with the lower (best) DF value is chosen.

When an un-occupied node becomes occupied by a new task, it starts dissipating power and thus heat. Initially, the heat is concentrated on the node itself, over time it starts impacting its neighbors. However, the effect of heat dissipated by this node on neighboring nodes gradually decreases as we move towards farther nodes. Inspired by the surface tension phenomenon [32], where energy distribution of a flat surface turns into a curved surface with applied surface pressure, we model the effect of heat transfer 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, greater the distance from an active core, lesser the effect of heat from it. We illustrate this effect in Figure 5, for the chip running 3 applications App1, App2 and App3. The regions where applications are mapped (deeper zones) have a DF as low as -4, while the regions that are far away (shallow zones) from them have a DF of 1. Also, as we traverse towards the far off shallow zones, the DF value improves indicating the



(a) x (b) y

Fig. 5: Effect of occupied cores on Distance Factor of un-occupied cores

importance of distributing the application spatially across the chip. This effect also depends on mesh size such that smaller mesh results in greater impact, due to more proximity among hot and its neighboring nodes. 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.

The algorithmic flow of first node selection is shown in Algorithm 1 which is inspired by our previous work presented in [9]. When a new application enters the system, power budget required for it is estimated. If the system has enough budget to be allocated for the new application, then the application request is serviced by finding required first node (lines 1-3). The node with maximum VC value, $maxVC$, among all nodes is chosen as the first node, and thus also selecting the 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 in Algorithm 2. The chip's mapping configuration changes after mapping one application. As a result, the VC and DF values for the remaining un-occupied nodes need to be updated for every newly occupied node. Once the application is mapped, we calculate the new node with maximum value of VC $maxVC$ for different radii and also update the DF values of un-occupied nodes. This way, we pro-actively calculate the first node for next incoming application and avoid the overhead caused in first node selection. VC values for square of different groups are updated (lines 6-23) to determine the new $maxVC$ node. In case of a conflict when more than one node has the same $maxVC$, it is resolved by comparing their DF values (lines 16-18). Thus the VC and DF values are pro-actively updated with the intention of servicing the next incoming application immediately, without any overhead. Nodes are released from the system when an application leaves after finishing its execution. The VC and DF values are once again updated according to the changed mapping configuration of the chip, with the exit of an application.

C. Dark Silicon Patterning

An optimal region for mapping an incoming application is chosen via first node selection. The mapping receives the first node (fn) as an input and builds a polygon P which is the set of all un-occupied nodes around the selected first node. We choose nodes among the region P such that the tasks are run on nodes that are sparsely aligned. Sparsity of a node n_{ij} represents the number of free nodes that are neighboring

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 = $[(\sqrt{M}-1)/2]$,$maxRadius$:Maximum radius of the square ($(\sqrt{M}-1)/2$), α:Mesh size parameter $\sqrt{M}$,P_{avg}: 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: $firstNode \leftarrow maxVC_{[(\sqrt{appSize}-1)/2]}$
4: $Q \leftarrow pattern(firstNode, newApp)$;
5: //Updating VC and DF values after mapping
6: **for** each $n_{xy} \in newApp$ **do**
7: **for** each core n_{ij} located in Row i and Column j **do**
8: r' = $maximum(|i-x|, |j-y|)$;
9: $DF_{ij} -= e^{-\alpha r'}$;
10: **for** r = 1 to $maxRadius$ **do**
11: **if** $r - r' \geq 0$ **then**
12: VC^r_{ij} -= r - r';
13: **if** $VC^r_{ij} > maxVC_r$ **then**
14: $maxVC_r \leftarrow VC^r_{ij}$;
15: **else**
16: **if** $VC^r_{ij} = maxVC_r$ and $DF_{ij} > DF_{maxVC_r}$ **then**
17: $maxVC_r \leftarrow VC^r_{ij}$;

it in all four cardinal directions (North, East, West, South). The Sparsity Factor (SF_{ij}) for a node n_{ij} located at (i, j) is expressed as:

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

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

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

Based on the Sparsity Factor, we prioritize the nodes that are more sparse. Sparse nodes ensure that they have less effect on neighbor's temperature. As a result, the dense nodes are assigned least priority and are 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 communication volume. We choose the task with highest communication volume and map it on to the first node, the most sparse node. We proceed to the task with next highest communication expense and map it onto the node with the highest SF among the nodes in the selected square S. We continue similarly, in order of communication of tasks and sparsity of nodes so that tasks with higher communication gets mapped onto nodes with higher sparsity, until all tasks of the application are mapped. This way, the less sparse nodes (i.e., the dense nodes) which are tightly packed with active cores as neighbors gets least priority. These denser nodes eventually remain un-occupied among the other occupied nodes of the region, minimizing the probability of creating potential hot-

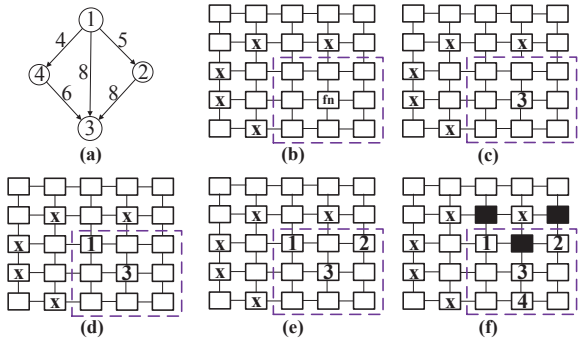


Fig. 6: Example for pattern based mapping

spots.

The patterning is explained through an example presented in Figure 6. Assuming an application with 4 tasks, as in Figure 6(a), has arrived, the selected first node fn and the corresponding square region is shown in Figure 6(b). The most expensive task of the applications is task 3, which is mapped on most sparse node, the fn. This automatically effects the SF values of nodes surrounding it. The next in order of communication is tasks 1, and the next node with higher SF is chosen and task 1 is mapped on it. In the similar fashion, the remaining tasks are mapped subsequently, as in Figure 6(c)-(f). It is to be observed that after the application is mapped, one node in the square is left such that it has SF = 0, which would be of least priority for any incoming application. On the same lines, there are other nodes with SF = 1, which also attained a relatively lower priority for getting mapped. These nodes are the candidates that would potentially be dark, and provide the cooling effect needed by their active neighbors. Thus, we pattern the dark cores among active cores to minimize the probability of heat getting accumulating at any single point on the chip. Conversely, had all the tasks of the application been mapped contiguously, they would have reached their critical temperatures by consuming only lower amount of power and eventually trigger dark cores else where on the chip. However, in the case where we patterned the inevitable dark cores, the cores will reach critical temperatures only after utilizing available power budget to a better extent, or offer more budget to activate more cores.

The algorithmic flow of patterning is shown in Algorithm 2. Patterning starts with a selected first node, $firstNode$ and chooses the square region ($Square^{size}_{firstNode}$) that can fit the application (App) (line1). The application is sorted into tasks ($Tasks$) as per their communication volume (line 2). The most expensive task is mapped onto the node with maximum Sparsity Factor, $maxSF$ (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. Intuitively, the SF value for nodes in selected square changes with every occupied and thus new $maxSF$ is computed for every unmapped task.

V. Evaluation

We simulated our proposed mapping strategy over applications modeled as task graphs of different sizes ranging from 4 to 35 tasks, generated using [33]. Communication volumes among these tasks are distributed randomly using Gaussian elimination. We simulated traffic patterns of these

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

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 [34] are used as the baseline for processing elements. The communication network infrastructure between processing elements is provided by a pruned version of Noxim [35] that uses mesh topology and XY routing. Parameters related to technology node scaling are extracted from Lumos framework [36], an open source framework which quantifies power-performance characteristics of many-core systems with technology node scaling. We used TSP library [10] to calculate the Thermal Safe Power. Proposed dynamic mapping is implemented by a Central Manager (CM), which is the node $n_{(0,0)}$ of the mesh in our many-core platform. In the many-core platform we implemented, (overview as in Figure4), a random sequence of applications enter the system and are buffered into a FIFO. Applications are serviced in a first-come-first-serve policy, subject to availability of enough power budget. If enough power budget can be allocated, the application is scheduled by the Central Manager (CM). A suitable first node for mapping the incoming application is chosen by the (CM), followed by mapping all the remaining tasks of the application.

We evaluate our dark silicon patterning approach (from here on referred to as *PAT*) against the combination of *SHiC* [8] and *CoNA* [19] (from here on referred to as *SC*), for first node selection and mapping respectively. These two approaches are state-of-the-art strategies that prioritize regions with free nodes and contiguity among nodes selected for mapping, with the primary objective of minimizing communication latency. Since we relaxed the constraint on contiguity among individual applications, we chose to compare against these works to quantify the effect of our patterning approach against contiguity. In view of applications that require a conservative upper bound on power, we also compare PAT against worst case power budget that can be offered with a given number of active cores (TSP_{wc}). Given a fixed number and sequence of applications, we compare power budgets offered by different mappings based on different mapping strategies. The entry sequence of applications is maintained the same for different mapping approaches for a fair comparison. In case of TSP_{wc}, we compare the power budget given by [10] for the number of active cores in the mapping generated by *PAT*. We run our simulations over different network sizes of 16×16 and 20×20. In addition, we emulate a varying dark silicon behavior

ranging from 50% darkness through 90% by adjusting initial upper bound on power consumption (TDP). We set the TDP to 177.7W and 277.7W respectively for 16×16 and 20×20 network sizes using 22nm technology, keeping the power density constant [36]. ITRS projections present the fact that by the year 2020, computer systems would face 90% dark silicon and that many-core platforms would be in upwards of 512 cores to gain maximum peak performance [4]. Hence, we considered the case of the chip being 90% dark, while also including contemporary projection of 50% dark area. We limit the number of applications entering the system to be in accordance with dark areas. We evaluate the proposed approach over power budget provided per mapping and corresponding throughput. The power budget is computed using TSP library for every mapping, traced with entry of a new application. The ambient temperature is set to 45°C and the safe operating temperature beyond which chip's functionality fails is set to 80°C.

The average (arithmetic mean) and best case percentage gains in power budgets of different mapping configurations using *PAT* strategy over *SC* for different network sizes are presented in Table I. *PAT* achieves a surplus power budget when compared to *SC*, as the active cores are optimally arranged, balancing heat distribution across the chip. In contrast, *SC* tries to map contiguously, leading to tightly packed active cores which get heated up already at lower power consumption, resulting in a lower power budget. In addition to a better power budget, the chip always operates under safe peak operating temperature (80°C), since the budgets are computed through TSP library which manages the upper bound on power avoiding hazardous hotspots. It can be observed that the surplus budget achieved in case of *PAT* increases with increase in amount of dark silicon on the chip. With 90% of the chip being dark, utilizing the remaining fewer number of cores that can originally be powered (active) becomes crucial. *PAT* performs better in such scenarios, given the wider choice of dark cores that can be patterned, while *SC* remains dark silicon agonistic. The gain also increases with increase in network size, once again due to increase in scope of the chip area that can be patterned. Moreover, this is in line with the power budget gain obtained for randomly distributed tasks compared to worst case budget generated by TSP library, as in Figure 3.

In view of future applications like big data, artificial intelligence etc., requiring many-core platforms of larger network sizes, and the issue of dark silicon predicted to grow worse, patterning becomes quintessential to extract higher performance as expected from a many-core system. The potential of patterning could be further better realized with highly scaled up networks and subsequent increase in dark silicon. Nevertheless, the gain in power budget can still be realized at contemporary many-core platforms of relatively smaller network sizes. We simulated 12×12 network for 90% dark area to observe significant gain in power budget in case of *PAT* compared to *SC* and TSP_{wc}. The average and best case (BC) gain in power budget and throughput achieved by *PAT* over *SC* and TSP_{wc} are shown in Figure 7. Some of the low power and safety critical applications follow a strict and conservative approach on setting a single upper bound on power budgets. We account for such applications by comparing our proposed approach against power budget that could result from a worst case mapping configuration for a given number of active cores, generated by [10]. The surplus gained when compared to worst

TABLE I: Surplus Power Budget (in %) of PAT over SC

Network Size	90% dark		75% dark		50% dark	
	Avg.	Best	Avg.	Best	Avg.	Best
16×16	5.74	13.9	4.15	11.3	2.19	7.68
20×20	6.54	17.17	5.06	8.55	2.63	4.28

TABLE II: Surplus Power Budget of PAT over TSP_{wc}

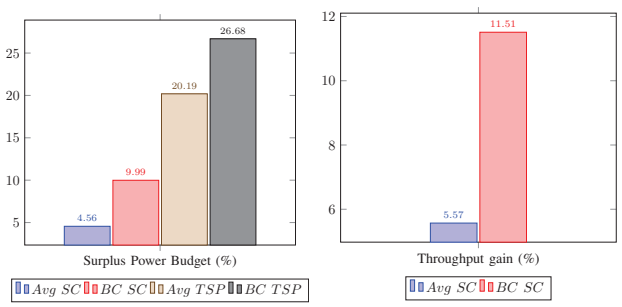
Network Size	90% dark		75% dark		50% dark	
	Avg.	Best	Avg.	Best	Avg.	Best
16×16	32.33	34.92	22.02	24.14	11.73	13.20
20×20	38.70	40.83	22.40	27.4	12.50	13.33

case power budgets for different network sizes and darker chip areas is shown in Table II. Understandably, PAT offers better power budget against worst case values. Although the comparison is against worst case budgets, it still establishes the impact of patterning on mitigating dark silicon to a large extent.

Since a surplus in power budget is gained through *PAT*, it can be utilized to power up more number of cores without violating any safe upper bound on power consumption, which reflects in throughput. The proposed first node selection method chooses diverse regions for different applications which has no impact on individual application's latency. However, few applications using *PAT* might have a lower individual latency at times, due to the compromise on contiguity among tasks of a patterned application. Despite the occasional latency, the overall throughput of the system using PAT still remains higher compared to that of *SC*, as the gain achieved in terms of power budget would (over) compensate for the latency. Gain in throughput for *PAT* compared to *SC* for different mesh sizes and dark regions is presented in Table III. Throughput gain depends largely on surplus budget gained, which in turn can be used to activate more cores. Thus, throughput achieved using PAT strategy follows a similar trend to that of surplus power budget achieved.

TABLE III: Throughput gain for PAT over SC

Network Size	90% dark		75% dark		50% dark	
	Avg.	Best	Avg.	Best	Avg.	Best
16×16	7.27	15.64	4.59	13.92	2.42	8.58
20×20	8.5	20.99	5.88	10.21	2.89	4.54



(a) Surplus Power Budget (in %) (b) Gain in throughput

Fig. 7: Surplus budget and throughput gain of PAT over SC and TSP_{wc} for 12×12 mesh

VI. Conclusion and Future Work

In this paper, we presented a dark silicon aware run-time mapping strategy for achieving a better power budget. We implemented our proposed approach in two phases viz., first node selection and patterning based mapping, where we evenly distribute tasks across the chip area to balance heat distribution. This lets the active cores to utilize relatively more power before they reach a maximum limit beyond which chip's functionality is harmed due to thermal violation. As a result, different applications utilize surplus power budgets better to activate more cores and thus gain in performance. Noticeably, the surplus power budget reflects in better resource utilization and throughput, while ensuring thermal safety of the chip, using a software run-time technique, with no hardware overhead. We observed that gain in terms of power budget and throughput increases with increase in network sizes and amount of dark silicon on the chip, stressing the importance of dark silicon aware mappings moving into the future workload characteristics and increase in dark silicon.

We considered our many-core platform to be homogeneous, although having a heterogenous combination of cores with different power-performance characteristics could have more potential. The surplus budget could be better used to activate even more number of cores at lower frequencies and also to run heterogeneous workloads on cores that suit them. Implementing our technique and allocation of surplus budget for a heterogeneous many-core platform is planned for future work.

Acknowledgment

The authors acknowledge the financial support by the Academy of Finland project entitled "MANAGE: Data Management of 3D Systems for the Dark Silicon Age", University of Turku graduate school (UTUGS) and EU COST Actions IC1103: Manufacturable and Dependable Multicore Architectures at Nanoscale (MEDIAN). The authors want to thank Mr. Mohammad Fattah for setting up NoC based system simulator with dynamic mapping feature. The authors also mention that they used the source code of Lumos from University of Virginia to extract the power data for different technologies.

References

[1] N. Goulding-Hotta et al. The GreenDroid Mobile Application Processor: An Architecture for Silicon's Dark Future. *IEEE Micro*, 31(2), 2011.

[2] Joonho Kong et al. Recent thermal management techniques for microprocessors. *ACM Comput. Surv.*, 2012.

[3] M.-H. Haghbayan et al. Dark Silicon Aware Power Management for Manycore Systems under Dynamic Workloads. In *ICCD*, 2014.

[4] H. Esmaeilzadeh et al. Dark Silicon and the End of Multicore Scaling. *IEEE Micro*, 32(3), 2012.

[5] Semiconductor Industry Association et al. International technology roadmap for semiconductors (ITRS), 2011 edition. 2011.

[6] A.-M. Rahmani et al. Dynamic Power Management for Many-Core Platforms in the Dark Silicon Era: A Multi-Objective Control Approach. In *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2015.

[7] Carvalho de Souza et al. Dynamic task mapping for MPSoCs. *Design & Test of Computers, IEEE*, 27(5):26–35, 2010.

[8] M. Fattah et al. Smart hill climbing for agile dynamic mapping in many-core systems. In *DAC*, 2013.

[9] M.-H. Haghbayan et al. Mappro: Proactive runtime mapping for dynamic workloads by quantifying ripple effect of applications on networks-on-chip. In *International Symposium on Networks-on-Chip (NOCS)*, 2015.

[10] S. Pagani et al. TSP: Thermal Safe Power: Efficient Power Budgeting for many-core systems in dark silicon era. In *Proc. of CODES+ISSS*, 2014.

[11] Intel Corporation. Intel Xeon Processor - Measuring Processor Power, revision 1.1. In *White paper, Intel Corporation*, April, 2011.

[12] M. Shafique et al. Dark Silicon As a Challenge for Hardware/Software Co-design. In *Proc. of CODES+ISSS*, pages 13:1–13:10, 2014.

[13] Muhammad Shafique et al. The EDA challenges in the dark silicon era. In *Proc. of DAC 2014*, pages 1–6.

[14] Chen-Ling Chou and R. Marculescu. Contention-aware application mapping for network-on-chip communication architectures. In *Proc. of ICCD*, pages 164–169, 2008.

[15] M. Fattah et al. Adjustable contiguity of run-time task allocation in networked many-core systems. In *ASP-DAC*, pages 349–354, 2014.

[16] 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*, 27(10), 2008.

[17] A.M. Bender et al. Communication-aware processor allocation for supercomputers: Finding point sets of small average distance. *Algorithmica*, 50(2):279–298, January 2008.

[18] E. Carvalho et al. Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs. In *RSP*, pages 34–40, 2007.

[19] M. Fattah et al. CoNA: Dynamic application mapping for congestion reduction in many-core systems. In *Proc. of ICCD*, pages 364–370, 2012.

[20] M. Deveci et al. Exploiting geometric partitioning in task mapping for parallel computers. In *Proc. of Parallel and Distributed Processing Symposium, 2014*, pages 27–36. IEEE, 2014.

[21] S. Shintaro et al. Pattern-based systematic task mapping for many-core processors. In *International Conference on Networking and Computing*, pages 173–178. IEEE, 2010.

[22] A. Coskun et al. Temperature management in multiprocessor socs using online learning. In *Proc. of DAC*, pages 890–893. IEEE, 2008.

[23] L. Thiele et al. Thermal-aware system analysis and software synthesis for embedded multi-processors. In *Proc. of DAC*, pages 268–273. ACM, 2011.

[24] Y. Liu et al. Energy and thermal aware mapping for mesh-based noc architectures using multi-objective ant colony algorithm. In *Proc. of ICCRD*, volume 3, pages 407–411, 2011.

[25] A. Coskun et al. Utilizing predictors for efficient thermal management in multiprocessor socs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(10):1503–1516, 2009.

[26] Min Bao et al. Temperature-aware task mapping for energy optimization with dynamic voltage scaling. In *Proc. of DDECS*, pages 1–6, 2008.

[27] T. Komoda et al. Power capping of CPU-GPU heterogeneous systems through coordinating DVFS and task mapping. In *ICCD*, pages 349–356, 2013.

[28] MB Taylor. Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse. *Proc. of DAC, 2012*, pages 1131–1136.

[29] Intel Corporation. Fourth Generation Mobile Processor Family Data Sheet. In *White paper, Intel Corporation*, July, 2014.

[30] AMD. AMD Kaveri APU A10-7800. Accessed: 2015-02-28.

[31] Dobkin et al. Searching for empty convex polygons. *Algorithmica*, 5(1-4):561–571, 1990.

[32] Harvey E. White. *Modern College Physics*. Van Nostrand, 1948.

[33] TGG: Task Graph Generator. *URL: http://sourceforge.net/projects/taskgraphgen/*, 2010.

[34] S. Li et al. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *MICRO*, 2009.

[35] F. Fazzino et al. Noxim: Network-on-chip simulator. *URL: http://sourceforge.net/projects/noxim*, 2008.

[36] 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.